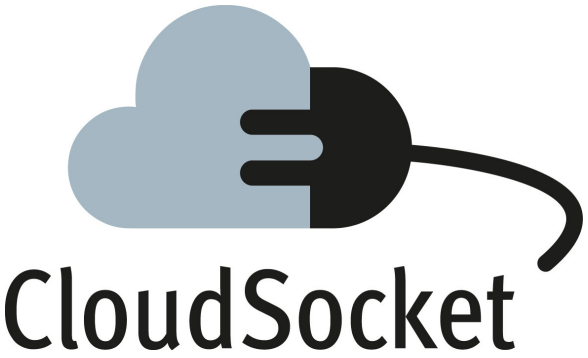


FINAL CLOUDSOCKET ARCHITECTURE

D4.5

Editor Name	Robert Woitsch (BOC)
Submission Date	September 30, 2016
Version	1.0
State	FINAL
Confidentially Level	PU

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

EXECUTIVE SUMMARY

This document introduces the final CloudSocket architecture that consists of loosely coupled, exchangeable and partly optional environments. The BPaaS Environments that supports the BPaaS lifecyle are: (a) the BPaaS Design Environment, (b) the BPaaS Allocation Environment, (c) the BPaaS Execution Environment, and (d) the BPaaS Evaluation Environment. Additionally, (e) the BPaaS Marketplace is required to enable the customer to buy the BPaaS.

Each environment is defined by a set of functional capabilities and a data exchange format to facilitate the exchange of environments with similar solutions and hence introduce flexibility into the architecture, which enables resizing CloudSocket to fit the CloudSocket Brokers' needs as well as avoiding any vendor lock. In addition to the basic set of functional capabilities offered, each environment may exploit innovative research functionality, originating from WP3, which can provide added value (e.g., introduction of semantics to automate the business process to workflow mapping). As such, such functionality can be considered as either an add-on to the existing capabilities or can be adopted and incorporated into them.

The BPaaS Design Environment introduces the user interface to design domain specific business processes, to perform the semantic lifting and process analysis of those processes as well as to specify an executable workflow. High level deployment rules can be expressed as decisions in DMN or by expressing cases. Semantic annotations are partly expressed in extensions of business process models and partly as ontologies. Key Performance Indicators (KPIs) are also specified indicating high-level non-functional business requirements. All this information is packaged in BPMN, DMN, OWL-Q and RDF format in a so-called BPaaS Design Package and handed over to the BPaaS Allocation Environment.

The BPaaS Allocation Environment adds cloud deployment information to the resulting BPaaS Bundle and hence introduces several deployments in a cloud environment. The workflow related BPMN part is extended with bundle information enabling both (a) the access to already deployed (external) services as well as (b) the provisioning of deployable packages for (internal) services. The construction of workflows in the cloud is a manual task using the interfaces of the BPaaS Allocation Environment by manually reading the provided business process & workflow models, decision models and semantic descriptions and by selecting appropriate cloud offerings, to hand over a complete BPaaS Bundle in a format based on an extension of CAMEL. Such a bundle also includes information that can drive the adaptive provisioning of the BPaaS in the form of SLO requirements (involving conditions on metrics) and adaptation rules.

The BPaaS Execution Environment is the most complex environment comprising two major parts. First, the BPaaS Marketplace registers and publishes the BPaaS Bundles. This Marketplace is based on a SaaS marketplace but instead of a single step selection of a SaaS offering, it provides selection assistance, where first, the domain specific artefacts from the business process and second the technical details are selected. Authentication, identification and service registry are specialised facilities handled and offered by the marketplace to the other environments.

Behind the Marketplace, there lies the core functionality of the BPaaS Execution Environment incarnated via a Workflow, Cloud Provider, Monitoring, Adaptation and SLA Engines. The Cloud Provider Engine checks if all required services are already deployed or if a deployment on request is necessary. When the Cloud Provider Engine is finished, it hands the workflow deployed in the production cloud over to the Workflow Engine, which creates and executes the user instance(s) of the concrete workflow. A Monitoring Engine monitors and aggregates across clouds

Copyright © 2016 BOC and other members of the CloudSocket Consortium

and layers the BPaaS performance and hands it over to the Adaptation and SLA Engines. Cross-cloud BPaaS reconfiguration is handled by the Adaptation Engine in the form of triggering adaptation rules. The SLA Engine observes the measurements produced by the Monitoring Engine and assesses corresponding SLO conditions, thus being able to follow and visualise the actual status of the SLA agreements between the BPaaS Broker and Customer.

The BPaaS Evaluation Environment draws monitoring and logging information from the Execution Environment, semantically enhances it and stores it in the Semantic Repository. As such, such information can then be queried to assess KPIs, or used to produce a business process intelligence knowledge in the form of best BPaaS deployments, adaptation rule suggestions and process mining reports. All derived/analysed information is merged with business process and key performance indicator models – a so-called model "assimilation" of log/evaluation information. This enables the abstraction back onto the level of the domain-specific business process.

DOCUMENT HISTORY

This document is an update of the first CloudSocket architecture published as D4.1 (D4.1 2015).

For completeness reasons, the text of the original document has been used as the basis for this document and has been updated where necessary. Hence large parts of the document are identical with the initial version D4.1. Updates have been performed by relying on: (a) lessons learned and user feedback after completing the first prototype, (b) gaining a more detailed understanding of novel parts during implementation as well as (c) including the research perspective of environments after completing the initial research cycle. However, the remaining research cycle will continue to contribute to the architecture.

The first prototypes of the individual environments have been developed, published on the CloudSocket website and documented in the joint deliverable D4.2, D4.3, D4.4 (D4.2_4.3_4.4 2016). This joint deliverable provided a fact sheet on the available prototypes, which has been copied into this document as chapter 2 to accompany the architecture description with an overview where to download and how to install the various environments.

Research components that have been introduced into the final architecture are described in more detail in the research deliverables D3.1 (D3.1 2015), D3.2 (D3.2 2016) and D3.3 (D3.3 2016). Condensed text of relevant versions have been introduced within the corresponding environment chapters as an own section.

As most part of the text is the same as in D4.1, the contributors and reviewers have been added to the original list of authors.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

PROJECT CONTEXT

Workpackage	WP4: BPaaS Implementation
Task	T4.1: Architecture and Design of CloudSocket
Dependencies	Development of first Architecture

Contributors and Reviewers

Contributors D4.1	**Reviewers D4.1**
Robert Woitsch, Mehmet Albayrak, Harald Kühn, Wilfrid Utz (BOC), Ana Juan Ferrer, Joaquin Iranzo (ATOS), Antonio Leonforte, Antonio Gallo (FHOSTER), Vlad Mihnea, Remus Pacurar, Calin Avasilcai, Gheorghe Arama, Roxana Boca (YMENS), Frank Griesinger, Daniel Seybold, Jörg Domaschka (UULM), Kyriakos Kritikos, Dimitris Plexousakis (FORTH)	Robert Woitsch (BOC), Antonio Leonforte (FHOSTER), Frank Griesinger (UULM), Kyriakos Kritikos (FORTH), Vlad Mihnea, Remus Pacurar (YMENS),
Contributors D4.5	**Reviewers D4.5**
Robert Woitsch (BOC), Damiano Falcioni (BOC), Wilfrid Utz (BOC), Roman Sosa, Joaquin Iranzo (ATOS), Mihai Pavelescu (YMENS), Simone Cacciatore, Antonio Gallo (FHOSTER), Frank Griesinger, Daniel Seybold (UULM), Kyriakos Kritikos (FORTH), Emanuele Laurenzi, Benjamin Lammel, Knut Hinkelmann (FHNW)	Robert Woitsch (BOC), Joaquin Iranzo (ATOS), Mihai Pavelescu (YMENS), Simone Cacciatore, Antonio Gallo (FHOSTER), Frank Griesinger, Daniel Seybold (UULM), Kyriakos Kritikos (FORTH), Emanuele Laurenzi (FHNW)

Approved by: Joaquin Iranzo Yuste [ATOS], as WP 4 Leader

Version History

Version	Date	Authors	Sections Affected
0.1	August 31, 2015	Robert Woitsch (BOC), Joaquin Iranzo (ATOS)	All
0.4	September 21, 2016	all contributors	Update based on lessons learned, and WP3 results. Approval at partner meeting
0.5	September 28, 2016	Robert Woitsch (BOC), Joaquin Iranzo (ATOS)	Compilation of Review Version
0.9	September 30, 2016	all contributors	Review approval in telephone conference.
1.0	September 28, 2016	Robert Woitsch (BOC), Joaquin Iranzo (ATOS)	Final Approval

Copyright © 2016 BOC and other members of the CloudSocket Consortium

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/

Copyright © 2016 BOC and other members of the CloudSocket Consortium

TABLE OF CONTENT

1 Introduction and Problem Statement 15
2 First Prototype IntroduCtion 17
2.1 Implementation Approach 26
3 Architectural Paradigm 27
3.1 Loose coupling of Building Blocks 27
3.2 Model Driven and Layered Alignment Approach 28
3.3 Data Exchange 29
3.4 Interfaces 29
4 User Scenarios and High Level Architecture 30
4.1 Use Case and Stakeholder Requirements 30
4.1.1 Business Incubator Scenarios: A Recap 30
4.1.1.1 Ecological Agriculture 30
4.1.1.2 Green Energy 30
4.1.2 Cluster Process Broker Scenario: A Recap 31
4.1.2.1 Internet Research and Procurement Process 31
4.1.2.2 Kiosk Distribution Process 31
4.1.3 The CloudSocket Customer 32
4.1.4 The CloudSocket Broker 32
4.2 Initial High Level CloudSocket Architecture 34
4.2.1 Key Functional Capability of BPaaS Environments 35
4.2.2 Data Interaction between BPaaS Environments 36
4.2.2.1 BPaaS Design Package 36
4.2.2.2 BPaaS Meta Data 36
4.2.2.3 BPaaS Bundle 36
4.2.2.4 BPaaS Monitoring 37
4.2.2.5 BPaaS Finding 37
4.2.3 BPaaS Security Layer 38
4.3 CloudSocket Usage Scenario 39
4.4 CloudSocket Ecosystem 42
5 BPaaS Design Environment 44

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.1 Introduction 44
5.2 Functional Capabilities 46
5.2.1 Business Process Design 47
5.2.2 Business Process Analysis 49
5.2.3 Semantic Lifting 51
5.2.4 Executable Workflow Design 53
5.2.5 KPI Definitions and Meta Model Completion 56
5.3 Components 58
5.3.1 User Interface Layer 58
5.3.2 Modelling Layer 59
5.3.3 Meta Model Platform Layer 60
5.3.4 Executable Workflow Designer 61
5.4 Research Contribution 64
5.5 Roles 67
5.6 Data Interface 68
5.6.1 BPMN Interchange 68
5.6.2 KPI, Meta Data and Decision Model Interchange 68
6 BPaaS Allocation Environment 69
6.1 Introduction 69
6.1.1 Description and structure of a BPaaS Bundle 69
6.1.2 Creation of a BPaaS Bundle 72
6.1.3 Architectural overview 73
6.2 Functional Capabilities 73
6.2.1 Creation of a BPaaS Bundle 74
6.2.2 Atomic Service Allocation 76
6.2.3 Software Component Allocation 79
6.2.4 KPI Model Editing 82
6.2.5 SLA Model Editing 85
6.2.6 Pricing Model Editing 87
6.2.7 Business Process Metadata Editing 89
6.2.8 BPaaS Bundle Publishing in the Marketplace 91
6.3 Components 93

Copyright © 2016 BOC and other members of the CloudSocket Consortium

6.3.1 User-interface layer 93
6.3.1.1 Bundle Instantiator 94
6.3.1.2 Bundle Designer 94
6.3.1.3 Bundle Browser 95
6.3.2 Business-logic layer 96
6.3.2.1 Bundle Repository Manager 96
6.3.2.2 Bundle Manager 97
6.3.2.3 Bundle Publisher 97
6.3.3 Persistency-management layer 98
6.3.4 Component Diagram 98
6.3.5 Research Prototypes 98
6.3.5.1 Smart Service Discovery and Composition Tools 98
6.3.5.2 DMN-to-CAMEL-Mapper 100
6.4 Roles 101
6.5 Data Interface 102
7 BPaaS Execution Environment 103
7.1 Introduction 103
7.2 Functional Capabilities 104
7.2.1 Deployment of BPaaS 104
7.2.2 Execution of the BPaaS 108
7.2.3 Monitoring of Agreement Status 113
7.2.4 Workflow Environment Management 116
7.3 Components 120
7.3.1 User Interface workspace 120
7.3.1.1 Web UI / Workflow-Engine 120
7.3.1.2 SLA Dashboard 120
7.3.1.3 Cloud Provider Engine Dashboard 121
7.3.1.4 Monitoring Dashboard 121
7.3.2 BPaaS Middleware 122
7.3.2.1 Workflow-engine 123
7.3.2.2 SLA Engine 123
7.3.2.3 Monitoring Engine 124

Copyright © 2016 BOC and other members of the CloudSocket Consortium

7.3.2.4 Adaptation Engine 125
7.3.2.5 Cloud Provider Engine 126
7.3.2.6 Process Data Mediator 127
7.3.2.7 Component Diagram 128
7.3.2.8 Roles 129
7.3.3 Data Interface 130
7.3.3.1 Interface to Deploy the BPaaS Bundles. 130
7.3.3.2 Interface to Manage Service Level Agreements 131
7.3.3.3 Interface to publish the monitored information 131
7.3.4 Research Prototypes 131
8 BPaaS Marketplace 133
8.1 Introduction 133
8.2 Functional Capabilities 134
8.2.1 Publish BPaaS Bundle to Product Catalogue 134
8.2.2 Purchase BPaaS Bundle 136
8.2.3 Register New Customer User 139
8.2.4 Onboard Cloud Service Provider 142
8.3 Components 144
8.3.1 Customer UI Layer 144
8.3.1.1 Marketplace 144
8.3.1.2 Customer Portal 144
8.3.2 Cloud Broker Engine 145
8.3.3 Identity Management System 145
8.3.4 Cloud Provider Hub 148
8.3.5 Repository Manager 148
8.3.5.1 Atomic Service Registry 149
8.3.5.2 Software Component Registry 149
8.3.5.3 Cloud Provider Registry 149
8.3.5.4 Component Diagram 150
8.4 Roles 151
8.5 Data Interface 151
9 Evaluation Environment 152

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.1 Introduction 152
9.2 Functional Capabilities 153
9.2.1 KPI Analysis & Visualisation 154
9.2.2 Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules 157
9.2.3 Process Mining Analysis & Graphical Representation 161
9.3 Components 163
9.3.1 User Interface Layer 163
9.3.1.1 Hybrid Business Dashboard 163
9.3.2 Business Logic Layer 164
9.3.2.1 Conceptual Analytics Engine 164
9.3.2.2 Process Mining Engine 165
9.3.3 Data Layer 165
9.3.4 Component Diagram 166
9.4 Roles 166
9.5 Data Interface 166
10 Summary and Conclusions 167
11 References 168
12 Annex 171
12.1 BPaaS Bundle Sending Christmas Greeting Cards 171
12.2 WS-Agreement Sample 183

Copyright © 2016 BOC and other members of the CloudSocket Consortium

LIST OF FIGURES

Figure 1 - Work item definition table 26
Figure 2 Initial High Level Architecture 34
Figure 3 BPaaS Marketplace 41
Figure 4 - CloudSocket ecosystem source:(D8.1 2016), p14 43
Figure 5 Use Case Diagram – DE-UC-1-Business Process Design 48
Figure 6 Sequence Diagram – DE-UC-1-Business Process Design 48
Figure 7 Use Case Diagram – DE-UC-2-Business Process Analysis 50
Figure 8 Sequence Diagram – DE-UC-3-Business Process Analysis 50
Figure 9 Use Case Diagram – DE-UC-3-Semantic Lifting 52
Figure 10 Sequence Diagram – DE-UC-3-Semantic Lifting 52
Figure 11 Use Case Diagram – DE-UC-4-Executable Workflow Design 54
Figure 12 Sequence Diagram – DE-UC-4-Executable Workflow Design 55
Figure 13 Use Case Diagram – DE-UC-5-KPI Definitions and Meta Model Completion 57
Figure 14 Sequence Diagram – DE-UC-5-KPI Definitions and Meta Model Completion 57
Figure 15 BPaaS Design Environment – Domain Specific Business Process Designer User Interface Mockup 58
Figure 16 BPaaS Design Environment – Executable Workflow Designer User Interface Mockup 62
Figure 17 BPaaS Design Environment - Component Diagram 63
Figure 18: Overview of the BPaaS Design Environment and Smart Business IT-Cloud Alignment 64
Figure 19: Modelling Environment - Web Service Communication 65
Figure 20: Prototype Showing Matching Results 66
Figure 21 BPaaS Design Environment - Actors 67
Figure 22 BPaaS Bundle Elements 71
Figure 23 BPaaS Bundle States 72
Figure 24 Use Case Diagram – AE-UC-1 Creation of BPaaS Bundle 75
Figure 25 Sequence Diagram – AE-UC-1 Creation of BPaaS Bundle 75
Figure 26 Use Case Diagram – AE-UC-2-Atomic Service Allocation 78
Figure 27 Sequence Diagram – AE-UC-2-Atomic Service Allocation 78
Figure 28 Use Case Diagram – AE-UC-3-Software Component Allocation 81
Figure 29 Sequence Diagram – AE-UC-3-Software Component Allocation 81
Figure 30 Use Case Diagram – AE-UC-4-KPI Model Editing 83
Figure 31 Sequence Diagram – AE-UC-4-KPI Model Editing 84
Figure 32 Use Case Diagram – AE-UC-5-SLA Model Editing 86
Figure 33 Sequence Diagram – AE-UC-5-SLA Model Editing 86
Figure 34 Use Case Diagram – AE-UC-6-Pricing Model Editing 87
Figure 35 Sequence Diagram – AE-UC-6-Pricing Model Editing 88
Figure 36 Use Case Diagram – AE-UC-7 Business Process Metadata Editing 90
Figure 37 Sequence Diagram – AE-UC-7 Business Process Metadata Editing 90
Figure 38 Use Case Diagram – AE-UC-8-BPaaS Bundle Publishing in the Marketplace 92
Figure 39 Sequence Diagram – AE-UC-8-BPaaS Bundle Publishing in the Marketplace 92
Figure 40 BPaaS Allocation Tool – User Interface Screenshot 93
Figure 41 Allocation Tool – Component Diagram 98

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Figure 42: The architecture of the Smart Service Discovery and Composition Tools 99
Figure 43: The architecture of the Unified Service Discovery Tool 100
Figure 44 - DMN-to-CAMEL-Mapper Architecture 101
Figure 45 Use Case Diagram – EE-UC-1-Deployment of BPaaS 106
Figure 46 Sequence Diagram – EE-UC-1-Deployment of BPaaS 107
Figure 47 Use Case Diagram – EE-UC-2 – Execution of the BPaaS 110
Figure 48 Sequence Diagram – EE-UC-2 – Launch/Execute BPaaS instance 111
Figure 49 Sequence Diagram – EE-UC-2 – Reconfiguration environnement 112
Figure 50 Use Case Diagram – EE-UC-3 – Monitoring of Agreement Status 114
Figure 51 Sequence Diagram - EE-UC-3 – Monitoring of Agreement Status 115
Figure 52 Use Case Diagram - EE-UC-4 – Workflow Environment Management 118
Figure 53 Sequence Diagram – EE-UC-4 – Workflow Environment Management 119
Figure 54 BPaaS Execution Environment – Workflow Engine User Interface 120
Figure 55 BPaaS Execution Environment – SLA Monitoring Dashboard User Interface 121
Figure 56 BPaaS Execution Environment - Cloud Provider Engine Dashboard 121
Figure 57 BPaaS Execution Environment – Monitoring Dashboard User Interface Mockup 122
Figure 58 The unprocessed monitoring data in the Cloud Provider Engine Dashboard 122
Figure 59 BPaaS Execution Environment – Component Diagram 128
Figure 60 BPaaS Execution Environment – Actors 129
Figure 61 BPaaS Execution Environment – Interfaces 130
Figure 62 SLA Interface 131
Figure 63 Use Case Diagram – MP UC2 - Publish BPaaS Bundle to Product Catalogue 135
Figure 64 Sequence Diagram – MP UC2 – Publish BPaaS Bundle to Product Catalogue 135
Figure 65 Use Case Diagram – MP-UC2-Purchase BPaaS Bundle 137
Figure 66 Sequence Diagram – MP-UC2-Purchase BPaaS Bundle 138
Figure 67 Use Case Diagram – MP-UC3-Register New Customer User 140
Figure 68 Sequence Diagram – MP-UC3-Register New Customer User 141
Figure 69 Use Case Diagram – MP-UC-4-Onboard Cloud Service Provider 143
Figure 70 Sequence Diagram – MP-UC-4-Onboard Cloud Service Provider 143
Figure 71 - Marketplace web page. 144
Figure 72 Marketplace – Customer Portal – User Interface Mockup 145
Figure 73 BPaaS Marketplace Components 150
Figure 74 BPaaS Marketplace Component Interaction 150
Figure 75 - Deploy diagram of the Marketplace 151
Figure 76 Use Case Diagram – EvE-UC-1-KPI Analysis and Visualisation 155
Figure 77 Sequence Diagram – EvE-UC-1-KPI Analysis and Visualisation 156
Figure 78 Use Case Diagram – EvE-UC-2-Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules 159
Figure 79 Sequence Diagram – EvE-UC-2-Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules 160
Figure 80 Use Case Diagram - EvE-Uc-3-Process Mining Analysis & Graphical Representation 162
Figure 81 Sequence Diagram - EvE-Uc-3-Process Mining Analysis & Graphical Representation 162
Figure 82 BPaaS Evaluation Environment - Business Dashboard User Interface Mockup 164

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Figure 83 BPaaS Evaluation Environment – Components.....166

Copyright © 2016 BOC and other members of the CloudSocket Consortium

LIST OF TABLES

Table 1 – Prototype Components of Design Environment 17
Table 2 - Prototype Components of Allocation Environment 18
Table 3 - Prototype Components of Marketplace and Execution Environment 23
Table 4 - Components of Evaluation Environment 25
Table 5 BPaaS Design Environment - Use Case 1 –Business Process Design 48
Table 6 BPaaS Design Environment – Use Case 3 – Business Process Analysis 50
Table 7 BPaaS Design Environment – Use Case 3 – Semantic Lifting 52
Table 8 BPaaS Design Environment – Use Case 4 – Executable Workflow Design 55
Table 9 BPaaS Design Environment – Use Case 5 – KPI Definitions and Meta Model Completion 58
Table 10 BPaaS Allocation Environment – Use Case 1 – Creation of BPaaS Bundle 75
Table 11 BPaaS Allocation Environment – Use Case 2 – Atomic Service Allocation 78
Table 12 BPaaS Allocation Environment – Use Case 3 – Software Component Allocation 81
Table 13 BPaaS Allocation Environment – Use Case 4 – KPI Model Editing 84
Table 14 BPaaS Allocation Environment – Use Case 5 – SLA Model Editing 86
Table 15 BPaaS Allocation Environment – Use Case 6 – Pricing Model Editing 88
Table 16 BPaaS Allocation Environment – Use Case 7 – Business Process Metadata Editing 90
Table 17 BPaaS Allocation Environment – Use Case 8 – BPaaS Bundle Publishing in the Marketplace 92
Table 18 BPaaS Execution Environment – Use Case 1- Deployment of BPaaS 107
Table 19 BPaaS Execution Environment – Use Case 2 – Execution of the BPaaS 112
Table 20 BPaaS Execution Environment – Use Case 3 – Monitoring of Agreement Status 115
Table 21 BPaaS Execution Environment – Use Case 4 - Workflow Environment Management 119
Table 22 BPaaS Marketplace – Use Case 1 – Publish BPaaS Bundle to Product Catalogue 135
Table 23 BPaaS Marketplace – Use Case 2 - Purchase BPaaS Bundle 138
Table 24 BPaaS Marketplace – Use Case 3 - Register New Customer User 141
Table 25 BPaaS Marketplace – Use Case 4 – Onboard Cloud Service Provider 143
Table 26 Identify Management Use Cases 147
Table 27 BPaaS Evaluation Environment – Use Case 1 - KPI Analysis and Visualisation 156
Table 28 BPaaS Evaluation Environment – Use Case 2 – Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules 160
Table 29 BPaaS Evaluation Environment – Use Case 3 - Process Mining Analysis & Graphical Representation 162

Copyright © 2016 BOC and other members of the CloudSocket Consortium

1 INTRODUCTION AND PROBLEM STATEMENT

This document explains the CloudSocket architecture in order to.

- Create a common technical understanding and hence enable a coordinated final implementation of the BPaaS environments that compose the co-called CloudSocket.
- Specify the functional capabilities, competencies and data interchange format of the BPaaS environments in order to enable combining subsets of those environments for creating fixed or personalized instances of the CloudSocket platform (the so called CloudSocket Exploitation Product Packages – see D8.1 (D8.1 2016)) or exchanging these environments with alternative implementations.

Hence the functional capabilities, the required competencies and roles are explained per BPaaS environment to describe their intensions and to enable the flexible implementation and packaging as well as the adaptation of the CloudSocket platform.

A top down and bottom up approach has been applied by analysing the end users' needs. For a better clarification the term "CloudSocket Customer" is used to stress that use case requirements of potential future clients have been considered, which are distinguished from the "CloudSocket Broker", which is the one offering the BPaaS and tus covering the use case requirements.

To collect the BPaaS Customer requirements as well as the CloudSocket Broker requirements, this document relies on the results of the "Use Case Analysis and Evaluation Criteria Specification" (Del2.1 2015), where BPaaS customer scenarios have been worked out. For completeness reasons, some BPaaS Customer and CloudSocket Broker scenarios are recapped in this document, which intends to provide a complete context for the CloudSocket architecture but does not aim to repeat those findings. To strengthen the understanding of the BPaaS Customer context, the "Cloud Transformation Framework", both in form of a written report (Del2.3 2015), as well as a demonstrator providing a set of business processes (Del2.3a 2015) have been considered.

The basic idea behind the CloudSocket architecture is to use business process models in different forms to bridge the gap from domain specific business processes to IT-cloud specific executable and deployable workflow. This bridge is realized by a set of intermediate business process layers. The most challenging part is the separation of those layers in: (i) domain specific business processes, (ii) executable workflows, (iii) cloud deployable workflow bundles and (iv) workflows that are deployed and hence in production. Based on the CloudSocket terminology (Del2.2 2015), the used standards and the described functional capabilities this document contributes to clarify the layered approach that is applied for business and IT-cloud alignment.

In parallel to top-down analysis of BPaaS Customer and CloudSocket Broker use cases and requirements, a bottom-up approach has been followed by providing already existing software solutions for each of the BPaaS environments and a collaborative discussion to adapt this software to interact with the other parts. Several workshops in different formats – moderated workshops, four corner workshop, wikis, shared document repositories including UML diagrams and a series of Internet workshops – have been performed to agree on the first architecture for all five BPaaS environments composing the so-called CloudSocket.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

The second iteration of the architecture includes lessons learned during the development of the first prototype (D4.2_4.3_ 4.4 2016) as well as a better understanding of the use cases worked out in the set of business processes (D5.2 2016) and BPaaS bundlesbundels (D5.2 2016). Research findings in (D3.1 2016, D3.2 2016) have been incorporated into the corresponding BPaaS environments in the form of possible adaptations/evolutions.

The aim of the following chapters is:

- First, to provide an overview on the existing CloudSocket prototype.
- Second, to introduce the underlying architectural paradigms and principles that have been agreed and are set as axioms for this architecture.
- Third, to supply a short recap of the use case scenarios and the expectations of BPaaS Customers and CloudSocket Brokers.
- Fourth, to describe the different BPaaS environments by their functional capabilities, roles and data interaction in order to enable a high level blue print for alternative implementations of the CloudSocket. This ensures an adaptation of the five BPaaS environments to the needs and competences of a particular CloudSocket Broker and enables to reduce complexity by focusing on one BPaaS environment at a time and iteratively build a complete CloudSocket platform.
- Fifth, to describe each BPaaS environment in such detail that other partners can not only rely on the functional capability, the data format and the required competencies but also enable to develop alternative solutions or to introduce research findings within a BPaaS Environment.

The outlook indicates that the analysed architecture is expected to be further updated in the second implementation phase of the CloudSocket prototype while updating the functionality of BPaaS environments or incorporating research results into them.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

2 FIRST PROTOTYPE INTRODUCTION

The following software component description is copied from the first prototype documentation (D4.2_4.3_4.4 2016) and contains for each environment a common factsheet to ease navigation and accessibility. Detailed contextual information for each artefact is available in (D4.2_4.3_4.4 2016).

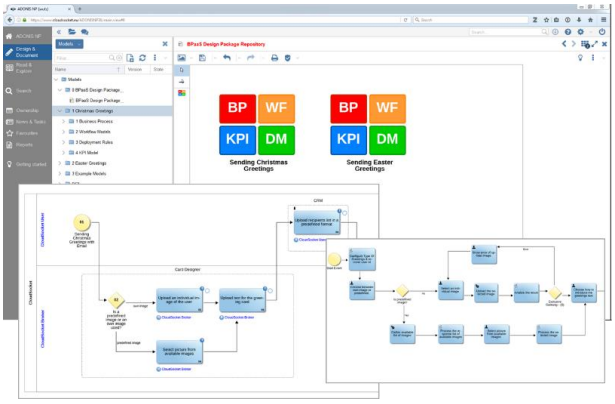
BPaaS Design Environment	
The BPaaS Design Environment has the overall goal to model aspects of a BPaaS by focusing on higher levels of abstraction. This leads to a generation of a BPaaS Design Package which describes an un-allocated BPaaS at the IT/cloud level by including various types of information, such as a domain specific business process model, an executable workflow-model, and a set of KPIs/requirements mapping to these two models. In addition, to enable the re-use of design knowledge as well as the automatic or semi-automatic alignment between business process and workflow models, the BPaaS Design Environment enables the storage, querying and retrieval of all model artifacts generated and their semantic annotation.	
Component	**Description**
BPaaS Design Tool	The BPaaS Design Tool has been created on the base of the CloudSocket metamodel and provides the possibility to model domain-specific business processes, execution workflows, decision models and key performance indicators. 
Access	SaaS Deployment: https://www.cloudsocket.eu/ADONISNP36/ (user credentials on demand) Experimentation Version Download: https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads
License	Closed source
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Design+Environment+Components
Lead Partner	BOC

Table 1 – Prototype Components of Design Environment

Copyright © 2016 BOC and other members of the CloudSocket Consortium

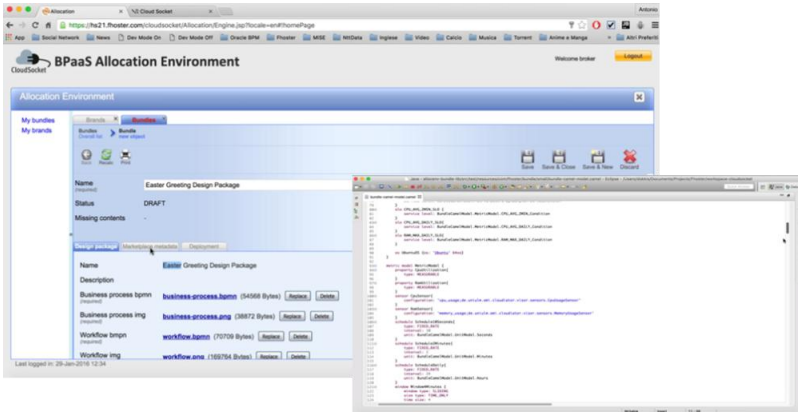
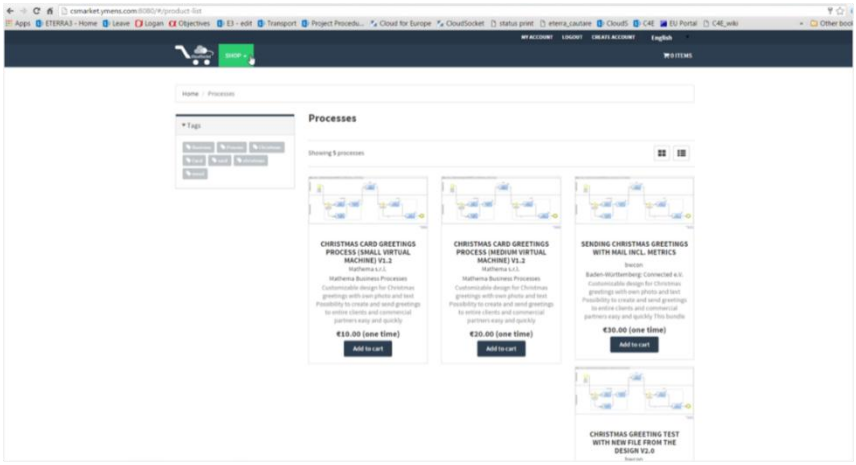
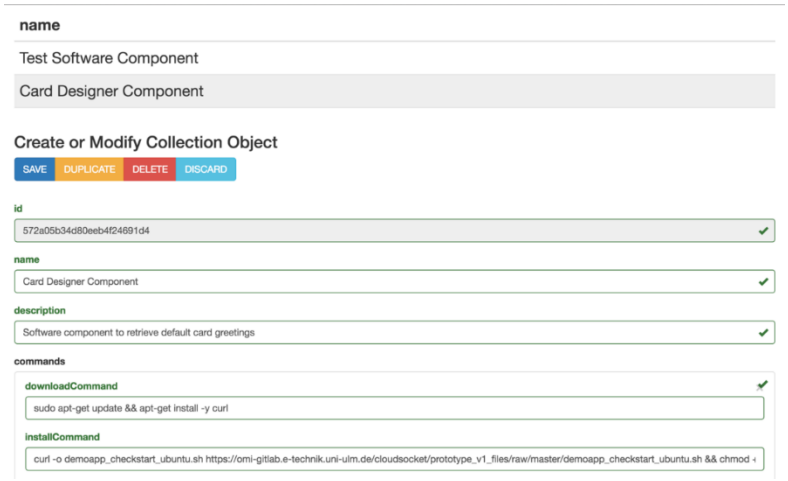
BPaaS Allocation Environment	
The goal of the BPaaS Allocation Environment is to configure allocation directives and rules for an executable workflow model to be deployed and executed in the cloud. An executable workflow model, as produced by the BPaaS Design Environment, does not contain information in terms of which concrete services can be exploited to realise the functionality of the workflow tasks. The respective selection of services per workflow task is supported by the BPaaS Allocation Environment. Similarly, driven by the same set of requirements, the same environment can also be used to address the selection of IaaS offerings to support the deployment and provisioning of (internal) BPaaS software components mapping to workflow tasks. Apart from these basic allocation decisions, the BPaaS Allocation Environment covers the specification of adaptation rules that drive the adaptation behaviour of a BPaaS as well as the specification of SLAs and marketing meta-data (e.g., pricing) for a certain BPaaS. In the end, the resulting product is a BPaaS bundle that can be published in the Marketplace, purchased and subsequently deployed in the cloud..	
Component	**Description**
Allocation Tool	It is responsible for selecting a BPaaS Design Package (previously created via the Design Environment) and creating a BPaaS Bundle ready to be published in the Marketplace and deployed in the Execution Environment. 
Access	SaaS Deployment: https://hs21.fhoster.com/cloudsocket/Allocation_prototype/Engine.jsp (user credentials on demand)
License	Proprietary
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Allocation+Environment+Components
Lead Partner	FHOSTER

Table 2 - Prototype Components of Allocation Environment

Copyright © 2016 BOC and other members of the CloudSocket Consortium

BPaaS Marketplace and Execution Environment	
The BPaaS Execution Environment deploys and executes a BPaaS bundle, once this has been purchased by a customer at the BPaaS Marketplace. The BPaaS deployment proceeds according to the deployment plan included in the bundle, along with additional configuration activities taken to enable the proper deployment of the workflow into a workflow engine and of the monitoring infrastructure. Once a BPaaS is successfully deployed, it can be run and managed by the BPaaS Customer. In addition, it is automatically monitored in a cross-layer manner and adapted, when needed, in order to keep up with the SLOs promised in the enclosed SLA of the BPaaS bundle.	
BPaaS Marketplace	
Component	**Description**
yCONNECT	It is an online frontstore through which customers discover, analyse and purchase BPaaS bundles by also initialising the respective BPaaS deployment in the cloud environment. Therefore, it is responsible for linking the Allocation to the Execution Environment, giving the client the opportunity to buy and configure the BPaaS bundles received from the Allocation and to send the configured bundles to the Execution for provisioning. 
Access	SaaS Deployment: http://csmarket.ymens.com:8080/ (user credentials on demand)
License	Proprietary
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Marketplace+Component
Lead Partner	YMENS

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Repository Manager	It is responsible for managing the information related to different entities such external services, software components, and cloud providers. It is a transversal component allowing the population, browsing and search of this information using standard web technologies. 
Access	SaaS Deployment: http://134.60.64.221/ (user credentials on demand) Download: as docker images • mongodb: https://hub.docker.com/_/mongo/ • restheart: https://hub.docker.com/r/softinstigate/restheart/ Restheart SchemaForm UI: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/restheart-schemaform-ui Registry Client Library: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/registry-client.
License	GNU AGPL v3.0 (GNU 2016)
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Repository+Manager+Component
Lead Partner	FHOSTER, ATOS

Copyright © 2016 BOC and other members of the CloudSocket Consortium

BPaaS Execution Environment	
Component	**Description**
Workflow Engine	It is responsible for managing the deployment, execution and management of the different workflow instances of a purchased BPaaS workflow at the execution phase. 
Access	http://134.60.64.132/activiti-webapp-explorer2/ (deployed as part of a bundle and user credentials on demand) Download Engine: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-engine. Download Workflow Parser: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-parser.
License	Apache License Version 2.0 (Apache 2016)
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Workflow+Engine+Component

Cloud Provider Engine	It is responsible for the complete deployment and lifecycle management of all the required components of the BPaaS bundle, including software components and VMs across multiple clouds, with transactional semantics (at least for the deployment part).
Access	Colosseum: http://134.60.64.155:9000 Entrypoint: http://134.60.64.155:9012/job Download: https://github.com/cloudiator/ Download EntryPoint Wrapper: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/execution-environment-simple-entrypoint
License	Apache License Version 2.0 (Apache 2016)
Manual	https://github.com/cloudiator https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Execution+Environment+Entrypoint
Lead Partner	UULM

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Monitoring Engine	It is responsible to monitor a BPaaS and correlate/aggregate monitoring data from different levels, from atomic services or cloud components up to the level of workflows.
Access	http://134.60.64.155:8080 Download: https://github.com/cloudiator/visor.git
License	Apache License Version 2.0 (Apache 2016)
Manual	https://github.com/cloudiator/visor
Lead Partner	UULM, FORTH

Adaptation Engine	It is responsible for reconfiguring the BPaaS possibly across different levels (via e.g., service substitution, workflow recomposition, horizontal and vertical scaling) to resolve the problematic situations identified by triggered adaptation rules.
Access	http://134.60.64.155:9000/api/composedMonitor http://134.60.64.155:9000/api/componentHorizontalOutScalingAction http://134.60.64.155:9000/api/componentHorizontalInScalingAction Download: https://github.com/cloudiator/axe-aggregator
License	Apache License Version 2.0 (Apache 2016)
Manual	https://github.com/cloudiator/visor
Lead Partner	UULM, FORTH

Copyright © 2016 BOC and other members of the CloudSocket Consortium

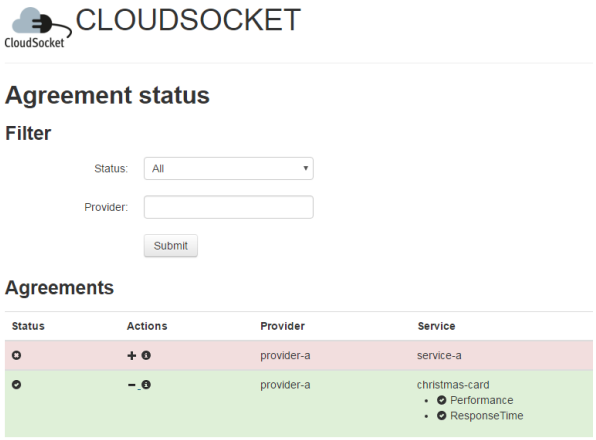
SLA Engine	The SLA Engine represents the component responsible for generating, storing and observing the formal documents describing electronic service level agreements (SLAs) between the parties involved in a BPaaS offering (CloudSocket Brokerbroker, cloud service providers supporting the BPaaS functionality), including of course the BPaaS Customercustomer 
Access	SaaS Deployment for SLA Dashboard: http://134.60.64.232:8000 (user credentials on demand) Core SLA Engine: http://134.60.64.232:8080 Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/sla-framework
License	Apache License Version 2.0 (Apache 2016)
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/SLA+Engine+Component
Lead Partner	ATOS

Table 3 - Prototype Components of Marketplace and Execution Environment

Copyright © 2016 BOC and other members of the CloudSocket Consortium

BPaaS Evaluation Environment	
The BPaaS Evaluation Environment has the overall goal to evaluate a BPaaS in order to provide optimization suggestions to its designer. This evaluation comes in various forms: (a) the assessment of KPIs, (b) the derivation of best deployments for the BPaaS, (c) the production of adaptation event patterns and rules and (d) the discovery of bottlenecks and problematic business model parts. Thus, the externally seen functionality of the BPaaS Evaluation Environment maps to initiating the performance of analysis tasks as well as the retrieval and graphical presentation of the various evaluation/analysis results produced according to suitable graphic metaphors by a business dashboard.	
Component	**Description**
Semantic Repository	A semantic repository enabling performing different types of analysis.
Access	http://134.60.64.222:8080/rest-test-swagger-0.0.1-SNAPSHOT/ Download: https://github.com/openlink/virtuoso-opensource.
License	GPL v2 (GPL 2016)
Manual	http://docs.openlinksw.com/virtuoso/.
Lead Partner	FORTH

Conceptual Analytics Engine	Provides an API through which KPI assessment can be performed on top of the semantic repository
Access	http://134.60.64.222:8080/rest-test-swagger-0.0.1-SNAPSHOT/ Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation_skb/repository/archive.zip?ref=master.
License	Mozilla Public Licence (MPL) 2.0 (Mozilla 2016)
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Conceptual+Analytics+Engine
Lead Partner	FORTH

Copyright © 2016 BOC and other members of the CloudSocket Consortium

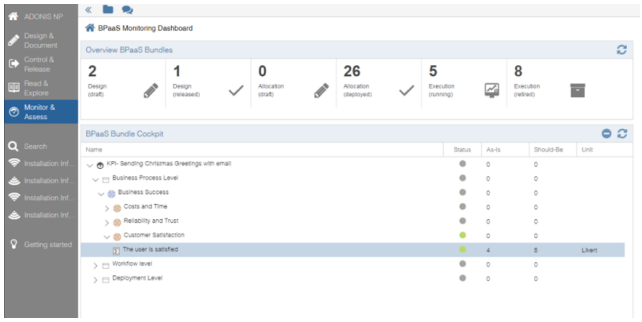
Hybrid Business Dashboard	Enables the visualisation of the analysis information via the use of suitable metaphors. Guides the user in properly performing the different types of analysis 
Access	SaaS Deployment: https://www.cloudsocket.eu/ADONISNP36/ (user credentials on demand)
License	Closed source
Manual	N/A
Lead Partner	BOC

Table 4 - Components of Evaluation Environment

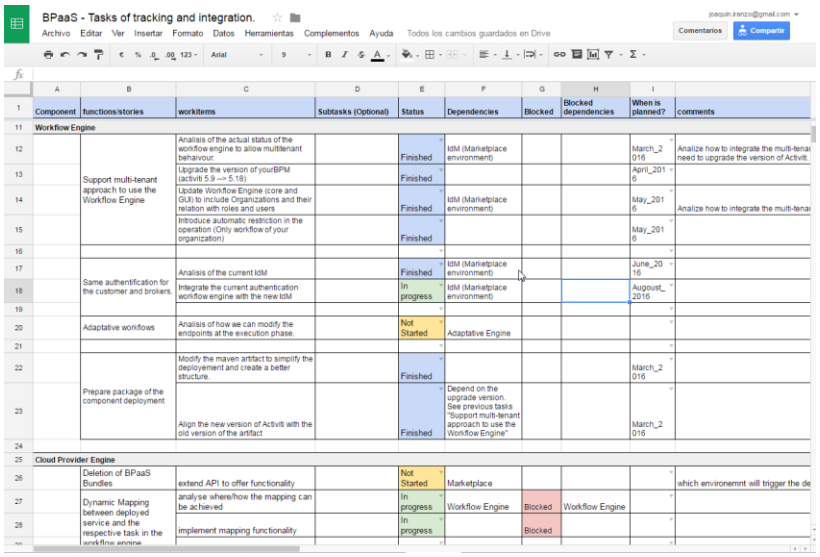
Copyright © 2016 BOC and other members of the CloudSocket Consortium

2.1 Implementation Approach

The different environments are aligned with the respective tasks in WP4: i) T4.2 is responsible for the Design and Evaluation Environment, ii) T4.4 for the Allocation Environment, and iii) T4.3 for the Marketplace and Execution Environment. Despite the fact that these environments are independent, the project has started to integrate and align them from the very beginning of the development phase. This allows to harmonize the different interfaces and interactions, which have been clearly defined in D4.1 (D4.1 2016), allowing to have a complete lifecycle including all the BPaaS (management) phases.

This approach has been incremental by following the four following steps: i) detailing the interfaces and identifying the basic functionalities for the different environments and components; ii) performing internal testing for internal components or between different (internal) components of the same environment; iii) inter-integration between environment pairs; iv) final integration of all components by also covering all phases and environments.

This integration has been followed up through short periods; identifying the risk and problems as soon as possible in order to take corrective actions, if needed. A shared excel file has been introduced to review the details of the work items and the respective functionality realisation status and their planning, besides the dependencies and blocked actions. Every environment has been covered by different sheets where the components report the status of their work items. MoreoverMorover, this file also defines the different sprints and their integration at the different levels. WP4 has realized periodic meetings every two weeks to analyse the integration, the work items per component and to identify the new parallel discussions, such as the definition of a registry-based approach, the CAMEL integration/extension, and the metric definition.


BPaaS - Tasks of tracking and integration.

Component	functions/stories	workitems	Subtasks (Optional)	Status	Dependencies	Blocked	Blocked dependencies	When is planned?	comments
Workflow Engine									
		Analisis of the actual status of the workflow engine to allow multitenant behaivour.		Finished	IdM (Marketplace environment)			March_2016	Analize how to integrate the multi-tenant need to upgrade the version of Activiti.
	Support multi-tenant approach to use the Workflow Engine	Upgrade the version of yourBPM (activiti 5.9 --> 5.18)		Finished				April_2016	
		Update Workflow Engine (core and GUI) to include Organizations and their relation with roles and users		Finished	IdM (Marketplace environment)			May_2016	Analize how to integrate the multi-tenant
		Introduce automatic restriction in the operation (Only workflow of your organization)		Finished				May_2016	
		Analisis of the current IdM		Finished	IdM (Marketplace environment)			June_2016	
	Same authentification for the customer and brokers	Integrate the current authentication workflow engine with the new IdM		In progress	IdM (Marketplace environment)			August_2016	
	Adaptative workflows	Analisis of how we can modify the endpoints at the execution phase.		Not Started	Adaptative Engine				
	Prepare package of the component deployment	Modify the maven artifact to simplify the deployement and create a better structure.		Finished				March_2016	
		Align the new version of Activiti with the old version of the artifact		Finished	Depend on the upgrade version. See previous tasks "Support multi-tenant approach to use the Workflow Engine"			March_2016	
Cloud Provider Engine									
	Deletion of BPaaS Bundles	extend API to offer functionality		Not Started	Marketplace				which environemnt will trigger the de
	Dynamic Mapping between deployed service and the respective task in the	analyse where/how the mapping can be achieved		In progress	Workflow Engine	Blocked	Workflow Engine		
		implement mapping functionality		In progress		Blocked			

Figure 1 - Work item definition table

This simple approach allows to integrate the different environments even at their early stagesstates and it is completely aligned with the Task T4.5 CloudSocket Integration and Consolidation, which will introduce more efficient tools for managing the continuous integration for all the levels; spanning not only the development, for example a ticketing system to cover the development, but also integration, bugs, and management of different environments (test, integration and production).

Copyright © 2016 BOC and other members of the CloudSocket Consortium

3 ARCHITECTURAL PARADIGM

This architectural paradigm is based on the fact that each of the five environments is provided by software vendors, who already provide at least parts of the required functional capabilities in the cloud. Hence CloudSocket improves first the way those individual software applications are interacting as a CloudSocket platform and second by introducing innovative extensions.

The paradigms to keep those software solutions independent ensures not only a joint solution in form of the CloudSocket, but enables the flexible exchange and adaptation of one or several environments to the needs of a particular CloudSocket Broker, thus ensuring individual improved solutions as a side effect.

In the following those paradigms are briefly introduced.

3.1 Loose coupling of Building Blocks

The architecture of CloudSocket comprises five environments which focus on the traditional lifecycle of business processes involving the phases of design, allocation, execution and evaluation as well as its usage. These environments are the following:

- BPaaS Design Environment: focuses on the design of BPaaSs
- BPaaS Allocation Environment: focuses on allocation of BPaaSs in terms of services realizing BPaaS tasks and VMs hosting the BPaaS components
- BPaaS Execution Environment: focuses on the execution, monitoring and adaptation of BPaaS
- BPaaS Evaluation Environment: focuses on the evaluation of a BPaaS with the special goal to provide helpful insights about how it can be optimized
- BPaaS Marketplace: focuses on the brokerage of BPaaS offerings to the customer

Each environment is autonomous and loosely coupled with respect to the other environments. This enables the integration of different implementations of the environments in order to provide an integrated CloudSocket platform for Brokers. It is also more suitable in cases that brokers desire to use some but not all of the environments as they might only need to focus on particular phases in the BPaaS lifecycle or they already have tools in place which deal with the rest of the phases.

The main functional capabilities of each environment are described to an extent sufficient for the other environments to interface with it. This means that this document describes first a high level overview of the BPaaS environments and second a more detailed view on the technology that acts for both: (a) a high level description of the BPaaS environments to be possibly exchanged with other implementations as well as (b) a more detailed level of technical description that acts as a common understanding between technical partners and specifications of interactions.

Data exchange format and APIs are described to set accordingly the way environments can interact and cooperate with each other in a loosely coupled manner. Such a manner dictates that the least and sufficient information pertaining to such an interaction is the specification of the API exposed by each environment and the data exchange formats mapping to the input and output.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Environments are intended to be realized by different solutions, hence a vendor lock is avoided as any solution can be exchanged by another one that follows the same data exchange format and provides the same APIs. This is actually the philosophy behind service-orientation as a particular system can be built or even be adapted through the appropriate selection of services that expose the same functionality.

3.2 Model Driven and Layered Alignment Approach

At this stage, the term “business process” is introduced as a sequence of manual, semi-automatic and automatic actions with the aim to achieve an organizational goal.

Hence the business process aligns all tasks with the business goals, independently from whether they are performed by machines, humans or a group consisting of several machines and / or several humans. There is a set of different application fields for business processes, such as but not limited to quality management, risk management, re-engineering, continued improvement, documentation, training but also model driven architecture and requirement analysis.

The first set of application fields see the business process model as an "Information Value Provider", hence graphical models are not seen as some necessary step to move on to concrete software code, but as an independent document that is needed for day to day work within the organization.

The second set of application fields is concerned with, the model driven architecture, software requirement analysis, configuration of software components or software design. In those application scenarios, the business process model is seen as a “Specification and Requirement Collection”, which is further detailed and transformed to either specific deployable workflows or executable software code.

Business and IT alignment in CloudSocket is concerned with both; on the one side, to smartly transform and detail business processes to become deployable workflows, and, on the other side, keep the information value aspect for the business users.

In the following, only the relevant layers are highlighted.

- Layer I – Domain Specific Business Processes: Domain specific business processes that describe the business activities, which are – in the way they are presented – not executable, by a workflow engine within or even outside the cloud.
- Layer II – Executable Workflows: Executable workflows are represented by workflows that orchestrate the interaction between software services. It is expected that one domain specific business process typically maps to many executable workflows depending on the level of automation, the selected services and failure / recovery / variant handling.
- Layer III - Cloud deployable Workflow Bundles: Deployable and executable workflows that are packaged for cloud deployment consisting of all relevant configurations, so that they can be deployed in the cloud on demand. It is expected that one executable workflow maps typically to many cloud deployable workflow bundles depending on the different cloud providers selected, the corresponding SLAs involved, the deployment management strategies and the actual multi cloud deployment.
- Layer IV – Deployed Workflow Bundle in Production: This reflects a bundle that has already been deployed in the cloud such that it can be exploited by the BPaaS Customer that has purchased it. The corresponding

Copyright © 2016 BOC and other members of the CloudSocket Consortium

workflow of the bundle is ready to be instantiated and executed. One cloud deployable workflow bundle can map to many deployed workflow bundles in production, purchased and deployed on behalf of one or more BPaaS Customers.

In order to align the aforementioned four layers, each layer has to be described in appropriate form. Hence, there are two challenges to be met. First, to find appropriate representation formats for each individual layer. Second to find appropriate mechanisms to link the different layers.

The representation format is described in the used standard data interchange of the different BPaaS environments; the appropriate mechanisms to link different layers – as well as link different BPaaS environments will be researched as a form of hybrid semantic and conceptual meta model integration. Details on such model weaving is represented in D3.1 (D3.1 2016) and D3.2 (D3.2 2016).

3.3 Data Exchange

The use of standards enables the setting of the format of the data to be exchanged as well as maps to a better integration of systems and components and the exploitation of a vast variety of alternative software that supports these standards. CloudSocket uses standards to describe business processes, their annotations and respective rules while extends them, when and where appropriate. For business processes representation, the widely used BPMN standard (BPMN 2014) is exploited, which is able to support both domain specific business processes and executable workflow models. Concerning the annotation of business processes, as we aim at providing automatic support for various tasks operating on such processes, we opt for using ontologies. A de facto standard for the description of ontologies is OWL (OWL 2012), so annotations will be described by this standard, or if RDF is sufficient by using RDF (RDF 2014).

We also have the case of business rules and adaptation rules. Business rules may be specified through the DMN standard (DMN 2015). In addition, the SRL (Scalability Rule Language) language (SRL 2015) and possible respective extensions are foreseen to achieve the appropriate expressiveness level to support the specification of not only scalability but any kind of adaptation rule.

Deployment plans, part of a BPaaS bundle, are already described by the CAMEL DSL (CAMEL 2015) (CloudML 2015) and in particular its part dedicated to the deployment of applications.

3.4 Interfaces

In order to keep the flexibility enabling different realizations of different BPaaS environments, the interfaces are described on several levels. The basic interface is the file exchange realising aforementioned data structures. Of course, a tighter interaction via service-orientation can be achieved through two different ways: (a) REST services and (b) SOAP services. However, it is not the focus of this deliverable to explicitly indicate which way to realize a particular interface. However, as a rule of thumb, in case that stateful operations need to be in place, then SOAP services should be used. In case stateless operations are needed and there is a need for better performance, then REST services should be used.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4 USER SCENARIOS AND HIGH LEVEL ARCHITECTURE

This section presents how use case and stakeholder requirements have been considered together with the initial high level architecture in order to derive usage scenarios and use cases, which are further detailed in the succeeding sections.

4.1 Use Case and Stakeholder Requirements

The requirement analysis is described in more detail in the CloudSocket Use Case and Evaluation Criteria Analysis (Del 2.1 2015). For completeness reasons, some major parts that are useful to raise the understanding are introduced in the next sub-sections as a summary of the aforementioned use case analysis. These parts focus on indicating the requirements involved for the different use cases which can of course have an effect on the design of the CloudSocket architecture.

4.1.1 Business Incubator Scenarios: A Recap

The Business Incubator focuses on supporting the "Coaching and Finance" efforts of start-ups facilitating designing, analysing and simulating individual business plans and processes. These aspects also demand a high degree of adaptability of Cloud Services for Start-ups, e.g., Customer Relationship Management, Order Management, and Human Resources Management. To this end, the following use cases have been developed for this CloudSocket Broker, according to requirements coming from the real-world which map to ideas for creating innovative start-up companies and how they could be supported through the CloudSocket platform.

4.1.1.1 Ecological Agriculture

A 28 year old, biologist, has an idea to take biological waste from a restaurant and stimulate a biological decomposition process. Usually such a process takes several years but the idea of the startup is to use worm to speed up this process.

Initial situation: The startup presented the ideas to the business incubators. After this, the consultants have discussed with her about how to transform this business idea into a solid business model.

CloudSocket technology intervention: The startup may require a range of different customer relationship and worm production management solutions.

Potential BPaaS solution: Business processes are a common instrument to explain, how the 28 year old biologist prefers to perform the customer relationship and worm production management. Based on those requirements, a mapping to either (a) already existing SaaS solutions that cover the whole business process, or (b) a combination of different SaaS and / or local installed applications might be recommended, while (c) no Cloud support cannot also be excluded from the suggestions supplied.

4.1.1.2 Green Energy

This startup maps to a small-scale virtual power plant which connects to a grid infrastructure with power generation from wind, photovoltaic, and biogas. The company serves its customers with environmentally friendly energy for household and provides smart home functions through its remote access capability for turning appliances on or off.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Initial situation: The company is intending to expand its services to include mobile energy sources for recharging electric cars and offer them for rental as range-extension for drivers, e.g., for a long weekend trip. To this end, the startup contacted the business incubator consultants.

CloudSocket technology intervention: The startup may require a range of different process-based solutions for customer relationship, partner management and internal management.

Potential BPaaS solution: Based on business process models, it is possible to define the expansion strategy and the required IT (Cloud) support. Depending on those requirements, the different alternatives of Cloud support can be worked out.

4.1.2 Cluster Process Broker Scenario: A Recap

The Business Process Broker use case identifies typical business episodes of potential SMEs in different application domains such as eHealth, Manufacturing, Photonics, Government, Security, e-Commerce, Retails, which, however, share a common set of business processes.

4.1.2.1 Internet Research and Procurement Process

An SME, employing 10 people, sells software and integrated appliances/electronic components that make devices "Internet ready" in a few seconds.

Initial situation: The SME continuously verifies prices of the electronic and mechanic components in the market and buys only products that match specific requirements in terms of customer needs and pricing. Monitoring the prices and the quality is a costly activity, which requires an ongoing analysis and trade-off between quality and price.

CloudSocket technology intervention: The SME needs a solution that reduces the costs for procurement activities by improving the effectiveness of the procurement process. Generic self-management infrastructure or specially designed research processes, involving crawlers and result databases, have the potential to run in the cloud and to raise the productivity of this SME.

Potential BPaaS solution: This is a representative sample where business processes can be used to optimize the business of an SME, by providing consulting via business processes. Internet Recherché, the actual and current expansion idea/solution, can be realized via many different business processes, ranging from pure manual to full automatic, from structured to unstructured or from ad-hoc to regularly triggered processes. Different business processes require different cloud support, ranging from no support at all, to SaaS support untill personalized configuration of workflows.

4.1.2.2 Kiosk Distribution Process

A company with 180 employees aims at distributing newspapers and magazines to kiosk and, in general, points of sales in an Italian town. Every day, about 250 different Italian and foreign newspapers are delivered to 600 points of sales.

Initial situation: Current customers are small kiosks with very limited IT infrastructure. Often, the order is realised via Facebook comments. In order to improve the maturity of the ordering and interaction process with those kiosks, new but still light-weighted Web-applications will be provided.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

CloudSocket technology intervention: A new order process can be handled in the cloud, without IT installation on both – the supplier and consumer – sides. The scaling of the application with the expected peeks in ordering at the end of the working day or triggered by a special event indicates the use of cloud solutions.

Potential BPaaS solution: The process of ordering can reflect a better understanding of the distribution process due to the transparent business process model by also raising awareness on IT difficulties – e.g. peek handling – and hence improve the distribution process.

4.1.3 The CloudSocket Customer

In addition to the two aforementioned use cases – that describe the targeted end users market – we describe the entry point of any interested user.

We propose three steps for a typical SME or startup as an end user:

- Check Cloud Readiness,
- Transform Business Processes to be executable in the Cloud,
- Enter the marketplace to access BPaaS.

The project provides a checklist for SMEs and start-ups in order to check, if they are capable in entering the cloud with their business processes. This framework is available in the form of a demo accessible at (Del 2.3 2015) while it is analysed in (Del 2.3a 2015).

The transformation of business processes to be executable in the Cloud is divided in two transformations. The first transformation is a horizontal one that transforms from one business process to another one. Although both business processes are not executable, the latter one has clear anchor points, where cloud offerings can be added. Hence the horizontal transformation extracts those parts of the process where a cloud offering can actually be applied.

The next transformation is a vertical one that maps the resulting business process to an executable workflow in the cloud - this actually injects the cloud offerings and enables the execution in the cloud. This next step is performed by entering the market place and selecting the most appropriate workflow that runs in the cloud. This selection can be supported by smart alignment mechanisms.

Readers are encouraged to visit those tools and provide feedback to enable a collaborative improvement in order to reduce the barriers for SMEs that have no cloud competence.

It is expected that in addition to the two aforementioned CloudSocket Brokers, new brokers may be interested in CloudSocket; hence in the following section, the entry point for new CloudSocket Brokers is provided.

4.1.4 The CloudSocket Broker

The CloudSocket is a brokerage platform with additional capabilities, where BPaaS are offered in a similar way as SaaS are supplied today. Hence, it is a common marketplace that is well known in cloud computing. The same mechanisms as in the case of SaaS are used for BPaaS with the main distinction that in the second case the service offered is in the form of an “executable business process”. The differences between a SaaS and BPaaS marketplaces concern mainly the selection criteria. A SaaS offering is typically selected based on technical properties, whereas the BPaaS involves a two-step selection, considering first the domain/business properties and second the technical

Copyright © 2016 BOC and other members of the CloudSocket Consortium

properties. This means that for one business processes there may be many different workflow realizations, and for one workflow realization there may be many different cloud offerings.

For organizations aiming to become a CloudSocket Broker, we propose the following steps:

- Identify the potential market for BPaaS,
- Plan Business Processes, by using a business process management tool that attracts potential clients,
- Build Business Processes, by implementing executable workflows for the first selection step and deployable bundles for the second selection step.
- Run Business Processes, by offering workflows on an operative cloud market place infrastructure.
- Check Business Processes, by abstracting cloud monitoring logs up to domain-specific business indicators.

These initial recommended phases for supporting CloudSocket Brokers are initially realised via the use of certain tools (CSBT 2015).

“Plan Business Processes” denotes the use of business process management tools to acquire, design, analyse and simulate and finally release domain-specific business processes. Here, we understand business processes as a know-how platform of an organisation; hence those processes have the potential for domain-specific consultancy and improvement. Traditional business process management tools, such as ADONIS® (ADONIS 2015), are used.

“Build Business Processes” denotes that each of the aforementioned business processes are made executable by a set of deployable and executable workflows. We agreed to use the term workflow for processes that are orchestrated and executed on an IT platform to strengthen the difference with respect to human orchestrated or executed business processes. Traditional workflow design tools like yourBPM (yourBPM 2015a, yourBPM 2015b) may be used.

“Run Business Process” indicates the provision and operation of a process as a service within a cloud market place that is executed and run across services offered in the cloud. Although this is technically the most challenging part, the focus of the CloudSocket project is on the alignment, hence the mapping between domain specific business processes and cloud deployable and executable workflows.

“Checking Business Processes” indicates the abstraction, using conceptual models and semantic, to introduce a semantic meaning into the technical data and process logs from the execution environment in the cloud. The meta model platform ADOxx (ADOxx 2015) will be used to develop conceptual and semantic models that can be analysed and mapped to business processes.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4.2 Initial High Level CloudSocket Architecture

The initial high level CloudSocket architecture introduces the vision of BPaaS Environments that together compose the CloudSocket platform. Each of the BPaaS Environments corresponds to a particular phase of the Business Process Management System methodology (Kar 96), including the support for business and IT alignment. This mapping from BPMS phases to BPaaS Environments derives an initial list of functional capabilities.

Those functional capabilities and the respective data exchanges involved are introduced in Figure 2. The BPaaS offerings are provided to the customer via the Marketplace, whereas the BPaaS Execution Environment enables their operation in the cloud. The conceptual challenge of bridging domain specific business processes to executable workflows that are in production in the cloud, is performed by the others BPaaS Environments.

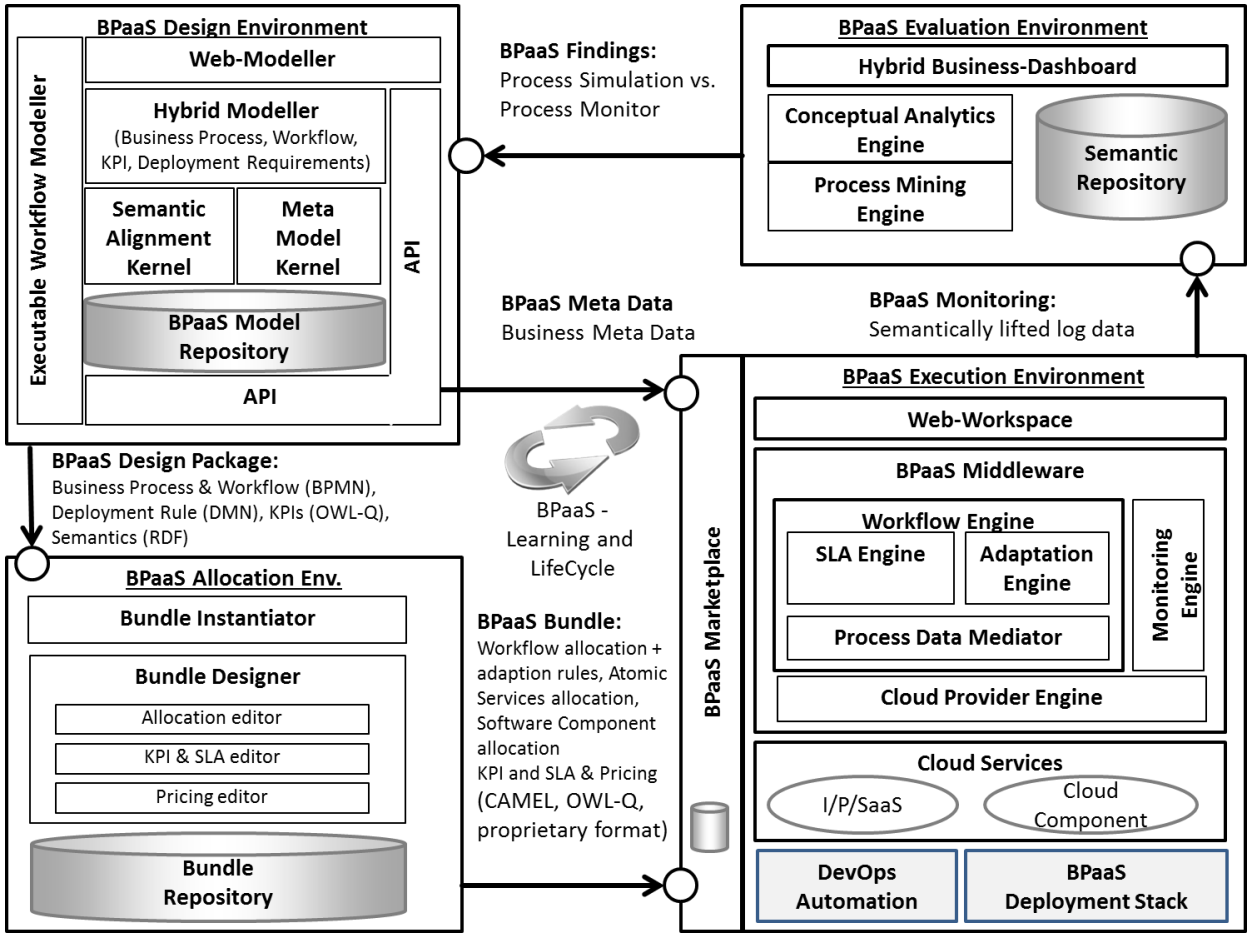


Figure 2 Initial High Level Architecture

In the following, an overview on the functional capabilities and the data exchange between the different BPaaS Environments is introduced.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4.2.1 Key Functional Capability of BPaaS Environments

The functionality of each BPaaS Environment focuses on a particular phase of the BPaaS lifecycle.

The BPaaS Design Environment has the overall goal to design a BPaaS. To this end, such a design should include the capabilities to edit domain specific business process models, edit executable workflow models, store business process models and workflows, map in a semi-automatic or automatic manner a domain specific business process model to an executable workflow model as well as semantically annotate a business process and a workflow model. Thus, as it can be seen, the above functionality maps to going from a domain specific need of the business process to a formal specification of an executable workflow and supporting all types of modelling that can be involved in such a transition. In addition, it also maps to the capability to semantically annotate the involved models in order to provide automatic support to various tasks that might have to be performed, such as the transformation of domain specific business process model to executable workflow models. What would someone exploit from such an environment is the retrieval of the models that have been produced in the context of the design of a BPaaS.

The goal of the BPaaS Allocation Environment is to configure allocation directives and rules for an executable workflow model to be deployed and executed in the cloud. An executable workflow model, as produced by the BPaaS Design Environment, does not contain information about which concrete services can be exploited in order to realise the functionality of the business process tasks. To this end and driven by the business and technical requirements, the BPaaS Allocation Environment supports the CloudSocket Broker in making an informed selection of which services from the candidate ones to select for each business process task. The same set of requirements should also drive the decision about which IaaS offerings to select in order to deploy software components of the BPaaS. Through both types of selection, the ending result would be not only a fully executable workflow model but also a deployment plan which will enable the deployment of the BPaaS, thus enabling its execution by the BPaaS Execution Environment. Another connecting piece related to a BPaaS and its deployment refers to the specification of adaptation rules that can drive the adaptation of a BPaaS when such a need arises. Such rules are important if a more or less constant service level needs to be exhibited by the BPaaS to its customers. Such a service level is specified in the form of an SLA template which will be incarnated into a real SLA when the BPaaS bundle is purchased by the BPaaS Customer. All above main products of the BPaaS Allocation Environment are encapsulated in a so called BPaaS bundle which can then be published in the Marketplace in order to be available to the CloudSocket Customers. Thus, similarly to the case of the previous environment, what can actually be externally exploited by any other environment is those models that are being produced mapping to the BPaaS bundle.

The BPaaS Execution Environment aims at deploying and executing a BPaaS bundle, once this has been purchased by a customer at the BPaaS Marketplace. 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 into a workflow engine in order to enable its execution by the customer that has purchased it. As such, the execution of the workflow encapsulated in the BPaaS bundle is supported. Another goal of this environment is to support the monitoring and evaluation of the BPaaS according to the 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. Concerning again the external functionality, as can be seen by CloudSocket Customer and the other environments, the BPaaS Execution Environment exposes a functionality which enables deploying a BPaaS, creating, executing and managing instances of the workflow encapsulated by the BPaaS and producing as well as supporting the

Copyright © 2016 BOC and other members of the CloudSocket Consortium

retrieval of BPaaS monitoring and assessment results (for evaluation purposes in the BPaaS Evaluation Environment).

The BPaaS Evaluation Environment has the main goal to evaluate a BPaaS in order to provide optimisation suggestions to its designer. This evaluation comes in various forms: (a) the assessment and drill-down of KPIs, (b) the derivation of best deployments for the BPaaS, (c) the production of adaptation event patterns and rules and (d) the discovery of bottlenecks and problematic business model parts. Thus, the externally seen functionality of the BPaaS Evaluation Environment maps to performing BPaaS evaluation and retrieving the various evaluation results produced.

4.2.2 Data Interaction between BPaaS Environments

The next sections introduce the data exchanges that are indicated in the aforementioned High Level Architecture.

4.2.2.1 BPaaS Design Package

The BPaaS Design part comprises the workflow, support information for the allocation and additional information on the original domain specific business process.

This is a package consisting of:

- The domain specific business process and the executable workflow model in BPMN
- Business process and workflow extensions such as semantic annotations in RDF, key performance indicators in OWL-Q and additional deployment relevant description in DMN.

4.2.2.2 BPaaS Meta Data

The BPaaS Meta Data package mainly provides optional information to any other environment.

This package provides:

- Additional domain specific business process information, such as an image and description text.
- Domain specific business process and workflow linkage
- Semantic annotation of business processes and workflows with the BPaaS ontology, which includes (i) APQC and (ii) functional description ontology.

4.2.2.3 BPaaS Bundle

The BPaaS bundle comprises mainly the deployment information, which makes the workflow complete for production in the BPaaS Execution Environment.

This is a package consisting of:

- Domain Specific business process - as received from the BPaaS Design Package in BPMN.
- Executable workflow models – as received from the BPaaS Design Package in BPMN - with corresponding deployment information.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

- Mapping of abstract atomic services involved in the workflow with actual concrete atomic services available in the cloud and registered in the Service Registry.
- Mapping of software components involved in the workflow with IaaS offerings registered in the Cloud Provider Registry.
- Extending KPIs in OWL-Q and creating SLA in WS-Agreement and pricing.
- Adaptation rules in DMN and extended SRL of CAMEL to drive the BPaaS adaptation behaviour during runtime.

4.2.2.4 BPaaS Monitoring

This package provides different kinds of – semantically enriched - monitoring information, from the BPaaS Execution environment to the BPaaS Evaluation environment.

It consists of:

- process logs for a BPaaS,
- monitoring information for a BPaaS and
- contextual/deployment information for a BPaaS.

4.2.2.5 BPaaS Finding

This is a package providing optimisation suggestions from the BPaaS Evaluation to the BPaaS Design Environment. It might comprise one or more from the following three main information items:

- best deployment suggestions,
- adaptation rules suggestions and
- business process model suggestions.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4.2.3 BPaaS Security Layer

In the definition of the architecture, we have introduced and defined different environments that are independent, decoupled and modular; hence the considered security solution will promote the maximum possible scenarios to maintain the same philosophy. The platform will provide and foster a cross-environment security, allowing to each Environment implementation owner to either adopt it or use its own solution.

The marketplace is responsible to publish and purchase the BPaaS bundle and interact with the customers, the organizations, the cloud providers (IaaS, PaaS and SaaS) and the system. Then, it will manage the authentication, the complete lifecycle between the consumers, services and the providers, maintaining them all together and coherently. Due to these natural capabilities, such cross-environment functionality should be part of the marketplace, allowing the rest of the environments to decide the level of integration that they want to implement.

Nevertheless, the BPaaS Execution Environment has to be integrated with the actual security solution, since it must manage the relations with customers and cloud (service) providers in an easy way. Besides the system has to guarantee the simplicity for the authentication mainly with the interaction of the customers and their quality of experience. The cross-environment solution is based on standards, such as Cross-domain Identity Management (SCIM 2015), SAML 2.0 (SAML 2015), OpenId Connect 1.0 (OIDC 2015), and OAuth 2.0 (OAuth 2.0 2015).

Based on this solution, there can be two different security modes: i) the cross-environment one where each environment adopts the common cross-environment solution; ii) a federated security mode which takes into account the trust between the different owning organizations of the environments such that users of one environment can be recognized as users also for the other environments.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4.3 CloudSocket Usage Scenario

Through the inputs of the use case analysis, we can derive a CloudSocket use case scenario that provides a high-level coverage for the functionalities / capabilities exposed to the main stakeholders/actors.

The identified functional capabilities are covered by the different BPaaS environments:

- The CloudSocket Broker has an idea about a particular business process he / she wants to offer to its customers. Based on the experience on the market potential and on feedback from potential clients, the CloudSocket Broker decides to offer a BPaaS on the CloudSocket platform he/she exploits.
- The CloudSocket Broker designs the domain specific business process using the BPaaS Design Environment. He/she may involve Business Process Modellers and Ontology or Domain experts to raise the quality of the business process. Business related artefacts are added to the domain specific business process in order to communicate the business needs to the technical experts.
- The CloudSocket Broker may then involve a Workflow Modeller, who creates one or several workflow models for each domain specific business process. Hence, by involving business process modellers, domain experts and workflow modellers, the CloudSocket Broker bridges the business to IT gap from the business process to the workflow and produces suitable design artefacts: domain business process models, executable workflow models, annotations and business rules.
- The CloudSocket Broker takes allocation decisions with the assistance of the BPaaS Allocation Environment in order to create BPaaS bundles. Technical assistance may be acquired from experts, such as technical consultants or operators that are providing the necessary production system.
- The allocation decisions rely on Service Providers offering services that could realize or provide support to a part or the whole BPaaS workflow of the broker. Such services are published in the Service Registry in order to be exploited.
- The CloudSocket Broker also selects IaaS services for hosting internal software components from the Cloud Provider Registry to realize the functionality of the respective executable workflow tasks. He/she also defines adaptation rules that can drive the runtime adaptation of the BPaaS. The allocation decisions at the IaaS level are described in the form of a deployment plan with failure semantics. The CloudSocket Broker should also define the respective SLA that would explicate the service level offered as well as the corresponding obligations and penalties of the signatory parties. The final allocation product, the BPaaS bundle, encompassing allocation decisions, plans, and SLAs, is published in the marketplace.
- A CloudSocket Customer browses the marketplace and selects a particular BPaaS bundle through the support of the marketplace assistance system. The CloudSocket Customer purchases this bundle and creates accounts for the cloud (SaaS or IaaS) services included in it, if such accounts do not already exist. Once the above actions finish, the bundle is deployable and thus sent to the Execution Environment for adaptive provisioning and operation. The latter operations are performed automatically, so the CloudSocket Broker needs just to observe them and intervene whenever required.
- The CloudSocket Customer can use the interface of the Execution Environment in order to create instances of the BPaaS as well as to monitor the SLOs that are part of the SLA conducted with the CloudSocket Broker. The BPaaS instances can be executed or managed (e.g., suspend workflow instance execution, resume, skip one task, etc.).

Copyright © 2016 BOC and other members of the CloudSocket Consortium

- The CloudSocket Broker uses the monitoring interface of the BPaaS Execution Environment to check which SLAs are met and which are not and in the latter case discover the main root cause(s) of the problem. He/she does not always need to intervene as the BPaaS Execution Environment is able to automatically adapt the BPaaS when SLOs are not met, provided that the respective adaptation rules have been already specified in the BPaaS bundle.
- The CloudSocket Broker also exploits the BPaaS Evaluation Environment to check whether KPIs are met or not, why they are not met as well as retrieve optimization suggestions that can lead to redesigning and re-allocating the BPaaS.
- The CloudSocket Broker can analyse current costs through the marketplace and can decide, by also considering the findings from the BPaaS Evaluation Environment, whether to: (a) change the pricing model of the BPaaS, (b) alter the allocation decisions on the executable workflow/bundle.
- The CloudSocker Broker can also inspect the monitoring and evaluation information provided by the Execution and Evaluation Environments in order to improve the BPaaS offered by, e.g., modifying the underlying business process, workflow, or bundle (SLA, pricing model, deployment plan, service allocation, adaptation rules).
- The BPaaS Customer can also inspect current charge and SLA status information so as to inspect whether he/she can: (a) continue using the BPaaS, (b) stop/cancel the BPaaS SLA when the respective conditions being currently applied allowing him/her to do so, (c) change offerings (if he/ she desires to, e.g., choose a better SLA to accommodate for the increased load from its customers).

As it can be seen from the above analysis, we have actions which are performed by the **CloudSocket Broker** to manage a BPaaS (design, allocate, configure, monitor and improve it) as well as actions performed by the **CloudSocket Customer** related to the purchasing and actual usage of a BPaaS.

In order to demonstrate the CloudSocket vision Figure 3 depicts sample BPaaS offerings. A BPaaS marketplace enables the **BPaaS Customer** to select from a business processes according his domain-specific needs. He/she then views a filtered list of business processes from which he/she can select the desired one. The BPaaS Customer can then choose the preferred workflow realisation by viewing: (i) the business process (ii) the workflow and (iii) cloud specific information, such as the technical details of the deployment.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

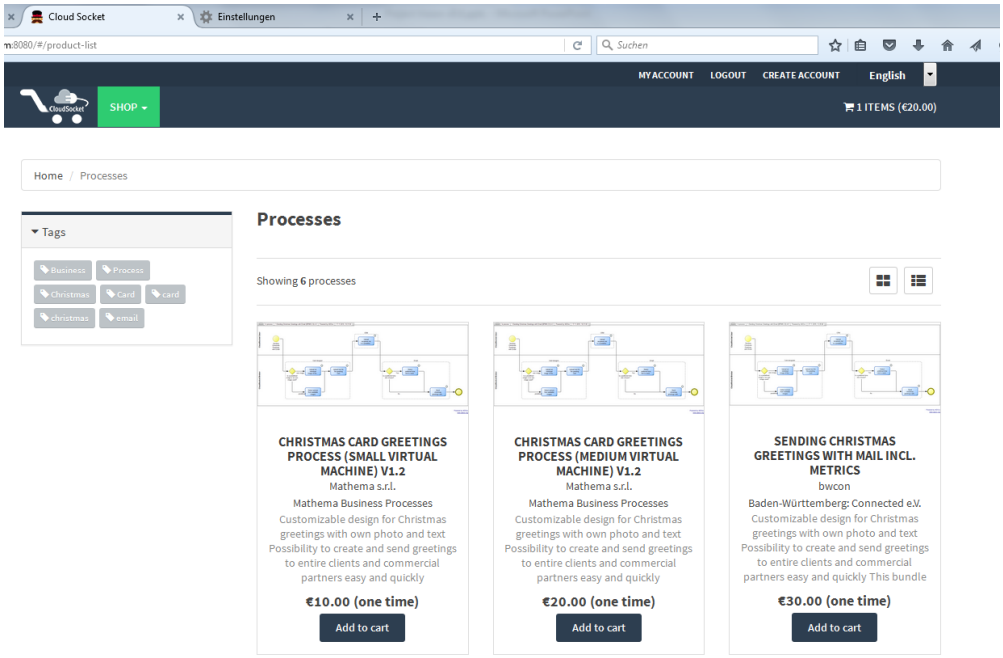


Figure 3 BPaaS Marketplace

The above scenario explains the vision, where the new type of cloud offerings in form of business processes is introduced. According this idea, additional application sceanrios can be enabled like cases where the BPaaS is used as a consulting service, where the CloudSocket Broker acts also as a business process consultant for the CloudSocket Customer. Hence, the resulting business processes may either be modelled by the CloudSocket Broker, the CloudSocket Customer or in a collaborative way between them. In the latter case, the roles involved for some environments are updated (e.g., customer roles in the Design Environment or even possibly at the Allocation Environment depending on the technical capabilities of the customer). The BPaaS reference models (D5.2 2016) describe in more detail the different ways business processes can be completed.

The steps of the **CloudSocket Broker** can be regarded as part of the CloudSocket lifecycle, whereas the steps of the **BPaaS Customer** can be regarded as CloudSocket user interactions.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

4.4 CloudSocket Ecosystem

Here the different actors and perspectives described in the first exploitation and business plan (D8.1 2016) as well as in the prototype documentation (D4.2_4.3_4.4 2016) are repeated for completeness reasons to explain the intended ecosystem. The project has considered a classification of the stakeholders, which are all the individuals, groups, units or communities that (a) could be interested in project development and exploitation or (b) just follow the project results and especially those activities of CloudSocket that could have a direct or indirect impact on them. These stakeholders have been split into 2 main levels: i) the different groups of project partners which are involved in the CloudSocket concept development as well as are directly linked to project success and further exploitation, ii) all further groups of stakeholders which are not directly in charge of the project execution but are somehow interested in its results. Therefore, the prototype is aligned with this first level, where the following stakeholders are considered (see Figure 4):

- CloudSocket Customer can be Small and Medium Enterprises (SMEs), founders and start-ups, both IT and not IT-based, which in case of CloudSocket project represent potential broker customers. These end-end users are identified in the sequel of this deliverable as BPaaS Customers as they represent potential purchasers of the BPaaS offerings provided by the CloudSocket Brokers.
- CloudSocket Brokers do not operate only as a third-party business that is an intermediary between the purchaser of a cloud computing service (SMEs and start-ups) and the provider of that service (Marketplace with its multi-clouds offer) but can also act as a consultant to support SMEs in transferring their business processes into the cloud, after assessing their cloud readiness. Moreover, such brokers can realize the whole lifecycle of cloud-based business processes thus saving for SMEs/startups the work of attempting to perform the business-to-IT alignment themselves as well as the investment of resources and time in order to support this lifecycle.
- Technology providers are project partners, which provide their software components/products, such as base CloudSocket functionality realization or add-on functionalities, or other (commercial) organizations which offer replacements of software components that have been developed by the members of the CloudSocket consortium. We foresee that such technology providers might also offer the whole CloudSocket prototype to brokers in order to enable the respective management of the BPaaS to be generated and offered to BPaaS Customers. In some cases, such providers may also offer particular environments, like the Marketplace, to be exploited by brokers, leading to a more loosely coupled instantiation of a running CloudSocket platform comprising different environments that are maintained by different operators.
- Researchers are university representatives (researchers, academics) and research groups (universities, institutes) focusing on research and development, elaborating their teaching courses, building network and research communities, consolidating and conceptualizing of abstracts or general ideas. This stakeholder kind performs research and development tasks which can result in add-on or component replacement prototypes that could be embraced in existing CloudSocket product variants.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

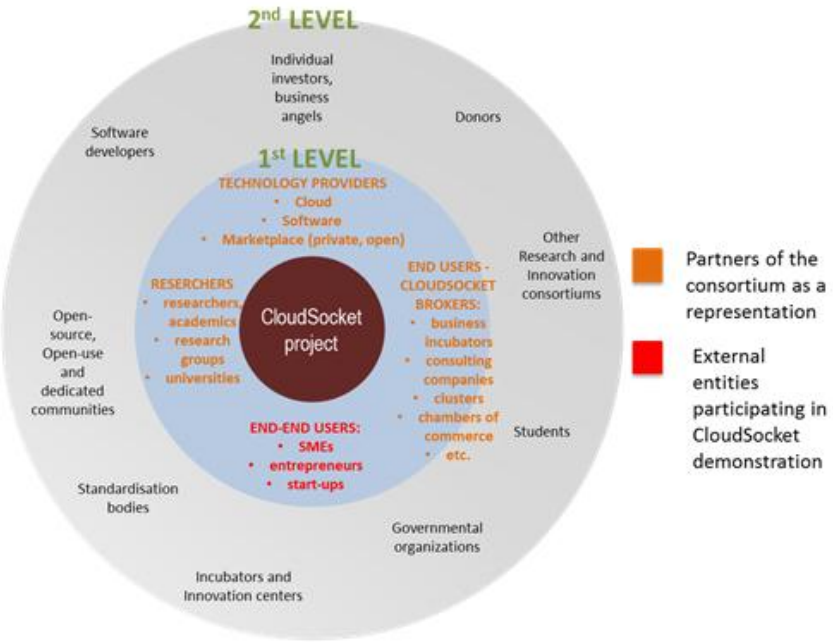


Figure 4 - CloudSocket ecosystem source:(D8.1 2016), p14

Therefore, the two main actors and their perspectives have been considered to show the demonstration in the following sections:

- CloudSocket Customer Perspective: Involves entities such as SMEs, founders or start-ups that want to reduce their costs and create added value for their business processes by moving some parts of them (or fully) in the cloud. They can search, review and purchase the different BPaaS bundles, which have been published previously by the CloudSocket Brokers. Afterwards, the purchased BPaaS Bundle is deployed automatically in the cloud and it is ready to be used by the customer.
- CloudSocket Broker Perspective: The CloudSocket Brokers want to create a BPaaS bundle in order to expose them to possible customers that might be interested in them. They need to interact with the Design and Allocation Environments to create the bundle. Afterwards, they can publish the bundles in the marketplace such that these bundles are exposed and available to their customers. Finally, the brokers can analyze the results of the Evaluation Environment to optimise existing BPaaS or create new ones by identifying the respective needs to be covered.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5 BPAAS DESIGN ENVIRONMENT

5.1 Introduction

The BPaaS Design Environment provides appropriate conceptual modelling tools for: (a) designing domain specific business processes, (b) executable workflows, (c) additional descriptions and rules for deployment as well as (d) Key Performance Indicators. In order to provide those different modelling tools within one environment, a meta modelling platform is used that enables the plug-in of different modelling aspects. For BPaaS, the aforementioned different modelling approaches correspond to the first two business process layers – the (I) domain specific business process as well as the (II) executable workflows – thus actually providing support to the alignment from the business to the technical layer.

Hence the meta modelling platform enables to keep all models in one repository as well as the interaction between the different layers via so-called model weaving and semantic lifting.

Independent modelling tools have the benefit of proprietary strengths in one particular aspect; hence the use of an independent workflow modelling tool is foreseen, which can be added on to the meta modelling platform. This enables the use of a proprietary workflow modeller corresponding to the workflow engine in the cloud production environment as an external tool; on the same time, it enables the interaction between the different modelling layers in the meta model platform.

A meta modelling platform enables the plug-in of different modelling aspects – they are called “modelling methods”.

The plug-in system is built on two main approaches:

- REST Services and ADOScript based components: Modelling methods are integrated as external components and may provide their features through the network. In such a case, a component written in ADOScript scripting language is used in order to communicate with the service, process the input and output and integrate it in the BPaaS Design Environment. In case the external component is deployed locally or is not a network component, the ADOScript is used to call and interact directly with its API.
- Javascript Model Features Block (MFB) components: The WebModeler part of the Design Environment gives the possibility to add features and extensions using a special javascript structured package named MFB. Each MFB is a plug-in that integrates directly into the modelling environment.

The BPaaS Design Environment consists of:

- A meta modelling platform that provides a BPaaS model repository for all its models, the corresponding management and security infrastructure, and a development environment that enables the implementation of modelling components.
- BPaaS modelling components are distinguished by their modelling languages – which are implementations of standards like BPMN, DMN, or RDF – as well as by modelling features, such as user interaction and model processing. The domain specific business process modeller and the executable workflow modeller are actually both such a meta modelling component.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

- The corresponding Web-GUI realizes the user interaction features, to manipulate a model.
- Interfaces enable the access to the BPaaS Design Environment, in particular to the BPaaS model repository, which comprises domain specific business process models, executable workflow models and additional business requirements. The respective interaction related to the access can rely on standard features, such as BPMN export / import or on implemented proprietary exchange formats.

In addition to this meta model based business process framework, the BPaaS Design Environment has the possibility to perform different kind of analysis and annotate the models with an ontology. The so-called semantic lifting enables the semantic annotation of BPaaS models with global ontology concepts, while simulation, formal correctness verification and cloud readiness check constitute the supported analysis phases. Hence, we introduce the following additional elements:

- The Semantic Annotation Kernel.
- Business Process Simulation Service.
- Business Process Verification Service.
- Cloud Readiness Check Service.

The functional capabilities, apart from designing business processes and workflows, include visualizing, querying, and simulating. Another functional capability involves the rule-based transformation from any kind of model format to another format. In addition, the environment offers capabilities to compose meta-data and to define KPIs in a top-down manner.

Each level enables incorporating the definition of Key Performance Indicators (KPI); hence KPIs can be defined on: (i) domain specific business process level, (ii) workflow level and (iii) and deployment level. The combination of the semantic lifting with KPI models results in the semantic annotation of business process models and workflow models with ontology concepts from business, IT and Quality of Service (QoS) / Quality of Business (QoB) ontologies.

Following roles are involved at the CloudSocket Broker side, when using the BPaaS Design Environment. The **Business Process Designer** is responsible for modelling domain specific business processes, while the **Workflow Designer** is responsible for modelling the executable technical workflows. The result of a business process design is a set of models that are made available to the next step in the life-cycle of a BPaaS which maps to its allocation supported by the BPaaS Allocation Environment.

The models produced are the following:

- a domain specific business process model in BPMN format and additional meta data such as figures, cloud specific requirements or KPIs along with information pertaining to the description of the CloudSocket Customer, its non-functional requirements and its main business objectives,
- a semantic lifting of the business process model typical form of a BPMN model along with RDF-based semantic annotations to business & IT ontology concepts,
- the executable workflow model in BPMN again along with RDF-based semantic annotations and
- the definition of KPIs based on OWL-Q.

When deploying the BPaaS Design Environment, the following components need to be deployed and start operating: (a) a web-based modeller offered in the form of a SaaS or web application which enables domain specific business processes design, (b) a BPaaS model repository which is responsible for the management & retrieval of (b1) domain

Copyright © 2016 BOC and other members of the CloudSocket Consortium

specific business process models, (b2) executable workflow models, and (b3) the KPI models; assisting in the task of the designer to compose models out of fragments constituting previous design knowledge.

5.2 Functional Capabilities

- Business Process Design including visualization, query, and transformation of domain specific business processes in different formats.
- Business Process Analysis including process simulation, formal correctness verification and cloud readiness check.
- Executable Workflow Design including visualization, validation and transformation of executable workflows in different formats.
- Top-Down KPI definitions for all layers, such as business process, executable workflow, deployment and operative process instance.
- Meta Data Composition to describe business processes and executable workflows.
- Semantic Lifting of domain specific business processes and executable workflows.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.2.1 Business Process Design

Use Case id	DE-UC-1-Business Process Design
Title and Description	A CloudSocket Broker wants to offer a new BPaaS to its clients; hence the broker models a domain business process specifying the required functional and non-functional process behaviour. This requires to login into the business process repository and might include browsing and searching within existing business process models. A new business process can be designed either by creating a new model and start modelling a new business process from scratch, or by copying another model and reconfiguring it via changing that model. Business process and KPI design has two major elements: (a) first the software tool that provides the appropriate modelling features and enables a user interface to the business process repository, and (b) second the modelling method that provides the modelling language. The modelling language consists of the business process modelling notation (BPMN), the decision model notation (DMN), the Resource Description Language (RDF) and a semantic description of the KPIs (OWL-Q).
Actors	CloudSocket Broker – Business Process Designer
Use Case Objective	Business Processes are designed from the business perspective
Pre-Condition	The user (Business Process Designer) is registered in the BPaaS Design Environment and has competence in business process modelling. For business process browsing and searching, previous business processes need to be stored or imported into the repository.
Process Dialog	User interaction for browsing and searching: • Selecting directories of models or creating / editing / deleting a directory • Create a new model, or select an existing model and save as a new model. • Model management by renaming, deleting or editing models. User interaction is required for the process modelling, consisting of: • Drag and Drop of modelling objects • Parameterisation of each modelling object • Graphical notation optimisation indicating selected parameters • Navigational support, such as hyperlinks between models • Modelling support, such as zoom, snap grid, connector bending, selection, undo or model navigation.
Variations	Model management may be used to massively reorganise the repository or import business processes in form of BPMN files.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

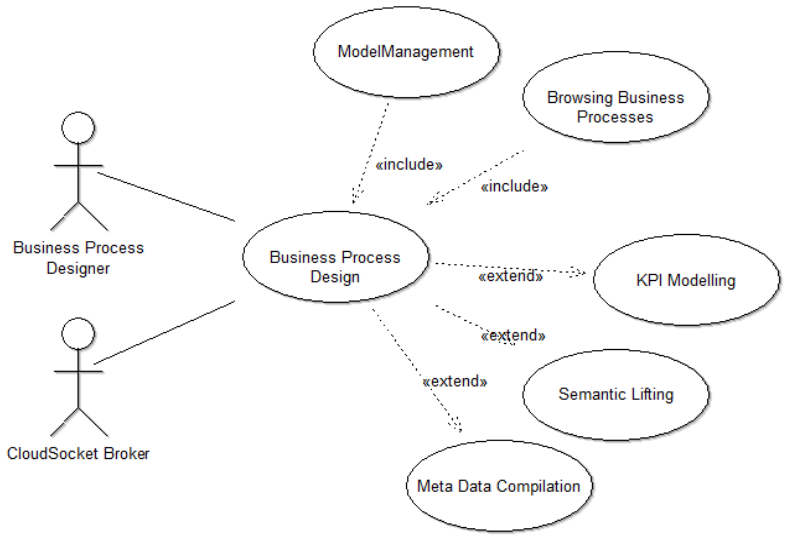
Post Condition	Business Processes are modelled and can be further used.
Diagram	

Figure 5 Use Case Diagram – DE-UC-1-Business Process Design

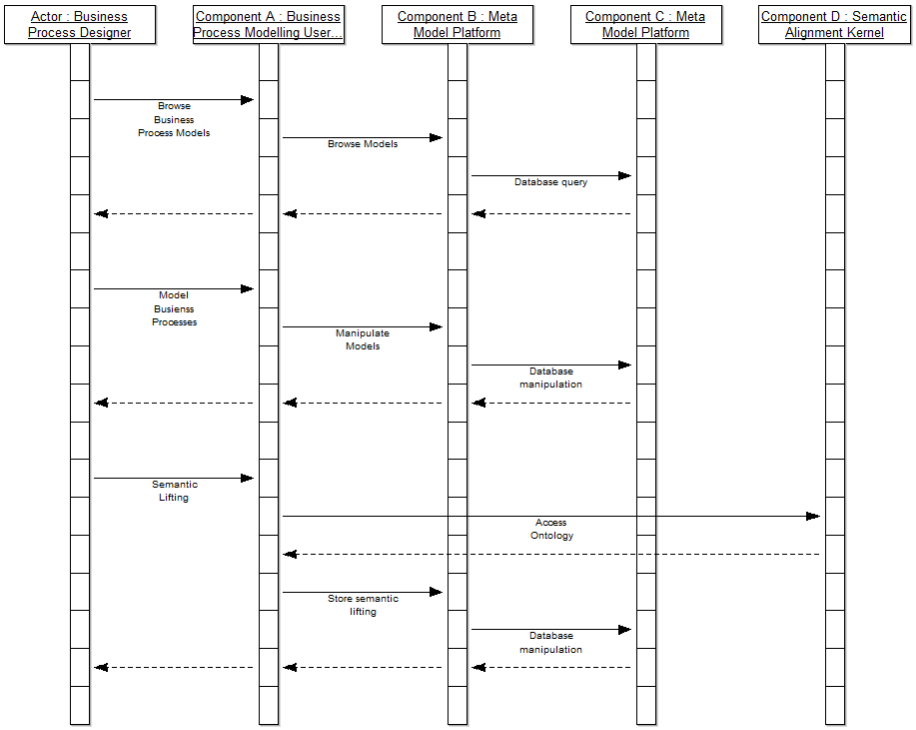


Figure 6 Sequence Diagram – DE-UC-1-Business Process Design

Table 5 BPaaS Design Environment - Use Case 1 –Business Process Design

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.2.2 Business Process Analysis

Use Case id	DE-UC-2-Business Process Analysis
Title and Description	Business Processes need to be analysed in order to check their correctness, their unexpected behaviours as well as evaluate costs and decisions like the readiness to be deployed in the cloud. The analysis phase in particular comprises the following functionalities. Simulation analysis in order to evaluate costs and times of the service in order to optimize and evaluate its feasibility. Verification analysis in order to check the presence of structural problems that may rise unexpected behaviours in the process execution like deadlocks or livelocks. Cloud Readiness analysis in order to evaluate through a series of questions, if a specific path of the Business Process Workflow is ready to be deployed in the cloud or not.
Actors	CloudSocket Broker - Business Process Designer. It is expected that the Business Process Designer knows sufficiently about statistic concepts and technical details in order to simulate and answer the cloud ready questionnaire. If not it is his/her responsibility to involve a technical expert and a mathematician.
Use Case Objective	The business process correctness is guaranteed
Pre-Condition	Business processes models are available.

Process Dialog	In the design environment, the user has to open the model explorer and right click on the model he wants to analyse and then choose between the following alternatives:
	
	• Select Remote Simulation in the menu in order to visualize the simulation interface and launch the simulation. Results will be available in minutes and be visualised in tabular and chars forms in order to be easily interpreted
	  ○ Details on paths, traces and activities involved in the simulation can be highlighted in the model using the appropriate “show” buttons.
	• Select Remote Verification in the menu in order to visualize the formal verification interface:
	  ○ Select the deadlock check in order to guarantee the absence of deadlocked paths
	  ○ Select the boundness check in order to guarantee the absence of livelocked paths.
	• Select Cloud Ready Check in the menu in order to start the cloud readiness checking process: this process will guide the user in answering questions on activities of the Business Process and will return a score indicating its feasibility to be deployed as a whole or partially in the cloud.

Variations	Formal verification can be performed in order to evaluate some user-defined rules like guaranteeing the sequence of two events. These methods are available in the Verification interface.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

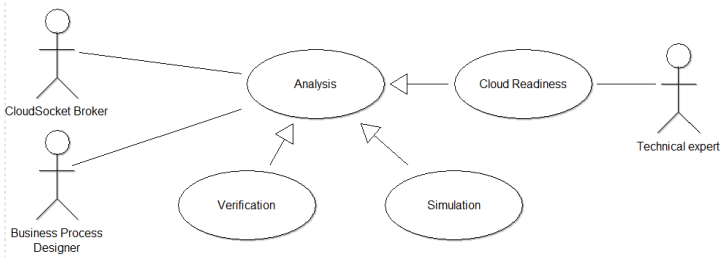
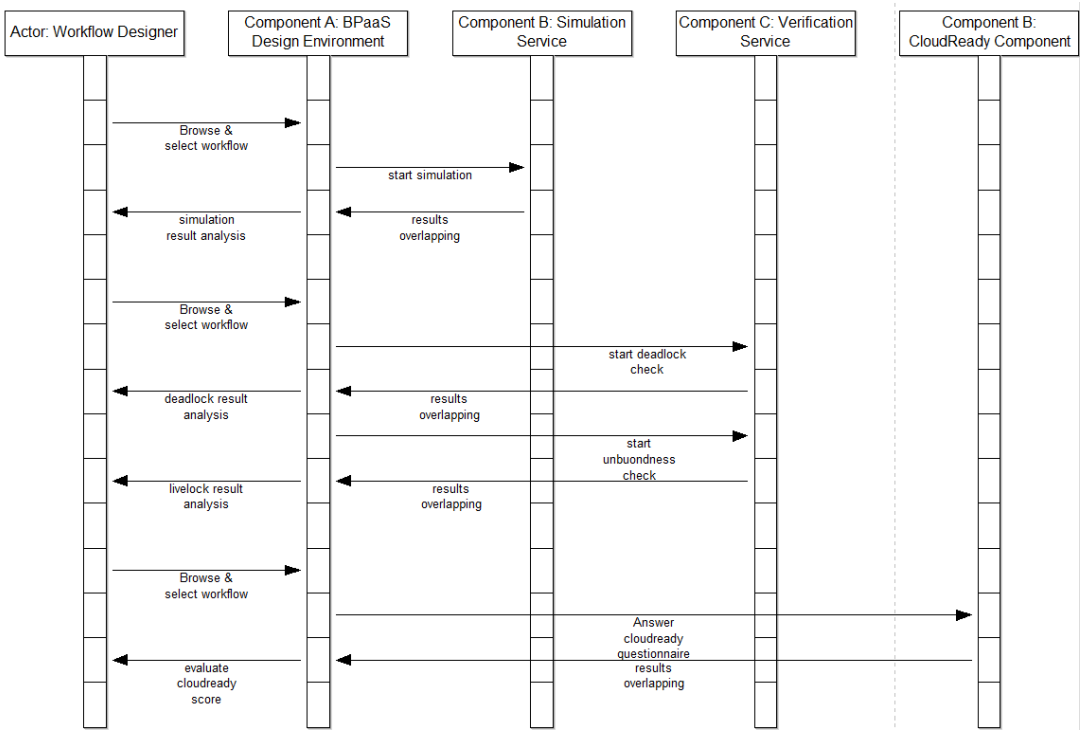
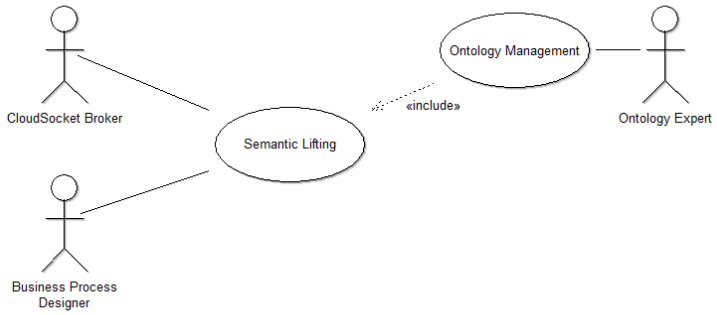
Post Condition	Business Processes are analysed and their behaviour is the one expected.
Diagram	 **Figure 7 Use Case Diagram – DE-UC-2-Business Process Analysis**  **Figure 8 Sequence Diagram – DE-UC-3-Business Process Analysis**

Table 6 BPaaS Design Environment – Use Case 3 – Business Process Analysis

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.2.3 Semantic Lifting

Use Case id	DE-UC-3-Semantic Lifting
Title and Description	Business Processes describe the business view and hence need to be annotated to cloud specific selection criteria, such as QoS, technical parameters or CloudSocket specific business indicators. Semantic Lifting enables to align business process descriptions so as to be linked to semantic concepts of an ontology. Hence, each element of a business process can be enriched with a globally defined – cloud specific - ontology.
Actors	CloudSocket Broker - Business Process Designer. It is expected that the Business Process Designer knows sufficiently about ontology handling; if not it is his/her responsibility to involve an ontology expert. However, for simple usage of the ontology, no ontology expert is expected.
Use Case Objective	Business process is described with parameters that enable the business oriented selection in the marketplace as well as the business oriented identification of Key Performance indicators.
Pre-Condition	Business processes models are available. The (cloud) ontology with concepts used for lifting must be provided.
Process Dialog	Within the user interface of the design environment there is a dialog enabling the selection of ontological concepts. This selection dialog can vary depending on the complexity: • Simple selection of a concept from the ontology • Pre-selection of ontology parts from the model and fine grading by selection from the ontology
Variations	In case there are missing concepts that need to be incorporated into the ontology, the ontology must be updated. This is performed by an ontology expert.
Post Condition	Business Processes are semantically enriched and hence semantically searchable.
Diagram	

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Figure 9 Use Case Diagram – DE-UC-3-Semantic Lifting

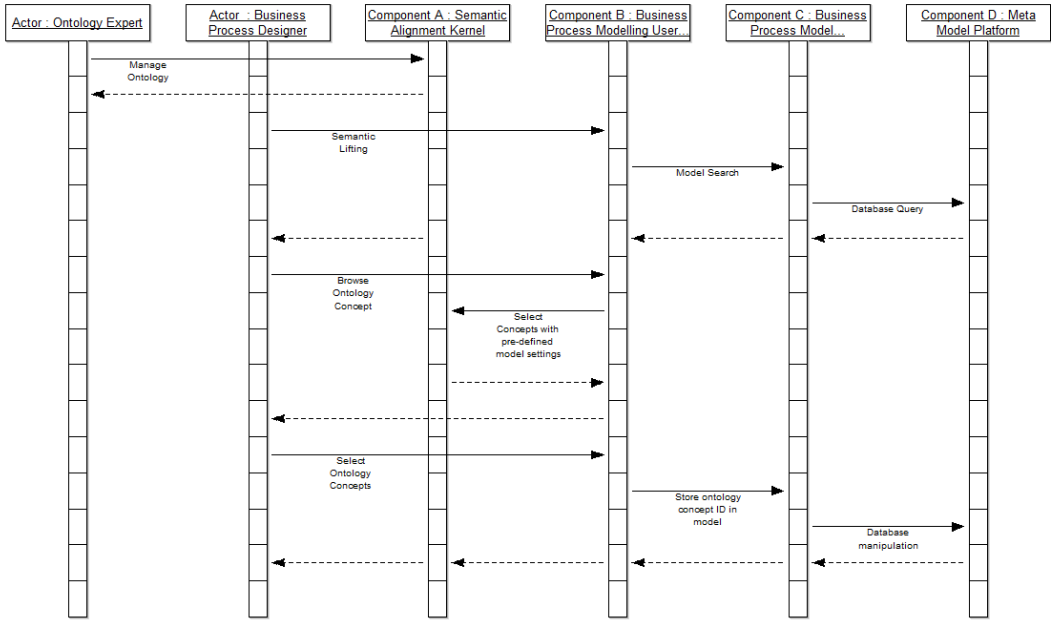


Figure 10 Sequence Diagram – DE-UC-3-Semantic Lifting

Table 7 BPaaS Design Environment – Use Case 3 – Semantic Lifting

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.2.4 Executable Workflow Design

Use Case id	DE-UC-4-Executable Workflow Design
Title and Description	The business process describes the domain view and hence does not include technical details to enable its execution. Hence, the executable workflow design is a technology oriented description of the previously described domain specific business process. This design transforms domain specific activities into technical specifications by: • Resolving all manual activities in actions that are understood by a workflow engine • Resolving all semi-automatic activities in actions that are understood by a workflow engine • Detailing all automatic activities with parameters that are necessary for a workflow engine • Detailing the data exchange from business level documents to technical data formats To this end, a business process can be realised in one or many executable workflows.
Actors	CloudSocket Broker – Workflow Designer It is expected that the Workflow Designer knows sufficiently about ontology handling; if not it is the responsibility of the Workflow Designer to involve an ontology expert. However, for simple usage of the ontology, no ontology expert is expected.
Use Case Objective	Definition of executable workflows that are linked to domain specific business processes.
Pre-Condition	The domain specific business process must be available and semantically enriched with cloud-specific details, thus providing sufficient input information for creating an executable workflow.
Process Dialog	The Workflow Designer needs first to select the business process by: • Browsing or searching for business processes • Copying the business process in a workflow template and establishing a link between the original business process and the workflow • Exporting the workflow template from the BPaaS model repository into a workflow designer tool that is compatible with the workflow engine in the cloud production environment. After the selected workflow template model is exported in the corresponding workflow design environment, the workflow is modelled by: • Using drag and drop feature of modelling objects • Creating, editing and deleting modelling objects • Modelling support, such as zoom, snap grid, connector bending, selection, undo or model navigation.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

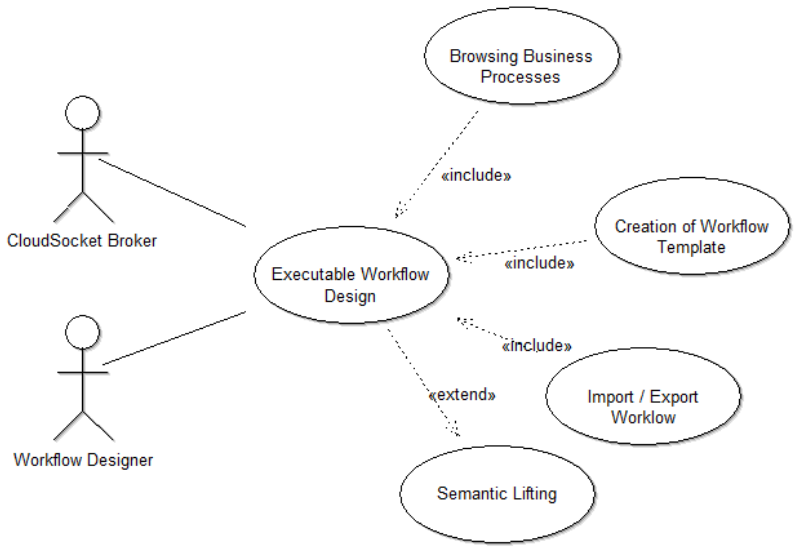
	• Parameterising each executable action with technical parameters. After designing the executable workflow, it has to be imported back into the BPaaS model repository, so that the corresponding executable package can be searched and found in combination with the business process: • A user interface is offered for importing “executable” workflows into the BPaaS model repository. After the workflow is imported into the BPaaS model repository, it is semantically lifted by: • Simple selection of a concept from the ontology • Pre-selection of ontology parts from the model and fine grading by selection from the ontology.
Variations	In case executable workflows are created without the original business process, they can still be imported as executable workflows, but need to be linked to the domain specific business process afterwards. So, the business process is then selected or created and linked to the executable workflow.
Post Condition	The executable workflow is linked to the original business process and is semantically lifted to both the business process and its semantic annotation as well to additional own semantic annotation.
Diagram	

Figure 11 Use Case Diagram – DE-UC-4-Executable Workflow Design

Copyright © 2016 BOC and other members of the CloudSocket Consortium

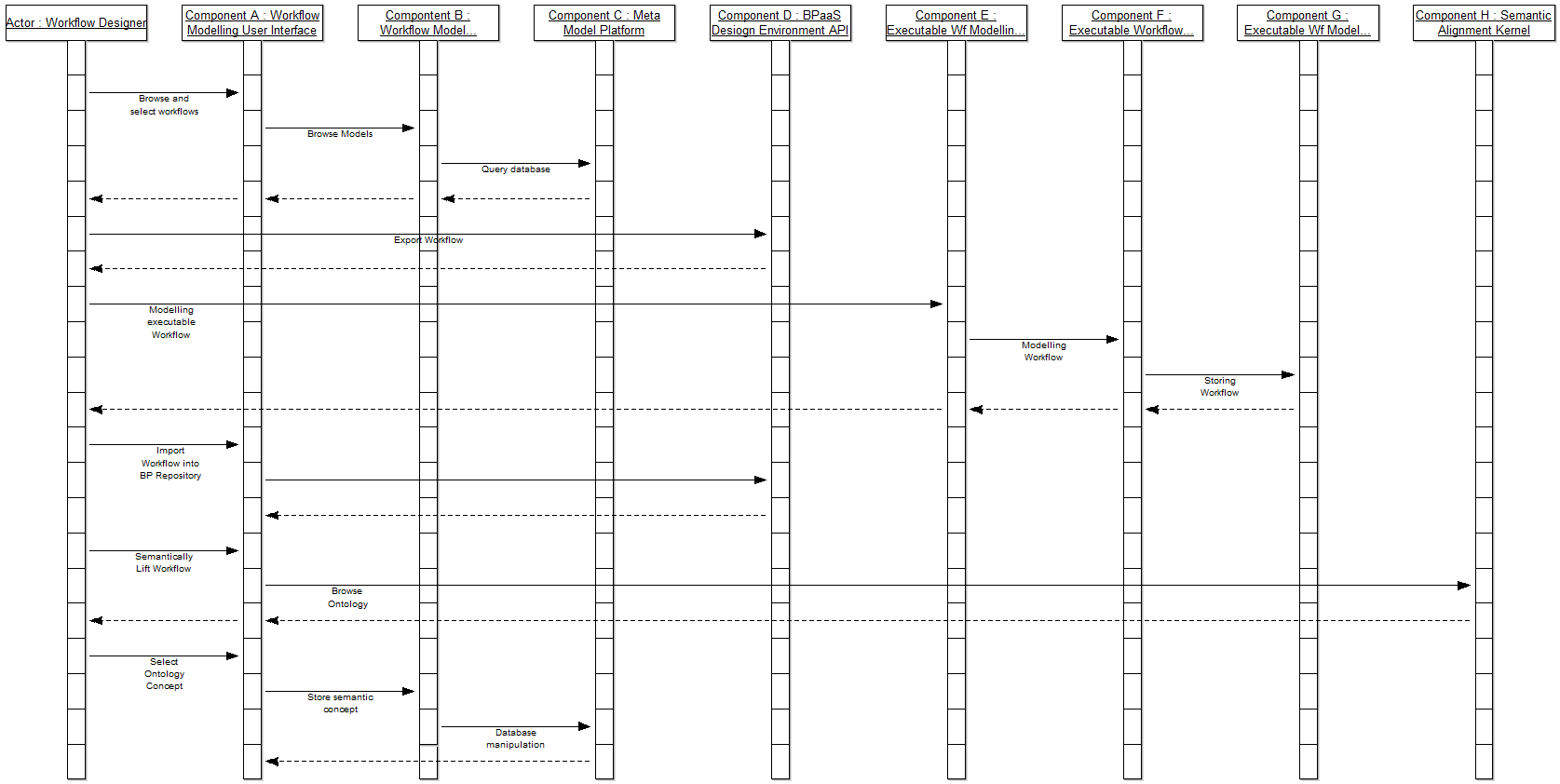


Figure 12 Sequence Diagram – DE-UC-4-Executable Workflow Design

Table 8 BPaaS Design Environment – Use Case 4 – Executable Workflow Design

Copyright © 2016 BOC and other members of the CloudSocket Consortium

5.2.5 KPI Definitions and Meta Model Completion

Use Case id	DE-UC-5-KPI Definitions and Meta Model Completion
Title and Description	In addition to business processes and their semantic lifting, additional meta data can be defined. Current identified meta data are: • Key Performance Indicators (KPI) from a business oriented viewpoint. Those KPIs are typically not cloud specific but business domain specific while giving an indication on how to map them to cloud specific measures. • Decision Models to indicate business rules for deployment and execution • "Business process based Service Requirements" map service descriptions to business process tasks as an extension of BPMN. Service requirements in form of business, technical, legal or data related statements provide additional domain specific description and when semantically annotated also semantics. For completeness reasons those additional aspects are mentioned here, as they are not specified in the BPMN standard but provide valuable information, which can be extracted and exploited during the later stage of the BPaaS lifecycle.
Actors	CloudSocket Broker – Business Process Designer CloudSocket Broker – Workflow Designer
Use Case Objective	The BPaaS Design Package is completed with these additional aspects.
Pre-Condition	Business process and / or workflows must be defined
Process Dialog	KPI definitions are extensions of the BPMN modelling language with the required OWL-Q concepts; hence the modelling interaction is the same as above either for KPIs with respect to business process, or KPIs with respect to workflows.
Variations	n.a.
Post Condition	The complete meta data package for a business process is defined.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Diagram

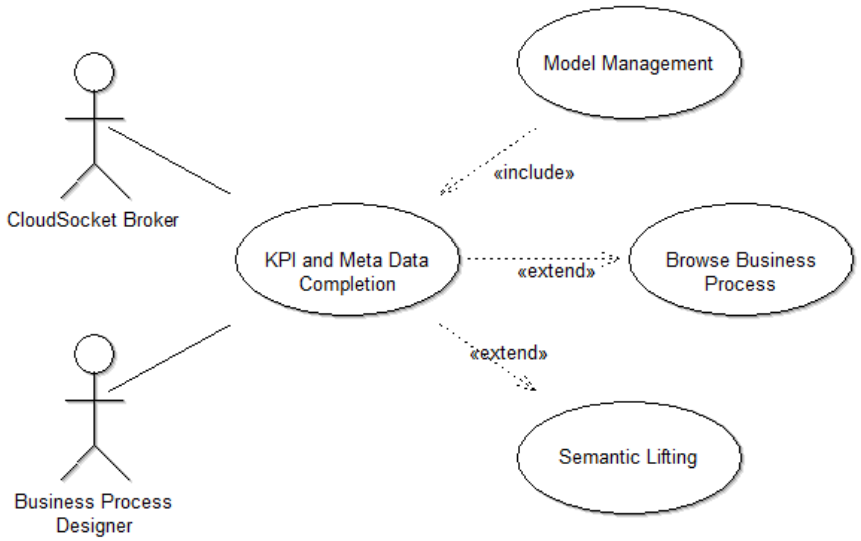


Figure 13 Use Case Diagram – DE-UC-5-KPI Definitions and Meta Model Completion

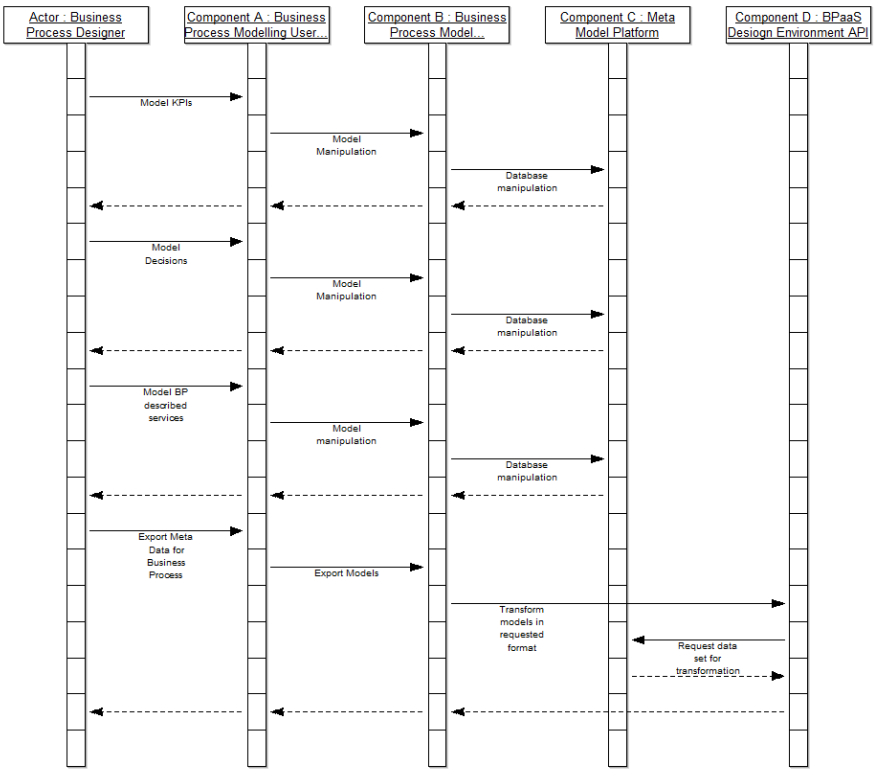


Figure 14 Sequence Diagram – DE-UC-5-KPI Definitions and Meta Model Completion

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Table 9 BPaaS Design Environment – Use Case 5 – KPI Definitions and Meta Model Completion

5.3 Components

Three layers: (a) one for the user interface, (b) one for model-specific functionality and (c) the third one for model management, have been identified.

5.3.1 User Interface Layer

This layer contains all the graphical interface components to interact with the users of the BPaaS Design environment:

- Business Process Modelling User Interface:
  This user interface provides Web interfaces to the corresponding features of the BP model component. An Authentication service provides access management, the Web-Modeller provides interfaces for model management and model design, the transformation interface provides import / export features to different formats, a Dashboard enables the representation of KPIs in correspondence with the business process. Semantic lifting of business processes is provided inside the model editor.
- Workflow Modelling User Interface:
  This user interface enables the creation of Workflow templates and the semantic annotation. It will be added to the aforementioned Business Process User Interface.

Both user interfaces interact with the components on the modelling layer, which have been configured for BPaaS usage and are built on top of BOC ADOxx Web-Application. Hence, the user interface is provided by the underlying meta modelling platform ADOxx, whereas the modelling language that can be used is different.

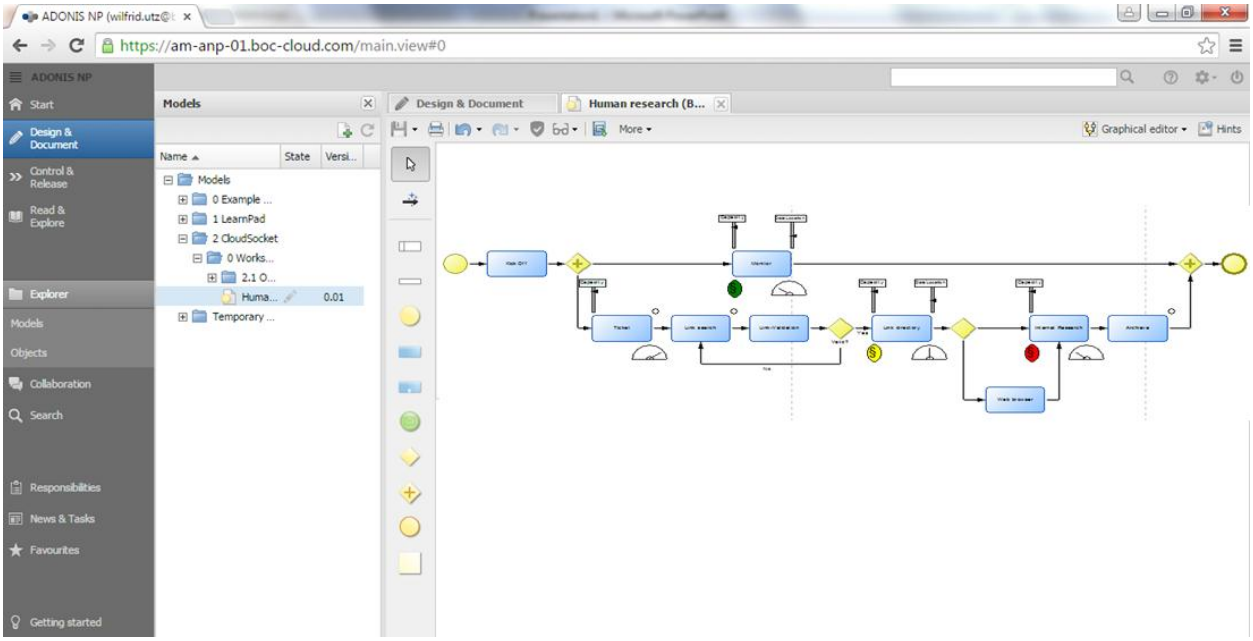


Figure 15 BPaaS Design Environment – Domain Specific Business Process Designer User Interface Mockup

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Figure 15 indicates the user interface demonstrating a business process with domain specific key performance indicators as additional graphical representations.

5.3.2 Modelling Layer

This layer provides modelling functionalities for different modelling languages. In general, there are four major functionalities for modelling: (a) Graphical Design, (b) Conceptual Query, (c) Simulation and (d) Transformation of Models.

Such major functionality is broken down to concrete features and mapped to the respective modelling language. Such model functionality can be distinguished into: (i) Generic implementation, hence in a modelling language independent way, (ii) Domain specific implementation, hence a one to one implementation for a specific modelling language, as well as (iii) Hybrid implementation, which a generic implementation requires special configuration to be used for a particular modelling language.

Those architectural baselines require identifying three different modelling components for:

1. Business Process Modelling
2. Workflow Modelling
3. Semantic Alignment Modelling.

Not all aforementioned major functionality needs to be applied to all three components; hence we define the following functional capabilities for each component:

- Business Process Modelling Component
  User management, model management and model design are some of the main functionalities supported by this component and required for Business Process Modelling.
  Business process analysis is offered in the form of support features, such as indicating which processes are available, or listing processes with a certain textual annotation.
  The Transformation capabilities enable exporting the graphical representation in form of an image, as well as exporting the business process in form of a BPMN.
  The features for semantic annotation of business processes are essential for Cloud Socket. Semantic lifting of business processes according to domain specific requirements is essential to enable an alignment.
  This component is based on the meta modelling platform ADOxx, and hence uses configuration of its components, the scripting of add-ons as well as the implementation of additional features.
- Workflow Modelling Component:
  The Workflow modelling component provides similar features with respect to those offered by the business process modelling component. The actual design and configuration of a workflow is performed in a separate so-called “Executable Workflow Designer” component.
  This enables a higher flexibility and reduces the vendor dependencies, as any workflow designer that is compatible with the Workflow Engine operating in the cloud, can be used. The alignment and CloudSocket relevant parameters can be modelled in the Workflow Modelling Component, and the workflow engine can be designed in the separate tool. The executable workflow is then imported back to enable semantic annotation and discovery but not for the sake of modifying the executable workflow.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Hence this component provides user management, model management and model design features for workflows.
The transformation capabilities – that are provided through the model environment API are essential to export workflow templates and import executable workflows from any other tool in order to store all workflows in one BPaaS model repository.
Semantic lifting of the workflow templates enables the discovery of workflows according to alignment parameters.

- Semantic Alignment Kernel:
  The semantic alignment kernel is a light-weight model editor without own user interface that enables the semantic lifting of business processes and workflows (templates). Hence modelling features for annotations are provided. In addition, full fletched discovery and analysis capabilities are established to enable the discovery of workflow (templates) for business processes and from respective business process requirements.

5.3.3 Meta Model Platform Layer

The Meta Model Platform Layer provides the BPaaS model repository, where all business processes, workflow templates, executable workflows and the corresponding semantic annotations are managed.

Two components are identified:

- Meta Modelling Platform
  The meta modelling platform consists mainly of a generic – meta model based – database with the corresponding meta model interpreter. This meta model interpreter is then configured by a modelling language. Hence, the following modelling languages are configured: (a) business process modelling, (b) workflow templates and workflow modelling, (c) semantic alignment, (d) KPI modelling, (e) Decision modelling, and (f) Business requirement modelling. The meta model platform provides a repository of all business processes, workflow templates, executable workflows and semantic annotations.
  User and security management is provided by the platform.
  The ADOxx meta model platform is the basis of the BPaaS Design Environment as it introduces conceptual flexibility and hides the meta model complexity from the other components.
- Meta Model API
  The model repository API is a configuration of interfaces that correspond with the meta modelling platform to provide interfaces for:
  - BPMN Import / Export:
    The BPMN import is used to import business process models and executable workflow models. The BPMN export is used to export business processes, workflow templates or executable workflows and interacts with the BPaaS Allocation Environment
  - DMN Import / Export:
    Decisions and rules will be used to guide the deployment specification of the BPaaS in the cloud and constitute a fundamental input for the BPaaS Allocation Environment.
  - RDF Import / Export:
    RDF Export provides semantic meta data about the business processes and workflows and hence enables the introduction of relevant SLA, deployment or allocation guideline information.
  - BP and Wf Discovery:

Copyright © 2016 BOC and other members of the CloudSocket Consortium

This interface provides a mapping file that can be used by a marketplace or any other environment to identify corresponding business processes to workflows and vice versa. This interface interacts with the Marketplace.

- KPI Import / Export:
  This interface requests and introduces KPIs for business processes and workflows and interacts with the BPaaS Evaluation Environment.

5.3.4 Executable Workflow Designer

The executable Workflow Designer is a third party tool that is compatible with the Workflow Engine in the BPaaS Execution Environment. Actually, the Workflow Designer should be part of the Workflow Engine. This tight linkage between the Workflow Designer and Workflow Engine is created as there are tool specific dependencies, and in order to enable practical and reliable workflow solutions.

In order to enable the business and IT Cloud alignment, the created executable workflow models are imported into the aforementioned Workflow Modelling Component. This is possible, as the BPMN standard allows the storage of vendor specific attributes, hence storing the BPMN workflow file for a particular Workflow Engine in the aforementioned Workflow Modelling Component will not harm any specific workflow configurations.

This approach enables both:

(a) reliable workflow configurations as the CloudSocket Broker selects the Workflow-Engine and Workflow Designer that best fits its needs,
(b) business and IT-cloud alignment via the meta model platform, as workflows from the third party system are imported and semantically annotated.

As this tool is a third party Workflow Designer, it is not further described, except that it is expected to provide typical design features for executable Workflows, as well as an import / export of BPMN files to interact with the Workflow Modelling Component. Figure 12 indicates the user interface demonstrating an executable workflow model, including the technical description.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

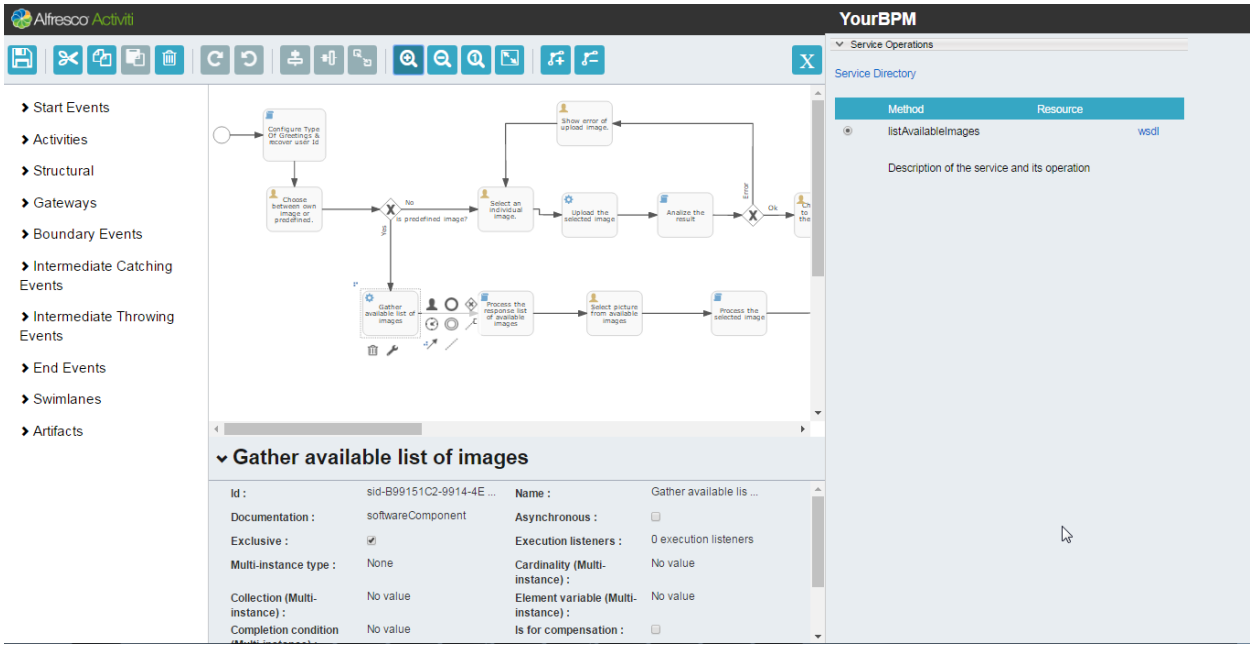


Figure 16 BPaaS Design Environment – Executable Workflow Designer User Interface Mockup

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Component diagram

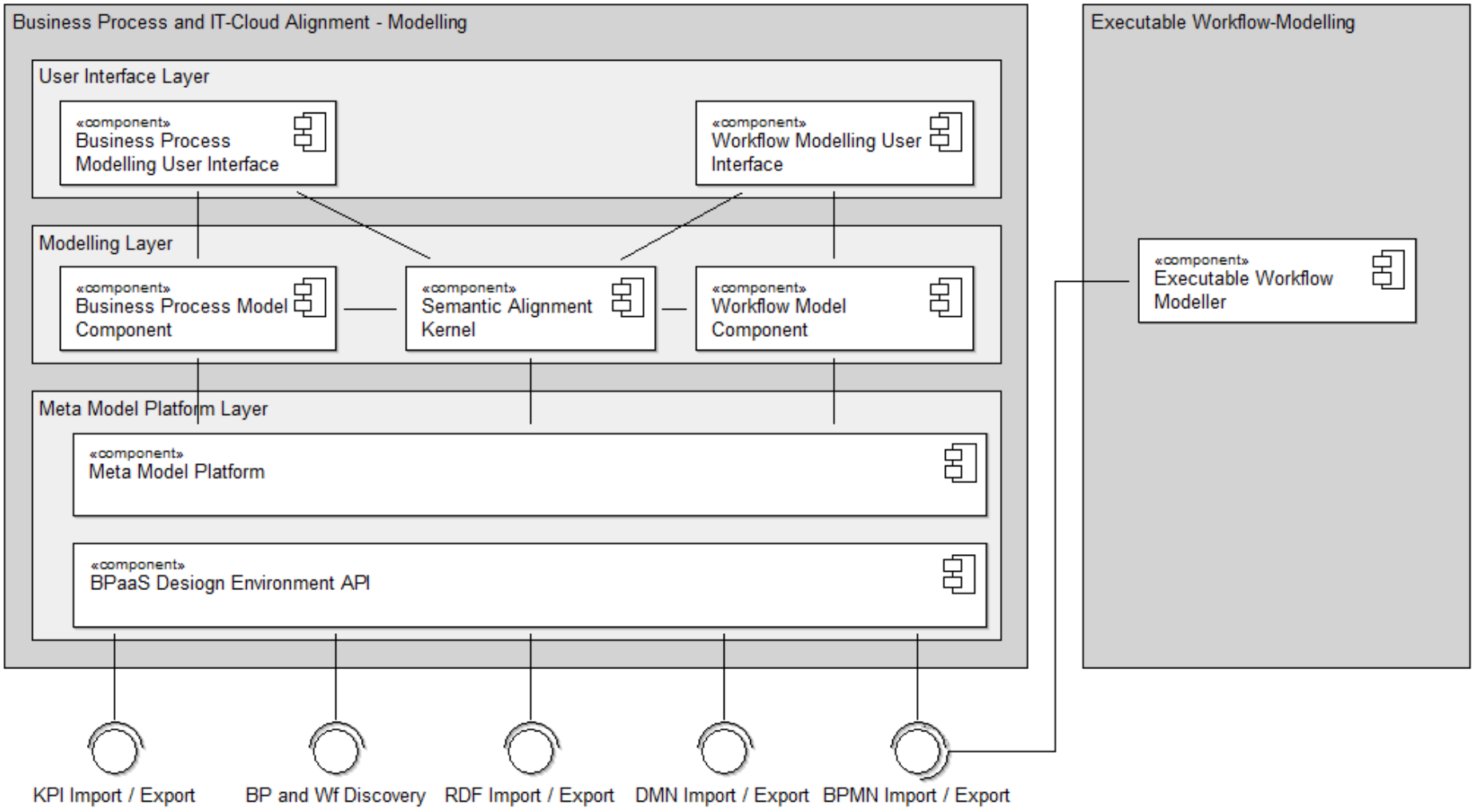


Figure 17 BPaaS Design Environment - Component Diagram

Copyright © 2015 BOC and other members of the CloudSocket Consortium

5.4 Research Contribution

An overview of the BPaaS Design research contribution (D3.1 2016) is sketched in Figure 18. On the left-hand side, there is the human-interpretable BPaaS Modelling Environment. On the right hand side, there is the machine-interpretable ontological representation and the inferencing for the smart business and IT alignment.

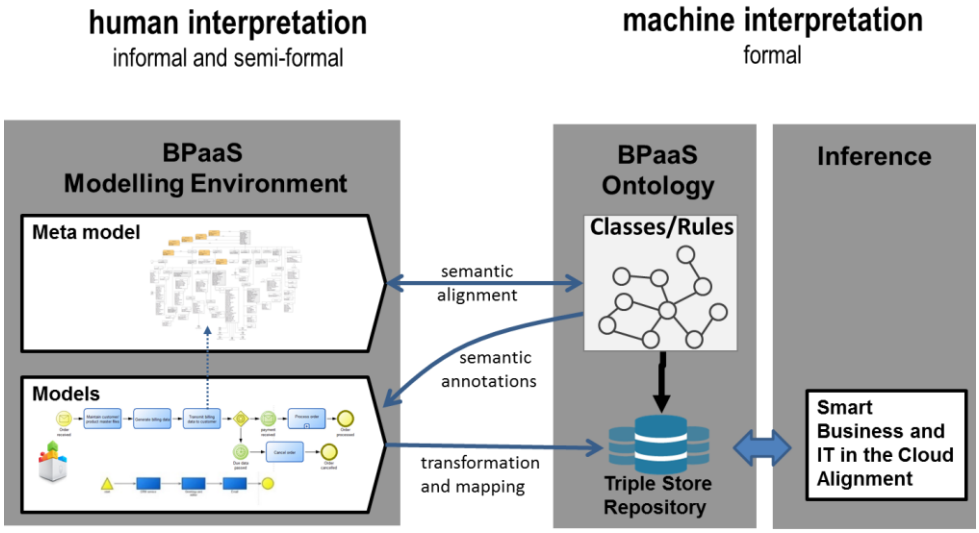


Figure 18: Overview of the BPaaS Design Environment and Smart Business IT-Cloud Alignment

The ontology defines the semantics of the meta model elements. This means, in particular, that it contains class definitions for the modelling elements of business processes and workflows. Furthermore, it also contains additional definitions, which can be used to annotate models and model elements. The facts of the knowledge base are created by transformation, which creates instances and maps them to the corresponding classes of the ontology.

The alignment can be regarded as a four step approach:

1. Business processes and workflows are modelled using the BPaaS Modelling Environment and the semantic lifting support via the BPaaS Ontology. Both the business process models and workflows are annotated with a description of their functional and non-functional capabilities.
2. The information of the models is translated into a machine-interpretable representation (called Triple Store Repository).
3. The business process model is mapped to one or several appropriate workflow models by comparing the business process requirements with the workflow descriptions. This workflow identification is done by the inference engine for smart business and IT in the cloud alignment using the mapping rules of the BPaaS Ontology component.
4. Finally, the BPaaS Design Package is created. It consists of the domain-specific business process and the executable workflow model, the key performance indicators and additional information which is relevant for allocation and deployment.

The prototype has a focus on the following parts: The business process and workflow models, the respective semantic annotations and the transformation and mapping rules for the business and IT alignment. The creation of the BPaaS Design Package still requires manual work by Cloud Broker experts, but could be considered to be automatized in the further development of the BPaaS design prototypes.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

A comprehensive description of the BPaaS modelling method and the related model types is provided in chapter 2 of Deliverable D3.2. Three model types are relevant for the alignment: (a) Business Process Model, (b) Workflow Model and (c) Service Description Model. The first two model types are expressed with the standard BPMN 2.0, while the third one represents a cloud specific description concept, which is based on the FODA approach (Kang 1990).

Based on the latter concept, both the business process and the workflow model can be semantically enriched according to both functional and non-functional aspects.

Both functional business process requirements and workflow descriptions are specified with the following attributes: (a) With the first attribute, the tasks or groups of tasks are categorised by assigning hierarchies from the APQC Process Classification Framework (APQC 2014); (b) Additionally, the semantics of the tasks can be specified by assigning an object and an action from a predefined taxonomy.

Conversely, non-functional requirements and specifications are determined based on the Cloud Service Level Agreement Standardisation Guidelines (C-SIG SLA 2014). These guidelines are an outcome of the European 2020 initiative “Digital Agenda for Europe” and have been published in order to standardise and streamline the terminologies and understanding of Cloud Service Level Agreements.

The mapping between non-functional business process requirements and non-functional workflow descriptions is done by matching rules, which are part of the smart business and IT alignment. The matching is not always map to a 1:1 correspondence. It can happen that one or more elements in the business process requirements can be mapped to one or more non-functional descriptions, hence we claim that their relationship is *n:m.*

The BPaaS Ontology is an extension of the ArchiMEO ontology. The BPaaS Ontology includes all concepts required to describe cloud specific requirements in order to achieve a smart alignment of business and IT. The cloud-specific extensions were determined from the analysis of the business scenarios as well as from competency questions and are described in detail in Deliverable D3.1 (CloudSocket 2015b).

The ontologies can be publicly accessed using the following links:

- APQC: https://github.com/BPaaSModelling/APQC-Ontology/blob/master/apqc.ttl
- Functional Business Process Description: https://github.com/BPaaSModelling/Functional-Business-Process-Description-Ontology (object & action taxonomy)
- BPaaS: https://github.com/BPaaSModelling/BPaaS-Ontology/blob/master/bpaas.ttl

There are different ways of implementing semantic lifting for weaving between the different modelling. In the prototype, we have implemented the deep integration with a web service. The web service creates and maintains the link to the different ontologies. The modelling environment calls the web service for a concept by providing the context and the web service returns the resulting classes and instances, depending on the context as shown in Figure 19.

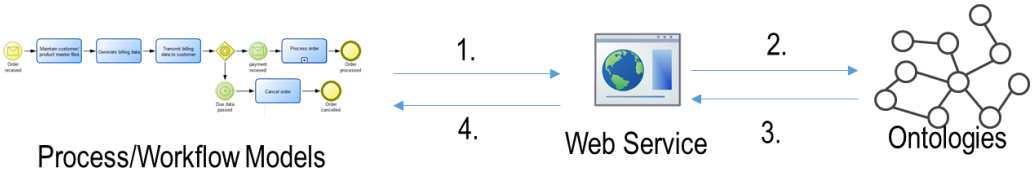


Figure 19: Modelling Environment - Web Service Communication

The modelling environment receives at the beginning the link to the web server. While the human models the diagram, the properties can be set. Properties that need to receive a conceptual annotation, provide the possibility to make a call to the web service (1). The web service receives the request containing contextual

Copyright © 2015 BOC and other members of the CloudSocket Consortium

information and queries the ontology accordingly (2). The result set is returned (3) and is being processed by the web service according to the interface description to the modelling environment. Once the processing is done, the web service returns the enriched result set (4) to the modelling environment that makes sure that the annotation is placed at the right place in the meta data of the modelling element.

Rules are used to map the business language to the technical one, while queries are used to compare business process requirements with workflow descriptions. Rules are used to convert one element value that resides in the business layer to an element value that resides in the workflow layer. Section 6 of Deliverable D3.2 provides examples of rules as well as a concrete example on the implementation of the smart business IT alignment.

Figure 20 shows the prototype user interface comparing “Social Media Campaign Process” to two workflows in the repository. For the given Business process entitled “Basic Social Media Campaign”, only one workflow matches, i.e. Workflow Social Campaign – Basic1. More in detail, the business process has two groups of activities semantically annotated, i.e. labelled as Group 1 and Group2. The two workflows have two lanes (or again groups) each semantically annotated, i.e. labelled as WFD1, WFD2 (acronym of WorkFlowDescription). The first matching row in the UI shows that both Group1 and Group2 match with WFD1 and WFD2, respectively. Conversely, the second row shows that there is no workflow description (i.e. no lane or no group) that matches with Group1 and neither with Group2.

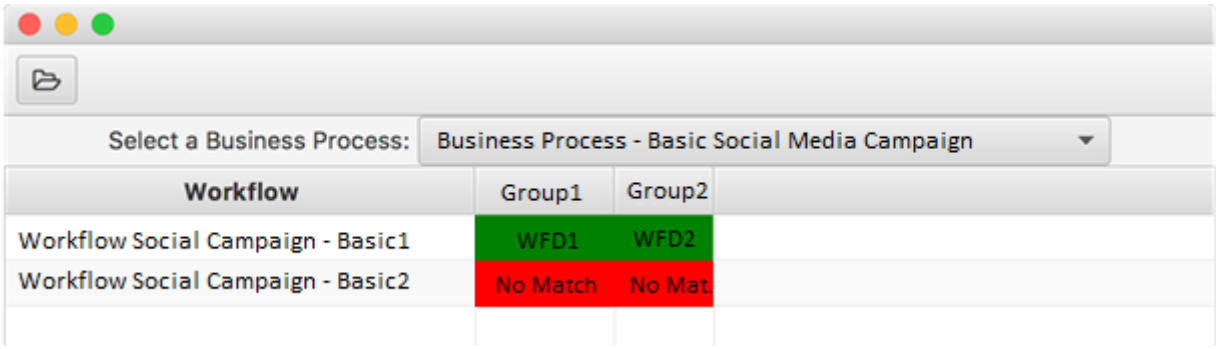


Figure 20: Prototype Showing Matching Results

Copyright © 2015 BOC and other members of the CloudSocket Consortium

5.5 Roles

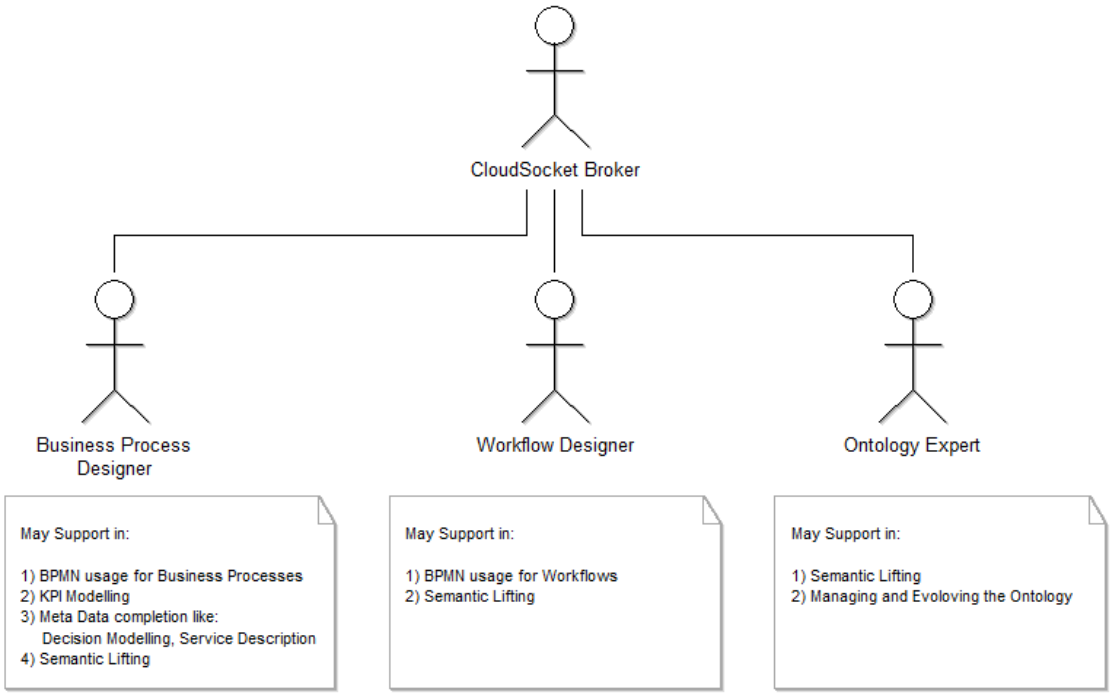


Figure 21 BPaaS Design Environment - Actors

The following actors are proposed for the BPaaS Design Environment:

The CloudSocket Broker is an organisation that provides a marketplace for business processes towards the CloudSocket Customers. Hence the competence in selecting, defining and providing business processes and of course in taking the business risks in creating business processes that are not sufficiently bought from CloudSocket Customers is overtaken by this organization. Depending on the competency, this organization will compensate business and / or technical skills with consultants.

A CloudSocket Broker with business background (see Business Process Designer role), will be capable of formulating business processes on domain level, but will need support to design those processes on workflow/execution level.

A CloudSocket Broker with technological background (see Workflow Designer role), will be capable of formulating technical workflows, but will need support to market the technical solutions via business processes.

The last role is the Ontology Expert. As the maintenance of the alignment ontology is only a minor aspect in the whole scenario, it is expected that the ontology is either bought, or the expertise is introduced by a consultant. This alignment ontology is expected to be a research result of the project. Therefore, the role is only briefly mentioned for completeness reasons.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

5.6 Data Interface

The meta model platform stores meta data for each model, which can be either extracted in raw format, or by post-processing transformation scripts to create one of the requested formats. Manual export into files and Web-Service or REST invocation is possible.

5.6.1 BPMN Interchange

The BPMN DI format is used for business process and workflow exchange. The BPMN 2.0 specification defines formats that contain:

- Meta information,
- Process models that define the sequence and semantics and
- Process diagrams that have the information of the visual representation (e.g. coordinates of the instances) of the process models.

Process automation can be done with BPMN DI.

The RDF format is used for the semantic alignment of the business processes.

5.6.2 KPI, Meta Data and Decision Model Interchange

KPIs have to be exchanged from the BPaaS Design Environment to any other Environment for the following layers: (i) domain specific business process, (ii) executable workflows, (iii) deployment and operative process monitoring.

The BPaaS Design Environment allows interchanging KPI models using the OWL-Q language.

Additional meta data interchanges, such as workflow and business process relations, business requirements or semantic information are provided in proprietary formats as well as in RDF.

For the exchange of deployment rules, the decision models format DMN is provided.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6 BPAAS ALLOCATION ENVIRONMENT

6.1 Introduction

The BPaaS Allocation Environment allows a CloudSocket Broker to select a BPaaS Design Package (previously created via the BPaaS Design Environment) and create a BPaaS Bundle ready to be published in the BPaaS Marketplace and be deployed in the BPaaS Execution Environment.

A BPaaS Bundle binds a BPaaS Design Package - taken from the BPaaS Design Environment - with the concrete Atomic SaaS and IaaS/PaaS services that will be invoked by the executable workflow and support its execution, respectively. A BPaaS Bundle also contains rules to guide the BPaaS Execution Environment in reconfiguring across clouds and layers the BPaaS, when the respective need arises.

A BPaaS Design Package contains a Workflow Model that can be abstract or executable. Hence, the BPaaS Bundle created from it, can be abstract or executable too. An abstract BPaaS Bundle can be published in the BPaaS Marketplace but cannot be deployed and executed. A BPaaS Customer can submit a request for an abstract BPaaS Bundle in order to show his/her interest in buying it. Hence, the CloudSocket Broker can decide which abstract BPaaS Bundle to implement and publish as an executable BPaaS Bundle by selecting bundles according to certain criteria, including the maximum number of abstract BPaaS bundle requests.

In the following, the executable BPaaS Bundles are referred to as BPaaS Bundles while the abstract BPaaS Bundles are treated as a particular case of executable BPaaS Bundles.

In the creation of a BPaaS Bundle, the CloudSocket Broker is responsible for defining the overall pricing and SLA for the bundle as a whole, by taking into account the price and the SLA of all the resources/services – e.g. atomic services and cloud infrastructures - involved in the bundle. The CloudSocket Broker will get paid from the BPaaS Customer using the BPaaS, will pay for the resources consumed by the deployed BPaaS Bundles and will be held responsible for the fulfillment of the agreed SLA. Creating and selling a BPaaS Bundle will therefore require critical and complex business decisions and will imply a degree of risk that will definitely require human intelligence and understanding to be properly evaluated and accepted. The Allocation Environment will however provide smart tools to support the Broker in defining the overall pricing and SLA of each BPaaS Bundle.

6.1.1 Description and structure of a BPaaS Bundle

A **BPaaS Bundle** is a data structure containing all the information required by the Marketplace to show and sell the bundle, and by the Execution Environment to deploy, execute, account, monitor and assess a BPaaS. Such information includes:

- **Business Process Model:** expressed in BPMN2.0 format while its respective image, also contained in the BPaaS Design Package, is also included in the bundle.
- **Executable Workflow Model**: expressed in BPMN2.0 format while its respective image from the BPaaS Design Package is also included in the bundle. This executable workflow model contains all the technical details required for its proper execution by a Workflow Engine. It might contain invocations to atomic services selected from an Atomic Service Registry, and invocations to software components taken from a Software Component Registry. An atomic service can be either a concrete service available in the cloud, or an abstract service not yet mapped to a concrete one. In order to execute a workflow model, each of the abstract services associated to workflow tasks must be mapped to concrete ones, and each software component also associated to a workflow task must be mapped on a compatible cloud infrastructure offering,

Copyright © 2015 BOC and other members of the CloudSocket Consortium

that is an infrastructure where the component can be technically deployed. The Software Component Registry provides all the information required to deploy, start, stop and undeploy a software component component on specific operating systems that could be supported by cloud infrastructure offerings.

NOTE: the Allocation Environment does not allow to create or edit a workflow model: it just allows to select an existing workflow model from the BPaaS Design Environment and create a new BPaaS Bundle "out of it". Thus, the workflow model is generated exclusively in the BPaaS Design Environment.

NOTE: the workflow model references the original business process model it has been derived from. The workflow-to-business process linking is created and maintained by the BPaaS Design Environment.

- **Atomic Service allocation:** This is the mapping of each abstract atomic service invoked by the workflow model to one or more concrete and compatible (i.e. same interface definition) atomic services available in the cloud, which are registered in the Atomic Service Registry. One from the set of concrete services mapped to an abstract one will be chosen at run-time by the BPaaS Execution Environment according to allocation/adaptation rules which are prescribed in the BPaaS bundle. The CloudSocket Broker, however, has also the capability to fix the mapping between one abstract atomic service to a concrete one, such that runtime allocation by the Execution Environment is not needed for this abstract atomic service.
- **Software Component allocation**: As the software component is associated to particular VM requirements within the Software Component Registry, a list of VM offerings drawn from the Cloud Provider Registry satisfying these requirements is shown to the CloudSocket Broker, which then selects the most suitable one. In the BPaaS Bundle specification, apart from identifying the VM offering selected, also the other VM offering candidates are included for reconfiguration/adaptation reasons (e.g., to migrate from one VM to another one when the former VM becomes problematic). The respective adaptation cases are covered by the corresponding adaptation rules which are also included in the BPaaS Bundle.
- **KPI Model:** This section defines which metrics can be used for defining SLOs (as conditions over metrics) within the SLA of a BPaaS bundle. The description of a metric will basically consist of an identifier, a description, and possibly a formula which includes standard math operators operating over other metrics previously defined and stored in the Metric Registry. We foresee the possibility for defining and exploiting domain-specific metrics and domain-independent metrics available for any BPaaS Bundle (e.g., number of concurrent users, concurrent process instances, storage space consumed, up-time percentage in the last 30 days, etc.).
  The formula of *raw_availability* metric, for instance, could be *uptime / considered_time_period* where *uptime* is another metric mapping to the total time that the BPaaS was available and *considered_time_period* is a metric configuration constant indicating the time period of measurement. The BPaaS Allocation Environment will support the Broker in defining or composing metrics as well as checking for circular dependencies within the derivation hierarchy of composite metrics. Apart from being used in SLOs, conditions over metrics can also be exploited in defining BPaaS adaptation rules. In particular, such rules include event patterns, where each event in a pattern can be mapped to a metric condition whose violation can trigger this event. The metrics supported by the CloudSocket platform are stored in the Metric Registry.
- **Service Level Agreement (SLA):** This information defines in a quantitative and technically verifiable way the level of service that will be guaranteed by the entire BPaaS once the BPaaS Bundle will be deployed into the BPaaS Execution Environment. The SLA will be formally defined as a collection of SLOs (Service Level Objectives), where each SLO is associated with a metric condition defined as a predicate (i.e., a boolean function) based on the metric already defined (e.g., *uptime_percentage* > *99.9*). Each SLO can be associated to a specific penalty to be applied when this SLO is violated (i.e., if the (daily) uptime_percentage > 99.9 is violated three times in a month, then apply a discount of 10% on the current monthly price). The CloudSocket Broker is responsible for providing a comprehensive and coherent SLA for the BPaaS Bundle, taking into account the SLA of each (real) concrete atomic service and of each VM offering allocated.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

- **Pricing Model:** This information defines how much using the BPaaS will cost to the CloudSocket Customer. The CloudSocket Broker is responsible for providing a comprehensive pricing model for the entire BPaaS Bundle taking into account the pricing of each real atomic service and of each cloud infrastructure allocated.
- **Business Process Metadata:** This is the information required by the BPaaS Marketplace to assist the customers in browsing, selecting and purchasing a BPaaS Bundle. Indeed, it contains categories (according to the APQC taxonomy) and tags used to filter the BPaaS Bundles offered in the BPaaS Marketplace and business process-related information (e.g., pictures and textual description of the business process) to let the customer select a specific bundle matching his/her needs. The Business Process Metadata will allow for instance a customer browsing the marketplace to filter only the Bundles implementing an Invoicing process whose description contains the term "consultancy" and using only Atomic Services hosted in Germany.

NOTE: an abstract BPaaS Bundle contains an abstract Workflow Model instead of an executable one, the Business Process Metadata and the hypothetical Pricing model that shows to the customer how much the BPaaS Bundle will cost once the Broker will publish an executable version of it. It does not include the allocation information and the KPI\SLA Model as these are the technical details which would make the bundle concrete and would also lead to the need of concretising the actual pricing model to be offered.

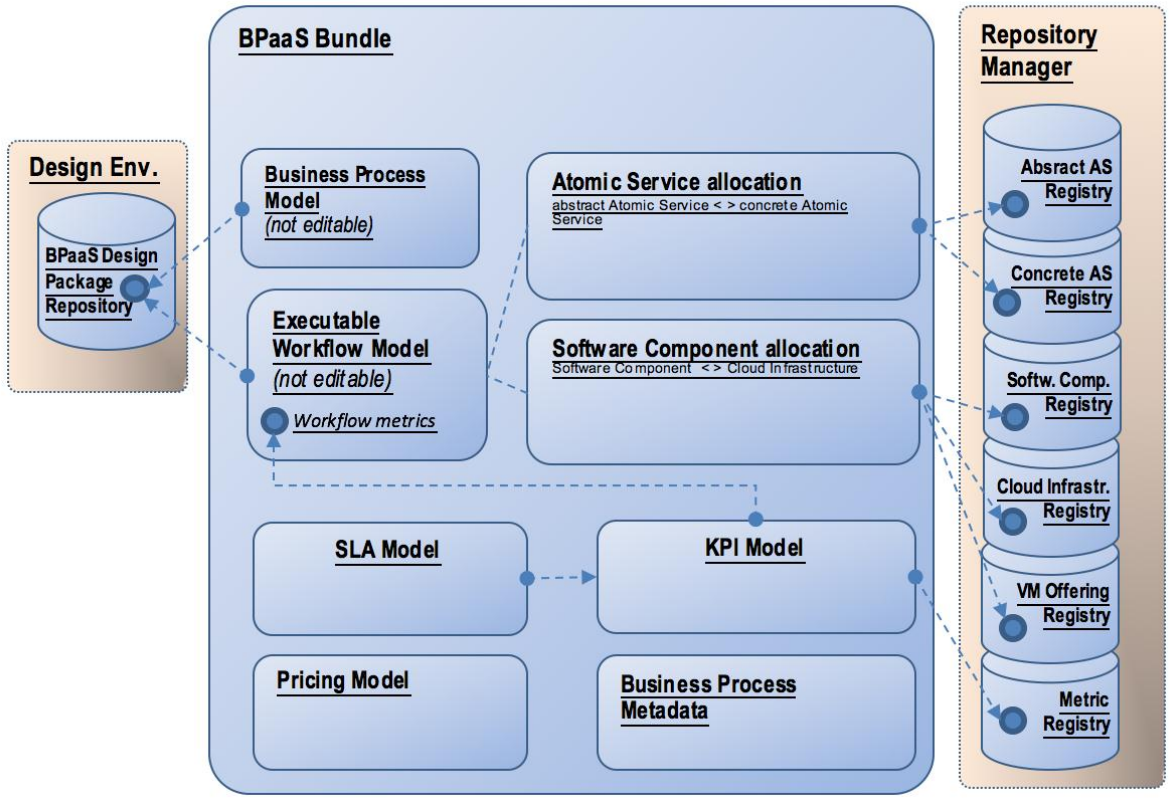


Figure 22 BPaaS Bundle Elements

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.1.2 Creation of a BPaaS Bundle

The CloudSocket Broker creates a draft BPaaS Bundle using the Bundle Instantiator, by selecting a BPaaS Design Package from the BPaaS Design Environment. The draft Bundle contains a copy of the Workflow and Business Process Model contained in the selected BPaaS Design Package, while the other parts of the aforementioned Bundle data structure are initially empty.

Once created, the draft bundle is immediately opened in the Bundle Designer for the Broker to edit. Whenever the Bundle is saved, a consistency check is performed server-side by the Bundle Manager. If all the parts of the Bundle have been properly configured, the Bundle is saved as Consistent.

When a BPaaS Bundle is in a Consistent state, the Bundle Designer allows the CloudSocket Broker to publish it in the marketplace by issuing a dedicated command. The Bundle Designer then delegates the Bundle Publisher component to interact with the marketplace management API in order to store a copy of the BPaaS Bundle in the marketplace. A similar behaviour occurs when a CloudSocket Broker issues a command to unpublish a BPaaS Bundle; this action changes the bundle state from Published to Consistent. When a BPaaS Bundle is updated, the Allocation Tool checks that the state is not altered to Draft, hence it remains Published. If the CloudSocket Broker needs to do major modifications to the BPaaS Bundle changing the state to Draft, he needs to unpublish the BPaaS Bundle and then modify it..

Bundles are stored in a Bundle Repository, which is partitioned per CloudSocket Broker, thus supporting broker-based multi-tenancy; that is, every Broker is a tenant of the Bundle Repository, so that it can only see and manage its own Bundles.

The consistency constraint for a BPaaS Bundle depends also on its type. The main difference is that an abstract bundle does not require the allocation information which is, however, mandatory for an executable bundle. KPI and SLA information are both optional for an abstract BPaaS Bundle. Pricing model and Business Process Metadata are both mandatory for both abstract and executable BPaaS Bundles.

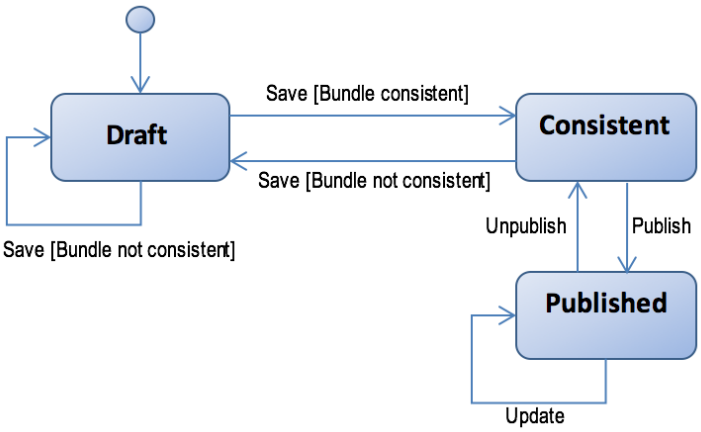


Figure 23 BPaaS Bundle States

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.1.3 Architectural overview

The web application provided by the BPaaS Allocation Environment follows a multi-tier architecture and the respective components are distributed across three main layers:

- a user-interface layer, responsible for the interaction with the users;
- a business-logic layer, implementing the logic of the services invoked;
- a persistency-management layer, responsible for saving, loading and managing data in the involved databases.

6.2 Functional Capabilities

The main functional capabilities of the Allocation Environment are:

- Creation of BPaaS Bundle
- Editing of a BPaaS Bundle
- Editing of a BPaaS Bundle / Atomic Service Allocation
- Editing of a BPaaS Bundle / Software Component Allocation
- Editing of a BPaaS Bundle / KPI Model Definition
- Editing of a BPaaS Bundle / SLA Definition
- Editing of a BPaaS Bundle / Pricing Model Definition
- Editing of a BPaaS Bundle / Business Process Meta-Data Editing
- BPaaS Bundle Publishing in the Marketplace

Since the CloudSocket Broker is the only responsible actor involved in the BPaaS Allocation Environment, the web application is not profiled and all the functionalities can be performed by any broker. All the use cases involved assume that a CloudSocket Broker is authenticated and authorized to use the web application.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.1 Creation of a BPaaS Bundle

Use Case id	**AE-UC-1 Creation of BPaaS Bundle**
Title & Description	A CloudSocket Broker creates a new BPaaS Bundle. The broker browses the BPaaS Design Environment, selects a BPaaS Design Package and confirms selection. A new draft bundle is created by the Bundle Instantiator component, opened in the Bundle Designer and finally saved in the Bundle Repository. The Broker can optionally edit the Bundle before saving it.
Actors	CloudSocket Broker
Use Case Objective	To create a new BPaaS Bundle in draft state into the Bundle Repository, and to immediately start editing that bundle via the Bundle Designer.
Pre-Conditions	The BPaaS Design Package needed by the CloudSocket Broker is stored in the BPaaS Design Environment in order to be selected using the Bundle Instantiator.
Process Dialog	1. The CloudSocket Broker launches the Bundle Instantiator. 2. The Bundle Instantiator invokes the BPaaS Design Environment API to get a list of the existing BPaaS Design Package along with meta-data to group and filter them. 3. The CloudSocket Broker uses the Bundle Instantiator browsing functionality to find and select the desired BPaaS Design Package. 4. The CloudSocket Broker issues the command to create a new Bundle based on the selected BPaaS Design Package. 5. The Bundle Instantiator creates a new draft Bundle in memory. 6. The Bundle Instantiator launches the Bundle Designer and provides the new Bundle as an input parameter (the Bundle is still not persisted). 7. The Broker optionally edits the Bundle using any of the Bundle Editing use cases. 8. The Broker issues the save command assigning a unique name to the new Bundle. 9. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured) then it is marked as Consistent (therefore eligible for publishing into the Marketplace).
Variations	7.a) If the BPaaS Design Package contains an abstract workflow model, not all Bundle Editing use cases are available (see the precondition of an use cases to see if it needs an executable workflow model to perform it). 8.a) The Broker aborts the edit and closes the Bundle Designer without saving. 8.b) A Bundle with the same name from the same Broker already exists in the Bundle Repository. The save command is rejected.
Post-Conditions	A new bundle is stored into the Bundle Repository.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

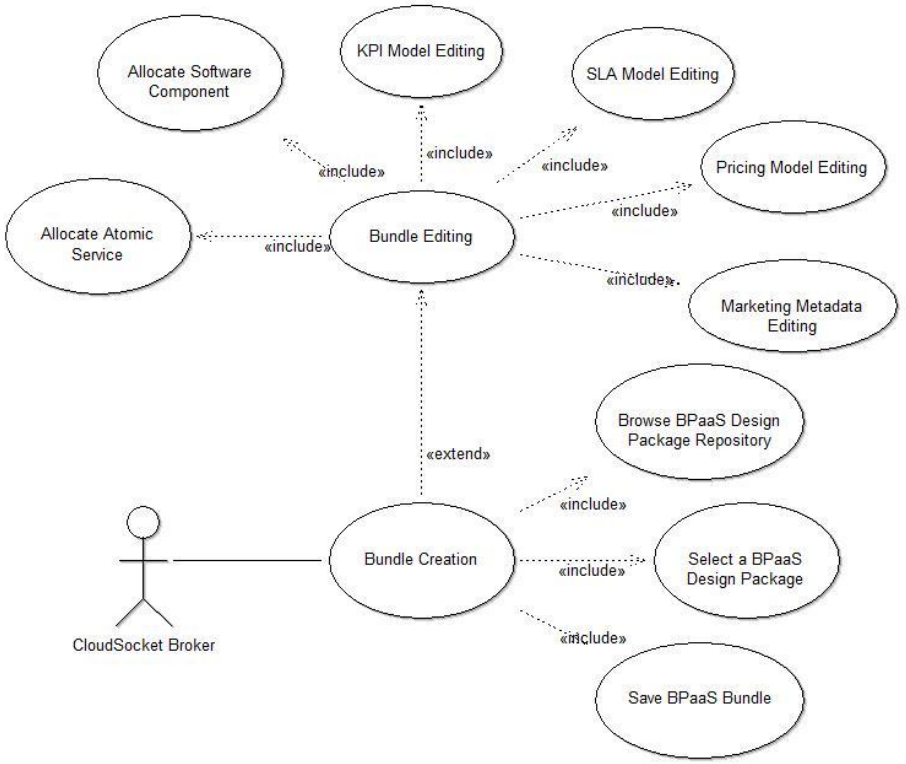
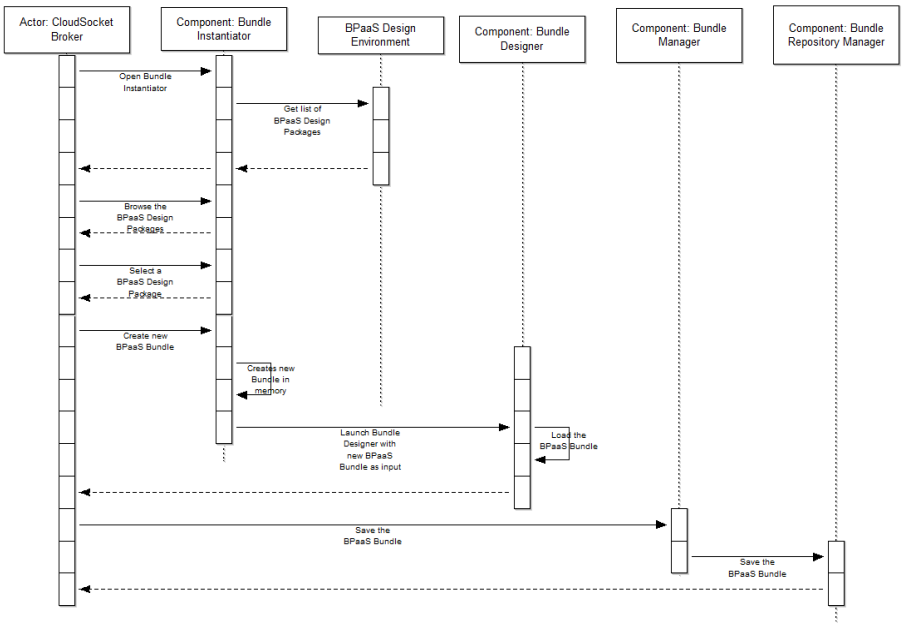
Diagrams	 **Figure 24 Use Case Diagram – AE-UC-1 Creation of BPaaS Bundle**  **Figure 25 Sequence Diagram – AE-UC-1 Creation of BPaaS Bundle**

Table 10 BPaaS Allocation Environment – Use Case 1 – Creation of BPaaS Bundle

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.2 Atomic Service Allocation

Use Case id	AE-UC-2-Atomic Service Allocation
Title & Description	A CloudSocket Broker allocates an abstract Atomic Service involved in the Workflow Model with one from the compatible concrete Atomic Services matching this service from the Atomic Service Registry. The CloudSocket Broker opens a Bundle in the Bundle Designer, enters the Atomic Service Allocation section, selects one of the abstract services involved in the Workflow Model, and chooses the concrete Atomic Service to be mapped to this abstract service. The concrete Atomic Services to be selected constitute a subset of services in the Atomic Service Registry which are compatible with the abstract Atomic Service at hand. Apart from the mapped/selected concrete atomic service, the other candidate ones matching the abstract service are also listed in the bundle description along with their pricing and SLA for adaptation purposes. Adaptation rules are then provided to indicate when the selected concrete atomic service will be substituted with the remaining candidate ones and according to which criteria. **NOTE**: for some abstract Atomic Services, the mapping to one compatible concrete Atomic Service (e.g., stateful, hard-to-reallocate services) will be unmodified-able once the BPaaS bundle is deployed. This actually maps to the inability to perform a service substitution or any other related adaptation action for such a service. As such, no adaptation rule can be specified for such a service. **NOTE**: In some cases, the selection of a concrete atomic service for an abstract one will be delayed until runtime. In such cases, the adaptation/selection rules will indicate which concrete atomic service is to be selected according to the current context. An example of an Atomic Service Allocation record - in case of delayed runtime selection - could be: Abstract Atomic Service *JIRA* mapped on • MyJIRA by provider X in UK (eligible only if its response time in the last month is less than 400 ms on average) • BestJIRA by provider Y in Germany (eligible only if its availability in the last year has been greater than 99%) • CheapJIRA by provider Z in Russia (eligible only if no other concrete Atomic Service turns out to be eligible)
Actors	CloudSocket Broker
Use Case Objective	To select a concrete atomic service for each abstract atomic one as well as map the abstract atomic service to a set of alternative concrete atomic services for adaptation purposes.
Pre-Conditions	The Bundle exists in the Bundle Repository. The Bundle has been created by selecting a BPaaS Design Package containing an

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	Executable Workflow Model (and not an abstract Workflow Model). The Executable Workflow Model in the Bundle contains at least a service task that invokes an abstract Atomic Service. The Atomic Service Registry contains at least one concrete Atomic Service compatible with the selected abstract Atomic Service. All metric definitions required to configure the eligibility rules for each concrete Atomic Service to be mapped already exist.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the Atomic Service Allocator UI component (part of the Bundle Designer). 3. The Atomic Service Allocator UI shows the list of the abstract Atomic Services mapping to service tasks in the Workflow Model of the Bundle. 4. The Broker selects an abstract Atomic Service from the list. 5. The Broker issues a command to open the mapping details for the abstract Atomic Service selected. 6. The Atomic Service Allocator UI invokes a service on the Bundle Manager to retrieve the list of the concrete Atomic Services which are compatible with the selected abstract Atomic Service. Compatibility actually means functional and quality-based matching of abstract to concrete Atomic Services. 7. The Bundle Manager invokes the API of the Repository Manager to retrieve the list of the concrete Atomic Services which are compatible with the selected abstract Atomic Service. 8. The Atomic Service Allocator UI shows the list of all the concrete Atomic Services matched with the selected abstract Atomic Service. 9. The Broker can select a matched concrete atomic service as the one to realize the abstract Atomic Service. He/she can also define eligibility rules indicating when to substitute the selected atomic service and according to which criteria (e.g., the less costly service with the highest availability encountered so far). At least one from the two aforementioned actions has to be performed (e.g., the first for fixed mappings and the second for dynamic ones). 10. The Broker repeats Steps 4-9 as desired, possibly until all abstract atomic services are covered. 11. The Broker confirms the changes and issues the save command. 12. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes, thus updating the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured), then it is marked as Consistent (therefore becoming eligible for publishing into the Marketplace).
Variations	8.a) If only one concrete Atomic Service turns out to be compatible with the selected abstract Atomic Service, then it is selected by default.. 8.b) If no concrete atomic service matches the abstract one, this is an indication that the functionality of the abstract service has to be realized from scratch. In that case, we will go to the next use case, once the respective software component has been realized

Copyright © 2015 BOC and other members of the CloudSocket Consortium

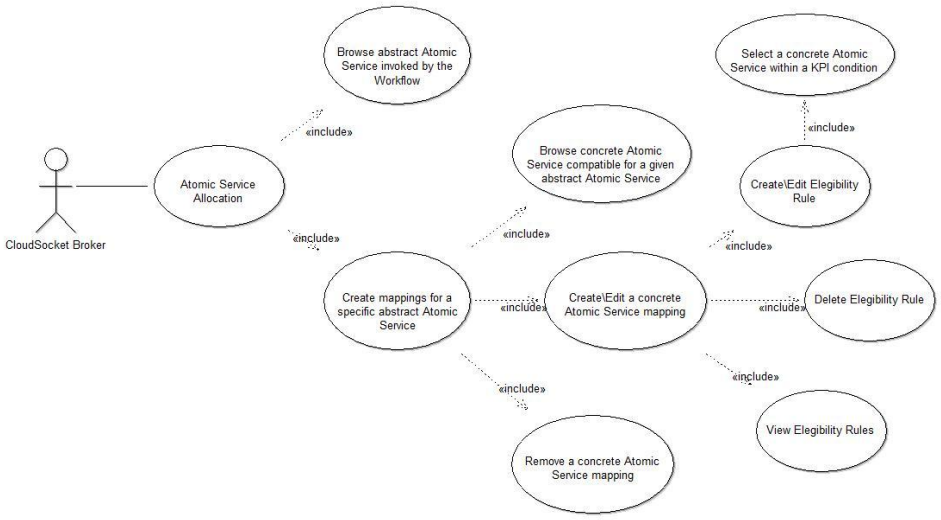
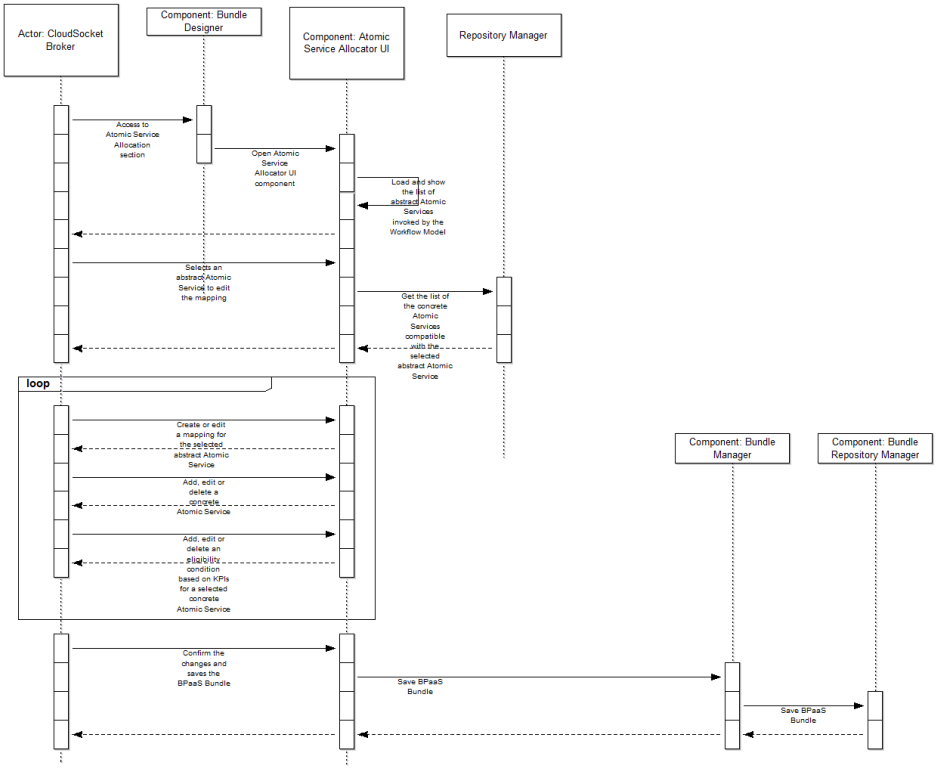
	and registered in the respective registry. 12.a) The Broker aborts the edit and closes the Bundle Designer without saving.
Post-Conditions	The Bundle is updated into the Bundle Repository.
Diagrams	 **Figure 26 Use Case Diagram – AE-UC-2-Atomic Service Allocation**  **Figure 27 Sequence Diagram – AE-UC-2-Atomic Service Allocation**

Table 11 BPaaS Allocation Environment – Use Case 2 – Atomic Service Allocation

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.3 Software Component Allocation

Use Case id	AE-UC-3-Software Component Allocation
Title & Description	CloudSocket Broker allocates a Software Component A CloudSocket Broker allocates a Software Component mapped to a Workflow Model task to a set of VM Offerings from the Cloud Infrastructures registered in the Cloud Provider Registry and selects one of these offerings as the one on which the component deployment will be performed by the BPaaS Execution Environment. The Broker opens a Bundle in the Bundle Designer, enters the Software Component Allocation section, selects one of the components mapped to Workflow Model tasks, and then selects the VM Offerings eligible to host this component. The VM Offerings are selected from the subset of VM Offerings in the Cloud Infrastructure Registry which are compatible - with respect to its VM requirements - with the Software Component selected. Each compatible VM Offering is listed along with its pricing and SLA. The broker selects one of the VM offerings as the one on which the component deployment will take place. He/she also defines adaptation rules indicating when to move to another VM (offering) when the currently selected one fails for some reason. **NOTE**: For some Software Components only a direct, static mapping to one compatible VM Offering will be specified, in the end, in case migration for such a component is not possible (e.g., stateful components for which no migration procedures have been defined in the Software Component Registry). An example of a Software Allocation record without adaptation rules could be: Software Component *SmartOCR* mapped on • VM with 4 cores,core 12 GB RAM, and 25 GB Disk by provider IBM SoftLayer (always eligible) (preselected) • VM T2.medium by provider Amazon AWS in Ireland
Actors	CloudSocket Broker
Use Case Objective	To allocate a Software Component invoked by the Workflow Model onto a set of compatible VM Offerings that will actually host the Software Component at run-time. Only one from the VM offerings is preselected. The other ones are used for adaptation/migration purposes.
Pre-Conditions	The Bundle exists in the Bundle Repository. The Bundle has been created by selecting a BPaaS Design Package containing an Executable Workflow Model (and not an abstract Workflow Model). The Workflow Model contains at least one service task mapping to a Software Component (which is registered in the Software Component Registry). The Cloud Infrastructure Registry contains at least one VM Offering compatible (according to both functional and non-functional aspects) with the selected Software

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	Component. All the metric definitions needed to specify the adaptation rules for a software component already exist.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the Software Component Allocator UI component (part of the Bundle Designer). 3. The Software Component Allocator UI shows the list of the Software Components mapping to service tasks in the Workflow Model of the Bundle, highlighting the Software Components not mapped to VM offerings. 4. The Broker selects a Software Components from the list and issues a command to open the Software Component selected. 5. The Software Component Allocator UI invokes a service on the Bundle Manager to retrieve the list of the VM Offerings which are compatible (according to functional and quality aspects) with the selected Software Component. 6. The Bundle Manager invokes the API of the Repository Manager to retrieve the list of the VM Offerings which are compatible with the selected Software Component. 7. The Software Component Allocator UI shows the list of all the VM Offerings matched with the selected Software Component. 8. The Broker can select one from the VM offerings matched as the one on which the deployment of the software component will take place. The Broker can also define adaptation rules to indicate which alternative compatible VM offering to select when the preselected one fails for some reason. 9. The Broker repeats Steps 4-8 as desired possibly until all Software Components associated to the BPaaS workflow tasks are allocated. 10. The Broker confirms the changes and issues the save command. 11. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes, thus updating the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured), then it is marked as Consistent (therefore eligible for publishing into the Marketplace).
Variations	11.a) The Broker aborts the edit and closes the Bundle Designer without saving. 11.b) The Broker saves the Bundle with a different name (“save as” command).
Post-Conditions	The Bundle is updated into the Bundle Repository.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

Diagrams	

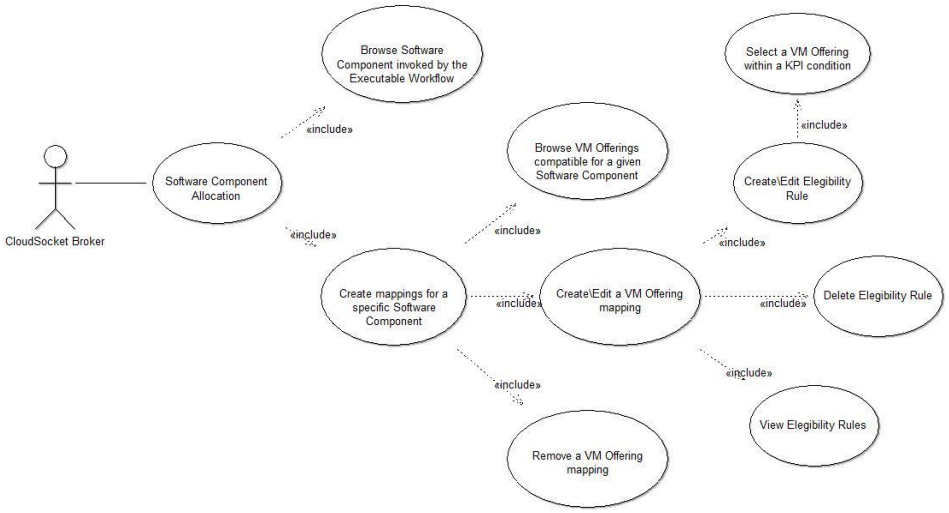


Figure 28 Use Case Diagram – AE-UC-3-Software Component Allocation

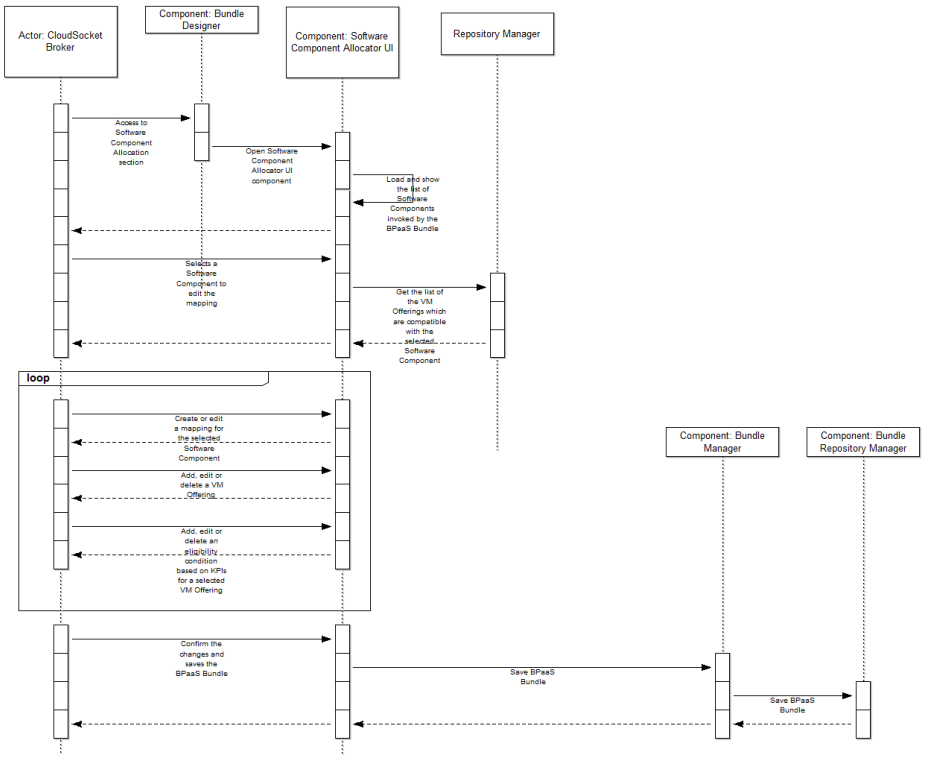


Figure 29 Sequence Diagram – AE-UC-3-Software Component Allocation

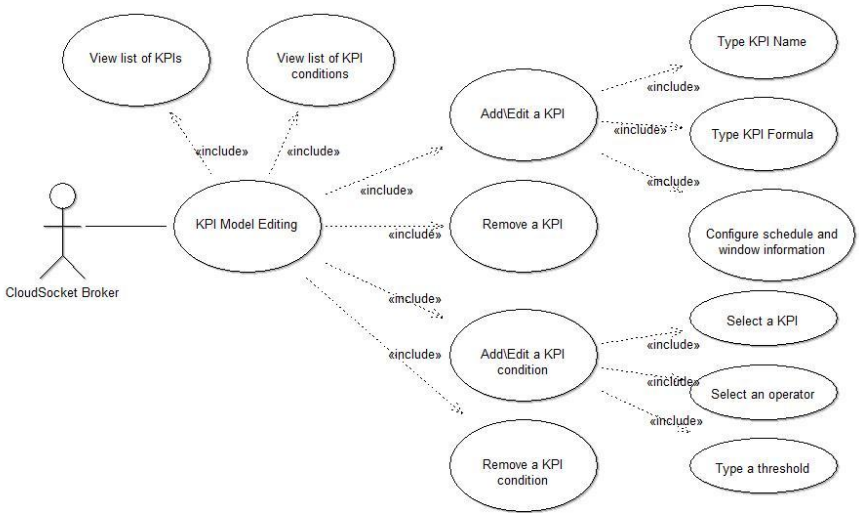
Table 12 BPaaS Allocation Environment – Use Case 3 – Software Component Allocation

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.4 KPI Model Editing

Use Case id	AE-UC-4-KPI Model Editing
Title & Description	A CloudSocket Broker defines a KPI for a given BPaaS Bundle. The Allocation Tool supports the Broker in exploring the KPIs defined in the Design Environment and provides features to translate them into technical KPIs that can be measured by providing all the information needed by the Monitoring Engine (e.g. formula, schedule, window and so on). It also allows creation of new KPIs, if they are not designed in the design phase, which can be used, for instance, to define allocation rules. The Broker opens a Bundle in the Bundle Designer, enters the KPI Model section and manages (creates, edits and deletes) the KPI definitions for the Bundle.
Actors	CloudSocket Broker
Use Case Objective	To define which KPI will have to be calculated and evaluated at run-time for a given Bundle. The KPI could also be used for evaluating the fulfillment of the Bundle's SLA (in case it maps to an SLO metric condition) and to assess the (metric) conditions of eligibility/adaptation rules defined in the Bundle's Atomic Service and Software Component allocation sections.
Pre-Conditions	The BPaaS Bundle exists in the Bundle Repository. The Bundle has been created by selecting a BPaaS Design Package containing an Executable Workflow Model (and not an abstract Workflow Model). The component metrics for the KPI's composite metric to be specified have already been defined.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the KPI modelling section of the Bundle Metadata Editor UI component (part of the Bundle Designer). 3. The KPI modelling section shows the list of KPIs and the list of KPIs conditions previously created; 4. The Broker issues a command to create or edit a KPI. 5. The Bundle Metadata Editor UI opens the KPI Editing Dialog. This dialog contains a text area to edit the KPI name, a text area to edit the KPI formula and a set of attributes to define the schedule and the window of the KPI: a. The Bundle Metadata Editor UI invokes a service on the Bundle Manager to retrieve the list of the raw metrics available for the BPaaS Bundle. b. The Bundle Manager invokes the API of the Repository Manager to retrieve the list of the domain-independent raw metrics available on the CloudSocket platform. c. The Bundle Metadata Editor UI invokes a service on the Bundle Manager to retrieve the list of the domain-dependent raw metrics declared in the Executable Workflow Model. 6. The Broker types the needed KPI information. Support is available for the insertion

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	in the formula of math operators, functions, and raw metrics. 7. The Broker confirms the changes in the edited KPI. 8. The Bundle Manager verifies that the formula of the KPI is syntactically correct. 9. The Broker issues a command to create or edit a KPI condition. 10. The Broker types the needed KPI condition information like the condition name, the selection of a KPI, the selection of a comparison operator and the typing of the threshold value. 11. The Broker confirms the changes in the edited KPI condition. 12. The Broker repeats Steps 4-11 as desired to complete the specification of the KPI Model of the Bundle. 13. The Broker confirms the changes and issues the save command. 14. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes thus updating the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured) then it is marked as Consistent (therefore eligible for publishing into the Marketplace).
Variations	4.a) The Broker has first to define a new raw metric in the Repository Manager, if needed, and then the actual composite metric of the KPI. 13.a) The Broker aborts the edit and closes the Bundle Designer without saving.
Post-Conditions	The BPaaS Bundle is updated into the Bundle Repository.
Diagrams	 **Figure 30 Use Case Diagram – AE-UC-4-KPI Model Editing**

Copyright © 2015 BOC and other members of the CloudSocket Consortium

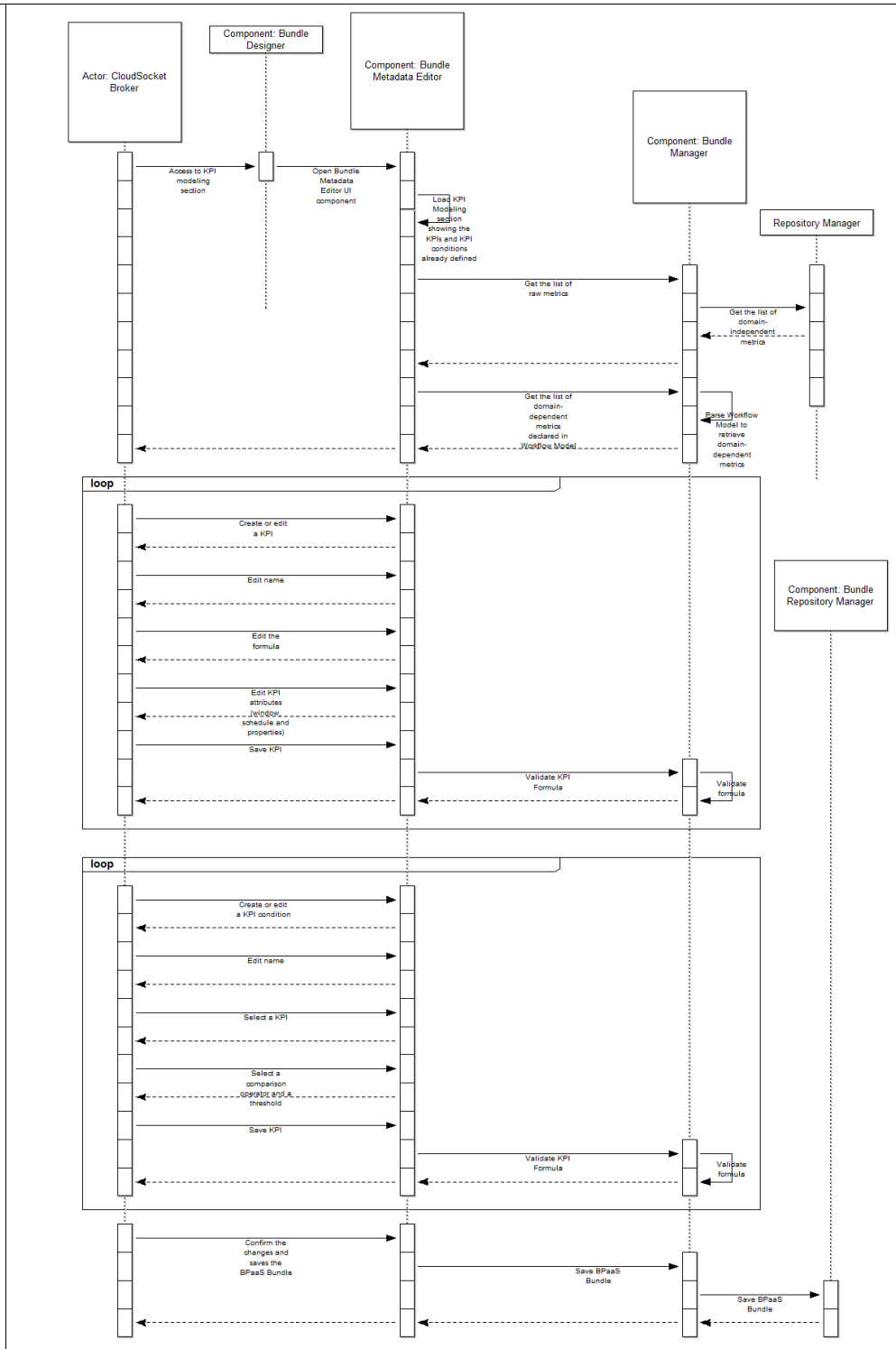


Figure 31 Sequence Diagram – AE-UC-4-KPI Model Editing

Table 13 BPaaS Allocation Environment – Use Case 4 – KPI Model Editing

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.5 SLA Model Editing

Use Case id	**AE-UC-5-SLA Model Editing**
Title & Description	A CloudSocket Broker defines the SLA for a given BPaaS Bundle. The Broker opens a Bundle in the Bundle Designer, enters the SLA Model section and edits the bundle's SLA by also managing (creating, editing and deleting) its SLO definitions.
Actors	CloudSocket Broker
Use Case Objective	To formally define the SLA guaranteed by the Broker to the potential CloudSocket Customer who desires to purchase and use the Bundle.
Pre-Conditions	The Bundle exists in the Bundle Repository. The Bundle has been created by selecting a BPaaS Design Package containing an Executable Workflow Model (and not an abstract Workflow Model). KPI conditions required to define the SLA Model have been already defined.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the SLA modelling section of the Bundle Metadata Editor UI component (part of the Bundle Designer). 3. The Bundle Metadata Editor UI shows the SLA's generic information as well as a list of the SLOs already defined, along with their respective KPI condition selected. 4. The Broker edits the generic information of the SLA as visualized in the SLA modelling section. 5. The Broker issues a command to create or edit a SLO. 6. The Bundle Metadata Editor UI opens the SLO Editing Dialog. This dialog contains a text area to edit the SLO description, a list to select a KPI condition, and a list of penalties applied if SLO violation occurs. 7. The Broker types all the required information. 8. The Broker confirms the changes in the edited SLO. 9. The Broker repeats the Steps 4-8 as desired to complete the definition of SLA Model of the Bundle. 10. The Broker confirms the changes and issues the save command. 11. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes, thus updating the Bundle into the Bundle Repository. If the Bundle is complete then it is marked as Consistent and therefore eligible for publishing into the Marketplace.
Variations	10.a) The Broker aborts the edit and closes the Bundle Designer without saving.
Post-Conditions	The Bundle is updated into the Bundle Repository.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

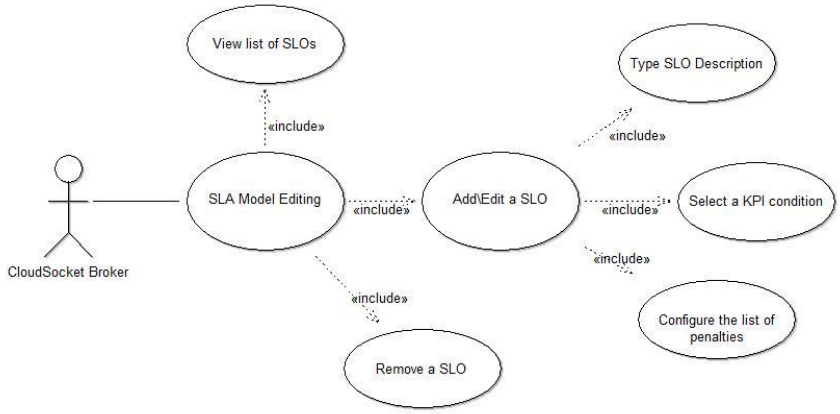
Diagrams	

Figure 32 Use Case Diagram – AE-UC-5-SLA Model Editing

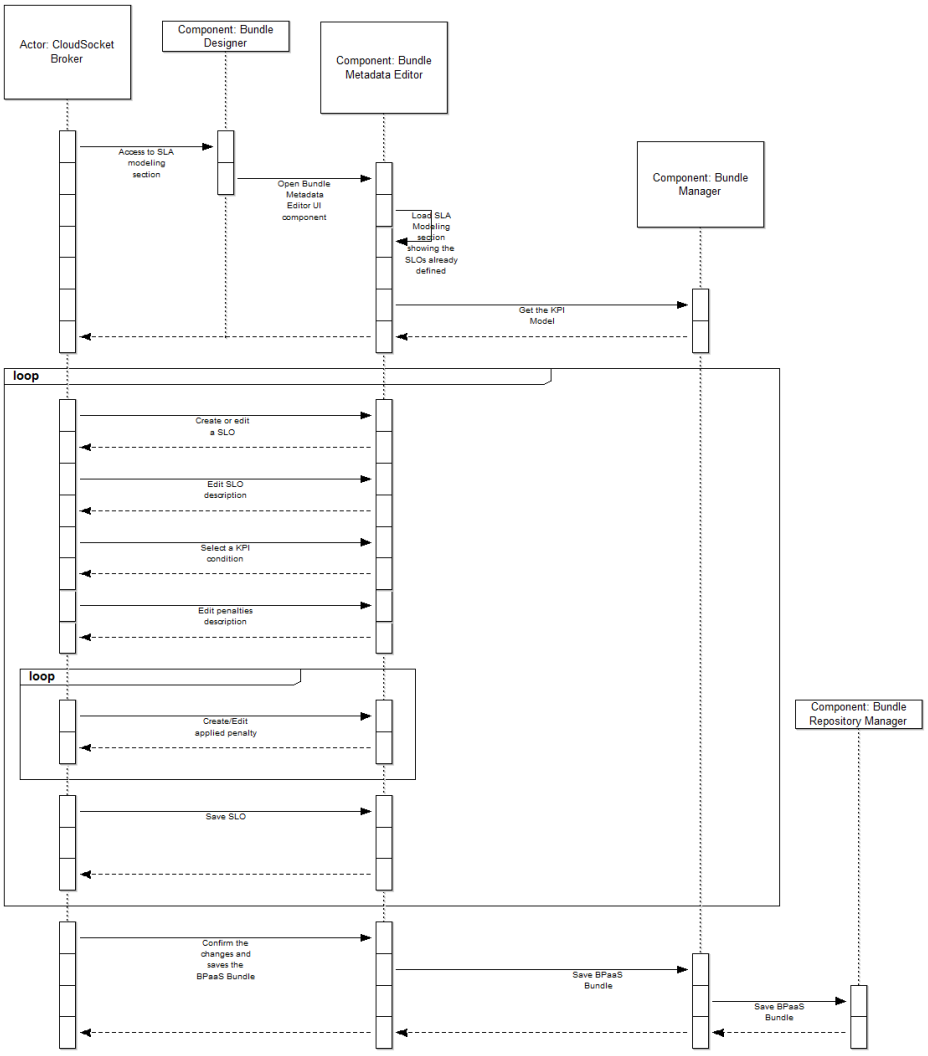


Figure 33 Sequence Diagram – AE-UC-5-SLA Model Editing

Table 14 BPaaS Allocation Environment – Use Case 5 – SLA Model Editing

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.6 Pricing Model Editing

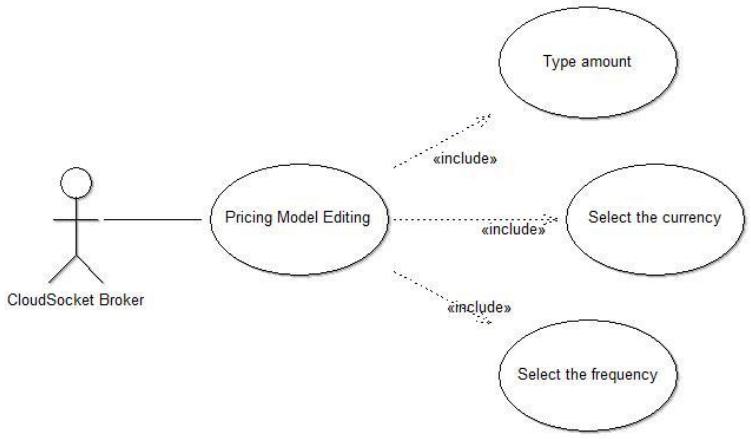
Use Case id	**AE-UC-6-Pricing Model Editing**
Title & Description	A CloudSocket Broker defines the Pricing Model for a given BPaaS Bundle. The Broker opens a Bundle in the Bundle Designer, enters the Pricing Model section and manages the Pricing model for the Bundle.
Actors	CloudSocket Broker
Use Case Objective	To formally define the Pricing Model of the bundle dictating the way the respective BPaaS usage is charged for a broker customer.
Pre-Conditions	The Bundle exists in the Bundle Repository.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the Pricing modelling section of the Bundle Metadata Editor UI component (part of the Bundle Designer). 3. The Bundle Metadata Editor UI shows generic pricing information like amount, currency and pricing frequency (e.g. daily, monthly and so on). 4. The Broker edits the generic information of the pricing model. 5. The Broker confirms the changes and issues the save command. 6. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes, thus updating the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured), then it is marked as Consistent (therefore eligible for publishing into the Marketplace).
Variations	5.a) The Broker aborts the edit and closes the Bundle Designer without saving.
Post-Conditions	The Bundle is updated into the Bundle Repository.
Diagrams	

Figure 34 Use Case Diagram – AE-UC-6-Pricing Model Editing

Copyright © 2015 BOC and other members of the CloudSocket Consortium

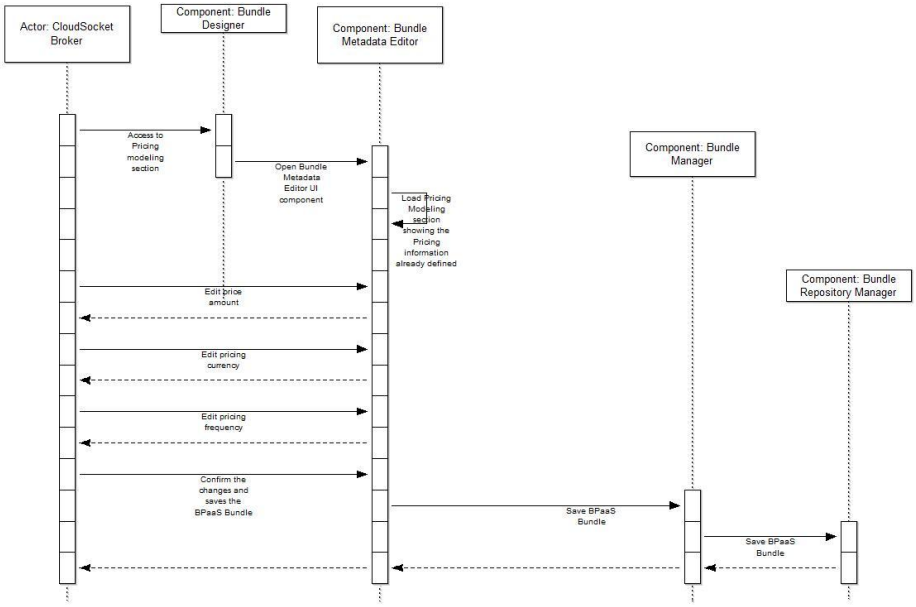


Figure 35 Sequence Diagram – AE-UC-6-Pricing Model Editing

Table 15 BPaaS Allocation Environment – Use Case 6 – Pricing Model Editing

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.7 Business Process Metadata Editing

Use Case id	AE-UC-7 Business Process Metadata Editing
Title & Description	A CloudSocket Broker defines the Business Process Metadata for a given BPaaS Bundle. The Broker opens a Bundle in the Bundle Designer; enters the Marketing metadata section and edits all the needed marketing-oriented information that could be useful to the customer (while browsing the Marketplace) in finding the Bundles best fitting his/her needs. The Marketing Metadata includes: 1. commercial name of the Bundle; 2. commercial image of the Bundle; 3. business-level functional description of the Bundle; 4. categories of the Workflow Model according to the APQC taxonomy; 5. list of tags; This information is intended to be used by the marketplace User Interface to present it to the CloudSocket Customer along with other relevant pieces of information already contained in the other Bundle sections.
Actors	CloudSocket Broker
Use Case Objective	To define all the additional marketing-oriented information needed by the Marketplace to support a Business Process User in finding the BPaaS Bundle best fitting his/her needs.
Pre-Conditions	The Bundle exists in the Bundle Repository.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker enters the Business Process Metadata section of the Bundle Metadata Editor UI component (part of the Bundle Designer). 3. The Bundle Metadata Editor UI opens the Business Process Meta Metadata Editing Dialog. This dialog contains text areas to edit commercial name, commercial image, business-level functional description, list of tags and an editor containing a control to select the APQC categories best describing the Business Process Model included in the Bundle. 4. The Broker fills the Business Process Metadata Editing Dialog. 5. The Broker confirms the changes and issues the save command. 6. The Bundle Manager receives the save command and delegates the Bundle Repository Manager to save the changes, thus updating the Bundle into the Bundle Repository. If the Bundle is complete (all sections properly configured), then it is marked as Consistent (therefore eligible for publishing into the Marketplace).
Variations	5.a) The Broker aborts the edit and closes the Bundle Designer without saving.
Post-Conditions	The Bundle is updated into the Bundle Repository.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

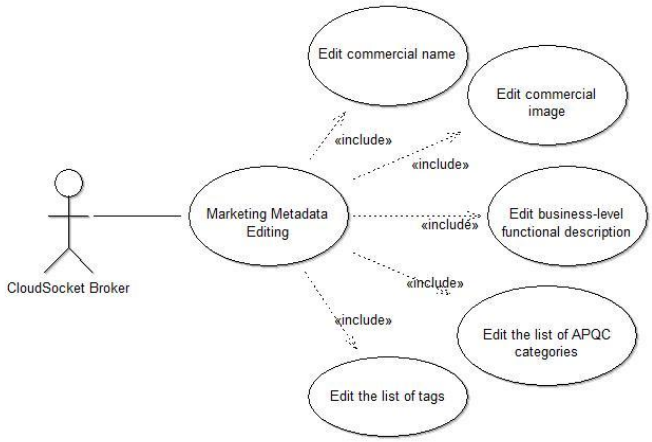
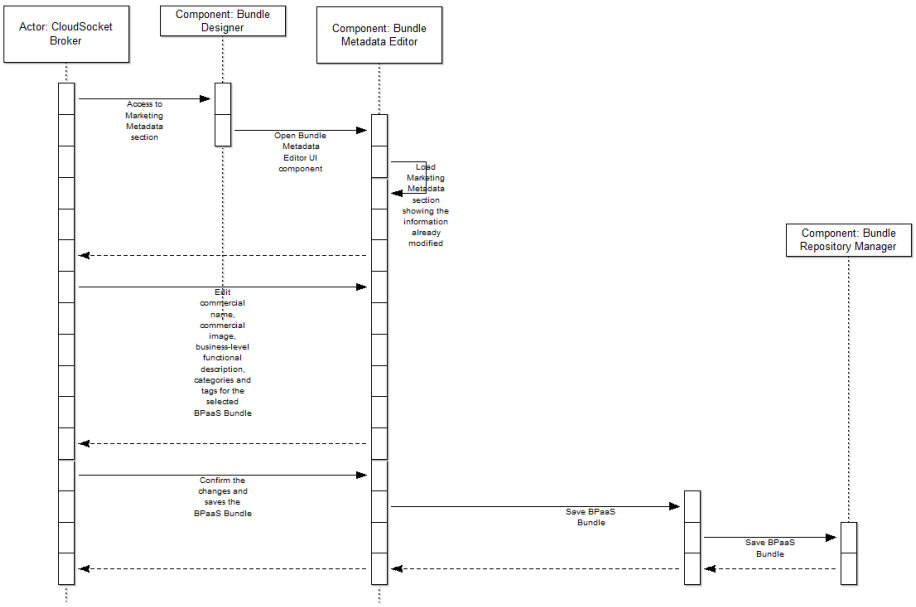
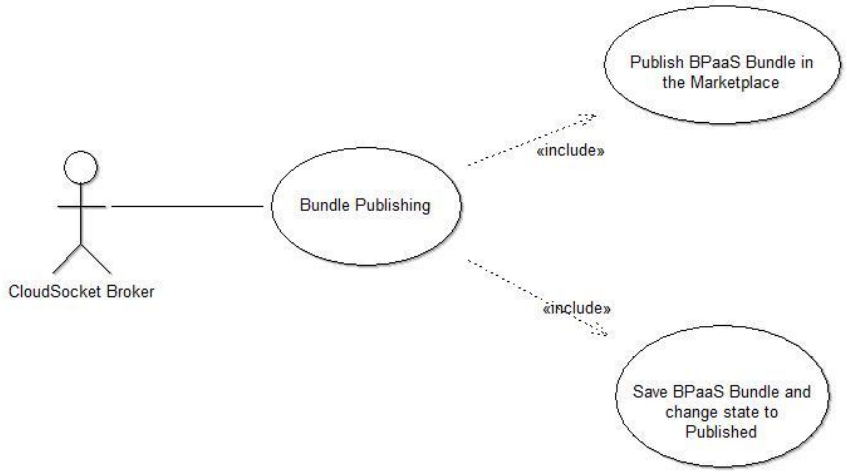
Diagrams	 **Figure 36 Use Case Diagram – AE-UC-7 Business Process Metadata Editing**  **Figure 37 Sequence Diagram – AE-UC-7 Business Process Metadata Editing**

Table 16 BPaaS Allocation Environment – Use Case 7 – Business Process Metadata Editing

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.2.8 BPaaS Bundle Publishing in the Marketplace

Use Case id	**AE-UC-8-BPaaS Bundle Publishing in the Marketplace**
Title & Description	A CloudSocket Broker publishes a BPaaS Bundle in the Marketplace. The Broker publishes a Consistent BPaaS Bundle in the Marketplace so that customers can purchase, deploy and use it.
Actors	CloudSocket Broker
Use Case Objective	To publish a BPaaS Bundle into the Marketplace and enable customers to view and eventually buy it (in the case of an executable BPaaS Bundle).
Pre-Conditions	The Bundle exists in the Bundle Repository, and it is in the Consistent state.
Process Dialog	1. The Broker selects a BPaaS Bundle from the Bundle Browser and opens it into the Bundle Designer. 2. The Broker issues the command to publish the BPaaS Bundle in the Marketplace. 3. The Bundle Designer invokes the Bundle Manager to publish the bundle. 4. The Bundle Manager delegates the publication of the Bundle to the Bundle Publisher. 5. The Bundle Publisher invokes the Marketplace management API exposed by the Marketplace Environment. 6. The Bundle Manager updates the BPaaS Bundle in the Bundle Repository in order to mark it as published and saves its date of publication.
Variations	6.a) In case a Bundle with the same name already exists in the Marketplace, appropriate conflict resolution must be taken, asking the broker to change the name of the current BpaaS Bundle.
Post-Conditions	The BPaaS Bundle is published in the Marketplace and the customer can find and buy it (in case of an executable BPaaS Bundle) using the Marketplace. The BPaaS Bundle is marked as Published into the Bundle Repository.
Diagrams	

Copyright © 2015 BOC and other members of the CloudSocket Consortium

Figure 38 Use Case Diagram – AE-UC-8-BPaaS Bundle Publishing in the Marketplace

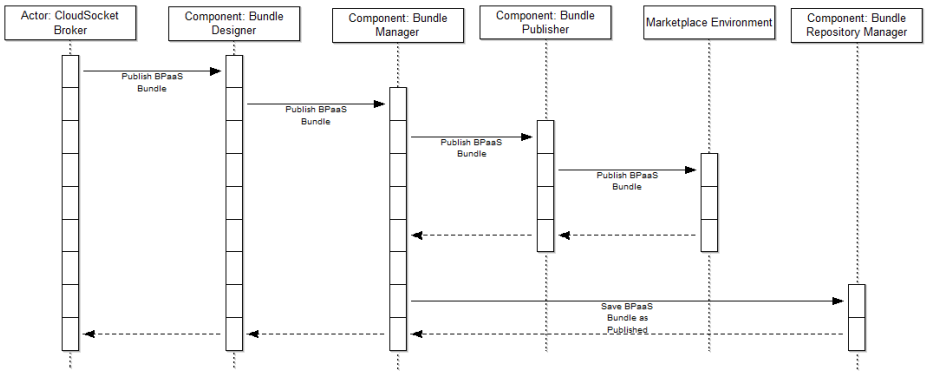


Figure 39 Sequence Diagram – AE-UC-8-BPaaS Bundle Publishing in the Marketplace

Table 17 BPaaS Allocation Environment – Use Case 8 – BPaaS Bundle Publishing in the Marketplace

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.3 Components

The BPaaS Allocation Environment provides the following high-level features: i) creation of new BPaaS Bundle; ii) search and editing of existing BPaaS bundles; iii) selection of BPaaS Design Package; iv) configuration of marketing metadata; v) workflow services allocation (for both abstract atomic services and software components); vi) definition of KPI and SLA Model; vii) publication into the marketplace.

In order to support aforementioned functionalities, the BPaaS Allocation Environment is composed by one main component called Allocation Tool.

6.3.1 User-interface layer

The User Interface Layer contains the graphical interface components used by the CloudSocket Broker to create, edit and manage his/her own BPaaS Bundles.

There are three main UI components:

1. Bundle Instantiator
2. Bundle Designer
3. Bundle Browser

The Bundle Instantiator and Bundle Browser are used to choose the input for the Bundle Designer which can accept a new bundle or an existing one, previously created and saved into the Bundle Repository.

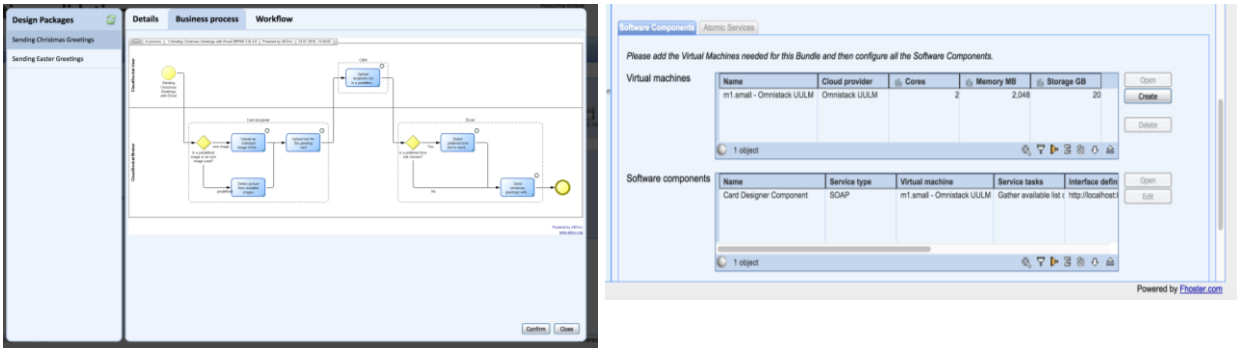


Figure 40 BPaaS Allocation Tool – User Interface Screenshot

Figure 40 indicates two screenshots of the Allocation Tool. The first one indicates that the Allocation Tool starts from the input of the BPaaS Design Environment, by choosing a BPaaS Design Package and viewing additional business relevant information. The second one shows the Allocation section containing different sub-sections for Software Components and Atomic Services Allocation.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.3.1.1 Bundle Instantiator

The Bundle Instantiator is the UI component allowing a Broker to create a new draft BPaaS Bundle. Since a Bundle is conceptually a commercial "packaging" of an executable Workflow Model, the creation of a draft Bundle necessarily starts with the selection of a workflow model from a BPaaS Design Package retrieved from the BPaaS Design Environment.

The Bundle Instantiator, therefore, interacts with the BPaaS Design Environment in order to retrieve the list of the BPaaS Design Packages along with all the available meta-data that might support the Broker in searching and identifying (if any) the BPaaS Design Packages best fitting his/her needs (actually the needs of the CloudSocket Customer(s) the Broker is working for or desires to attract).

The BPaaS Design Packages are retrieved using the API provided by the BPaaS Design Environment. The Bundle Instantiator provides an user-friendly interface allowing the Broker to apply multiple filters based on the meta-data associated with each Workflow Model, and finally select the preferred one.

Once the Broker has selected a BPaaS Design Package, the Bundle Instantiator creates in volatile memory a data structure representing a new draft BPaaS Bundle based on that BPaaS Design Package, and it passes such data structure to the Bundle Designer, which allows the Broker to immediately start the configuration of the Bundle.

At any time, the Broker can either close the Bundle Designer or discard the changes to abort the Bundle instantiation, or save the Bundle making it persistent in the Bundle Repository.

Tasks performed by the Bundle Instantiator:

- Browse the BPaaS Design Packages from the BPaaS Design Environment.
- Instantiate a new BPaaS Bundle selecting one BPaaS Design Package.

Main Interfaces of the Bundle Instantiator:

- Input: None. BPaaS Design Packages are retrieved from the BPaaS Design Environment upon user request - in the end, one BPaaS Design Package is selected to be instantiated.
- Output: a draft BPaaS Bundle, containing only the selected BPaaS Design Package which can be sent to the Bundle Designer.

6.3.1.2 Bundle Designer

The Bundle Designer is the main UI component of the Allocation Tool. It provides all the functionalities allowing a CloudSocket Broker to configure (fill or edit) the sections of a BPaaS Bundle and move it from the initial Draft (incomplete) state to the Consistent (ready to be published) state.

The Bundle Designer includes three main sub-components:

1. Atomic Service Allocator UI: it's the UI component allowing a Broker to allocate each abstract Atomic Service invoked by the Executable Workflow Model in the BPaaS Bundle onto a set of compatible concrete Atomic Services out of which only one will be invoked at run-time, along with the rules for enabling their selection eligibility. Concrete Atomic Services are selected from the Service Registry of the Repository Manager.
2. Software Component Allocator UI: it's the UI component allowing a Broker to allocate each Software Component invoked by the Executable Workflow Model in the bundle onto a set of compatible VM Offerings out of which only one is selected to host this component. VM Offerings are selected from the Cloud Provider Registry in the Repository Manager. The Software Component Allocator UI also allows to set adaptation

Copyright © 2015 BOC and other members of the CloudSocket Consortium

rules for each software component in terms of the underlying resources to be exploited (e.g., mapping to migrate to another VM or scaling-out to a new VM instance in case of increased load).

3. Bundle Metadata Editor: it's the UI component allowing a Broker to configure all the information needed to sell the Bundle: a KPI Model, a SLA (based on conditions over existing or newly defined metrics), a Pricing Model (potentially based on existing metrics and attributes/variables), and Business Process Metadata required to publish the bundle into the Marketplace and make sure that customers can actually find it and understand what it does at a very high level (business executive) of abstraction.

Tasks performed by the Bundle Designer are the following:

- Allocate all the abstract Atomic Services invoked by the Workflow Model of the Bundle, and define the eligibility rules of the selected concrete Atomic Services.
- Allocate all the Software Components invoked by the by the Workflow Model of the Bundle, and define the respective adaptation rules.
- Edit the Bundle Metadata (KPI definitions, SLA, Pricing Model, Marketing metadata)
- Save the Bundle into the Bundle Repository and contextually change its state from Draft to Consistent or vice versa, if required.
- Publish the bundle into the Marketplace and contextually change its state from Consistent to Published.

Main Interfaces of the Bundle Designer are:

- Input: a draft Bundle is received from the Bundle Instantiator or an existing one has been selected using the Bundle Browser.
- Input: a list of concrete Atomic Services is received from the Bundle Manager when the Broker uses the Atomic Services Allocation UI component.
- Input: a list of VM Offerings is received from the Bundle Manager when the Broker uses the Software Component Allocation UI component.
- Output: the edited Bundle to be sent to Bundle Manager when the Broker issues the save or the publish command.

6.3.1.3 Bundle Browser

The Bundle Browser is the UI component allowing a Broker to explore and manage the Bundle Repository. The component shows to the Broker a list of all the Bundles he/she has created along with their status (Draft, Consistent or Published) and the date-time of their last update.

The Bundle Browser also features controls to select a Bundle and to either open it in the Bundle Designer for viewing/editing or to delete it from the Bundle Repository.

When a published Bundle is removed from the Marketplace, the same Bundle in the Bundle Repository changes its state from Published to Consistent, so that it can be edited again.

Tasks performed by the Bundle Browser:

- Browse the BPaaS Bundles related to the authenticated broker, which have been stored into the Bundle Repository.
- Enable the editing of a BPaaS Bundle.
- Duplicate an existing BPaaS Bundle.
- Delete an existing BPaaS Bundle

Copyright © 2015 BOC and other members of the CloudSocket Consortium

Main Interfaces of the Bundle Browser:

- Input: a list of Bundle descriptors is retrieved from the Bundle Repository Manager, in order to allow the Broker to browse the content of the Bundle Repository.
- Input: a Bundle is retrieved from the Bundle Repository Manager when the Broker selects the Bundle and issue the open/edit command.
- Output: a Bundle identifier is sent to the Bundle Repository Manager when the Broker selects the Bundle and issues the delete command.
- Output: a Bundle is sent to the Bundle Designer to open/edit the Bundle.

6.3.2 Business-logic layer

The Business-logic layer provides functionalities to the User-interface layer. It includes the following components:

1. Bundle Repository Manager
2. Bundle Manager
3. Bundle Publisher

6.3.2.1 Bundle Repository Manager

The Bundle Repository Manager acts as a management interface to the Bundle Repository for all the other components of the Allocation Tool. It provides all the functionalities to list and search the contents of the Bundle Repository, to load a bundle in memory, to save a bundle in the Repository and to delete a bundle from the Repository.

It is responsibility of the Bundle Repository Manager to let each Broker access only its own bundles as well as to allow only legitimate state transitions and actions to be executed over the bundles of a certain broker.

Tasks performed by the Bundle Repository Manager:

- Provide a list of descriptors of the bundles in the Bundles Repository.
- Search for a bundle in the Bundle Repository.
- Load a bundle from the Bundle Repository into the memory.
- Save a bundle from the memory into the Bundle Repository.
- Delete a bundle from the Bundle Repository.
- Change the state of a bundle.

Main Interfaces of the Bundle Repository Manager:

- Input: receives from the Bundle Browser the identifier of a bundle to be loaded in memory (loaded from the Bundle Repository)
- Input: receives from the Bundle Browser the identifier of a bundle to be deleted from the Bundle Repository.
- Input: receives from the Bundle Manager a bundle in memory to be saved into the Bundle Repository.
- Input: receives from the Bundle Browser or Manager a request for changing the state of a bundle
- Input: receives from the Bundle Browser a search condition for searching and identify one or more bundles stored in the Bundle Repository
- Output: sends (to the Bundle Browser) a list of descriptors of all bundles or those matching the search conditions/criteria provided in the Bundle Repository.
- Output: sends (to the Bundle Browser) a bundle to be loaded in memory.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.3.2.2 Bundle Manager

The Bundle Manager implements all the business logic required by the Bundle Designer UI and its sub-components.

It manages the BPaaS Bundle in memory, making sure that all the sections of the Bundle's data structure are properly aligned at any time (i.e., no invalid cross-references, overall data structure consistency) and contain only allowed values.

The Bundle Manager also detects when a Bundle has been completely configured (i.e., configured with at least the minimal but sufficient information required to be published) and performs the transition from the Draft to the Consistent state or vice-versa (depending on the previous and new level of configuration).

Tasks performed by the Bundle Manager:

This server-side component manages in memory the Bundle currently edited by the Broker. The Bundle Manager interacts with the following components:

- the registries (Software Component Registry, Cloud Provider Registry, VM Offering Registry, Abstract Atomic Service Registry, Concrete Atomic Service Registry and Metric Registry). **Note**: the registries are not accessed directly by the Bundle Designer because the raw lists provided by each registry must be pre-filtered in order to isolate only the registry items which are compatible with the bundle in memory;
- the Bundle Repository Manager, which is responsible for loading/saving the bundles from/to the Bundle Repository;
- the Bundle Publisher, which is responsible to interact with the Marketplace management API in order to publish a bundle.

Main Interfaces of the Bundle Manager

- Input: receives from the Bundle Designer a Bundle in memory when the Broker issues the save command (the Bundle is forwarded to the Bundle Repository Manager).
- Input: receives from the Atomic Service Registry a list of Atomic Service descriptors.
- Input: receives from the Cloud Provider Registry a list of VM Offering descriptors
- Output: sends (to the Bundle Browser) a list of descriptors of the Bundles in the Bundle Repository.
- Output: sends (to the Bundle Repository Manager) the Bundle in memory to be saved.

6.3.2.3 Bundle Publisher

The Bundle Publisher is responsible for publishing the BPaaS Bundle in the Marketplace. The Bundle Manager forwards to the Bundle Publisher the Bundle currently in memory and delegates the latter component to perform all the interactions with the Marketplace management API required to get the Bundle published.

If the publishing operation succeeds, the Bundle Manager changes the state of the Bundle from Consistent to Published and sends it to the Bundle Repository Manager in order to make the state transition persistent.

The Bundle Publisher allows also the update of a Bundle already published in the Marketplace with the same flow followed in case of the first publication.

Tasks performed by the Bundle Publisher: this server-side component is responsible to publish a Bundle into the Marketplace.

Main Interfaces of the Bundle Publisher:

Copyright © 2015 BOC and other members of the CloudSocket Consortium

- Input: receives from the Bundle Manager a Bundle in memory
- Output: sends to the Marketplace all the input parameters required by the Marketplace management API for the bundle publishing.
- Output: state change of bundle to be persisted

6.3.3 Persistency-management layer

The Persistency-management Layer contains the Bundle Repository where the BPaaS Bundles are stored. The Bundle Repository is multi-tenant so that each CloudSocket Broker can browse and edit only the BPaaS Bundles owned by him/her.

6.3.4 Component Diagram

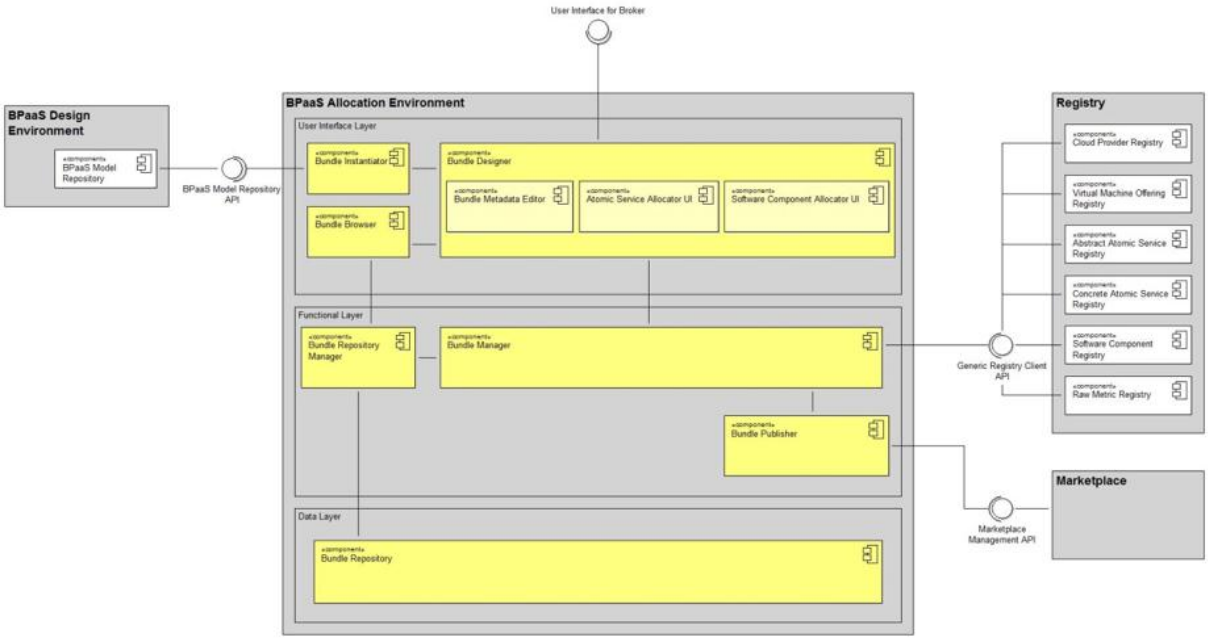


Figure 41 Allocation Tool – Component Diagram

6.3.5 Research Prototypes

In the context of Deliverable D3.3 “BPaaS Allocation and Execution Environment Blueprints” the following research prototypes are introduced as optional components.

6.3.5.1 Smart Service Discovery and Composition Tools

The Smart Service Discovery and Composition Tools (D3.3 2016, sections 3.2, 3.3) can increase the automation level in the BPaaS Allocation Environment by enabling the automatic matchmaking and selection of both SaaS and IaaS services in a conjunctive manner by also taking into account the broker’s functional and non-functional constraints. These tools comprise: (a) a semantic functional and non-functional matchmaker which can be exploited in a standalone manner in case that the broker requires to discover the services that functionally and non-functionally match the BPaaS workflow task. The services discovered can be ranked such that the broker can select the one that best suits his/her functional requirements. The same matchmaker can also be used for the discovery of IaaS services that can support the internal service components of certain BPaaS workflow tasks; (b) a concurrent SaaS and IaaS selection tool which is able to exploit the matchmaker in order to find those services per BPaaS workflow task that can satisfy the local functional and non-functional broker requirements as well as to

Copyright © 2015 BOC and other members of the CloudSocket Consortium

then produce and solve a constraint optimisation problem which attempts to discover in conjunction the best external SaaS service from the candidate ones for each BPaaS workflow task as well as the best IaaS services from the candidate ones to support the internal service components for certain BPaaS workflow tasks by of course considering the broker's global non-functional requirements. The latter tools draw the respective content, actually mapping to cloud service advertisements, supporting their functionality from the Registry.

While D3.3 partly covers the architecture of the first tool, the semantic matchmaker, a combined architecture is now reported on this deliverable which can be seen in Figure 42. This architecture comprises three main components: (a) a front-end which assists the broker in graphically specifying the BPaaS allocation information in terms of a BPaaS bundle. This component can be surely replaced by the actual BPaaS Allocation Environment implementation which includes a quite sophisticated front-end; (b) the semantic matchmaker which is called as Unified (meaning covering both functional and non-functional service matchmaking) Service Discovery Tool; (c) the global IaaS & SaaS service selection tool which is called Service Concretisation Tool.

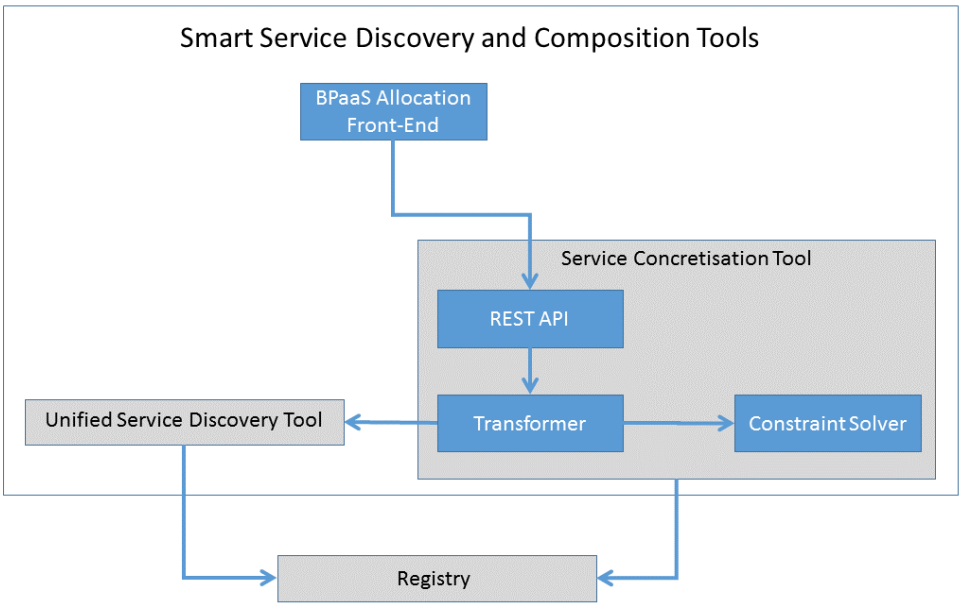


Figure 42: The architecture of the Smart Service Discovery and Composition Tools

For the latter tool, the internal architecture is exposed comprising three main components: (a) the REST-API which exposes the service concretisation functionality offered; (b) the Transformer which is responsible for obtaining the appropriate input from various sources (e.g., cloud service offerings, broker requirements), by also calling the appropriate components (e.g., Registry for the offerings, the Unified Service Discovery Tool for service candidates per BPaaS workflow task), as well as for generating the actual constraint optimisation problem to solve; (c) the Constraint Solver which solves the problem and reports back the solution discovered.

As D3.3 does not yet cover also the complete architecture of the Unified Service Discovery Tool, this architecture is depicted in Figure 43. It includes seven main components: (a) the REST-API which exposes the functionality for registering and matchmaking functional and non-functional specifications of services; (b) the Specification Processor which takes care of the syntactic, semantic and constraint-based validation of the service specifications issued via the REST-API as well as the alignment of these specifications according to a basic set of non-functional terms; (c) the Compositor orchestrates the functional and non-functional registration and matchmaking of services in a transactional manner by ensuring that when an aspect-specific registration fails, then the system roll-backs to its previous state; (d) the Functional Matchmaking Tools comprising mainly one component, the Functional Matchmaker; (e) the Non-Functional Matchmaking tool which includes components for

Copyright © 2015 BOC and other members of the CloudSocket Consortium

performing either ontology subsumption-based or constraint-based matchmaking of services (see also additional details in D3.3); (f) the Combined Registry which ensures that suitable URI entries exist per service covering both functional and non-functional entries by also assisting in achieving transactionality during service registration; (g) the Semantic Repository which is a semantic version of the Registry (meaning that respective information from this registry is drawn, semantically transformed and then stored into the Semantic Repository) covering mainly the semantic specification of services.

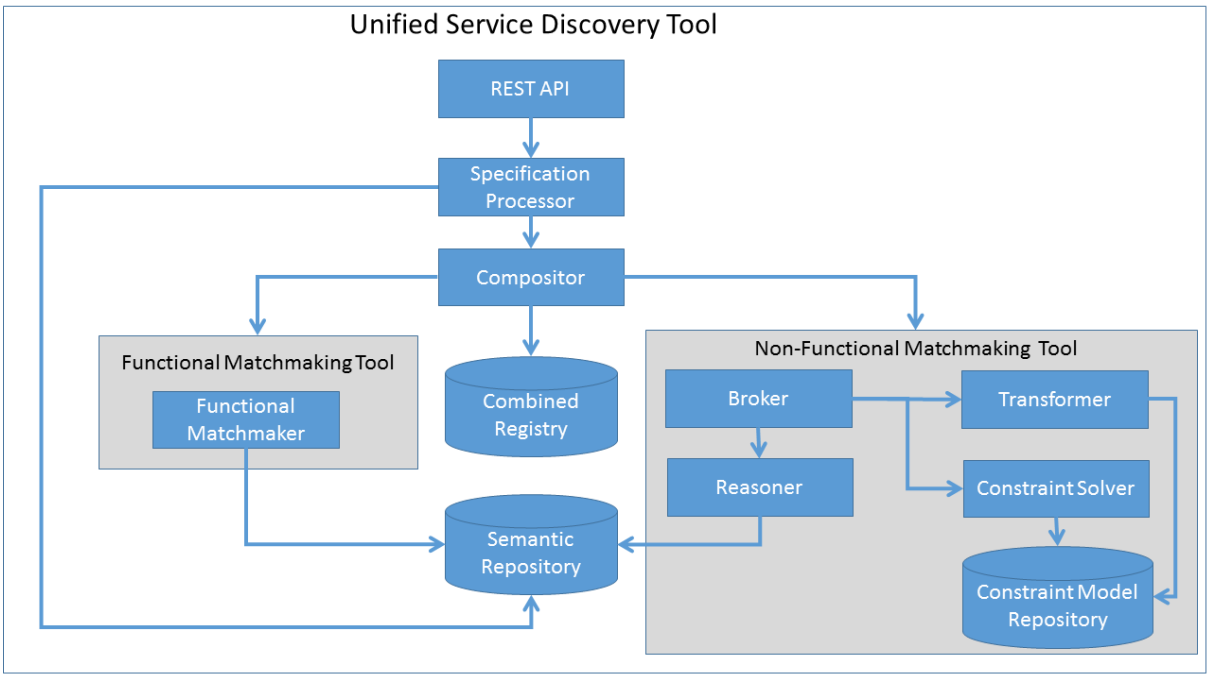


Figure 43: The architecture of the Unified Service Discovery Tool

6.3.5.2 DMN-to-CAMEL-Mapper

The DMN-to-CAMEL-Mapper[1] reduces the technical complexity of the software component allocation (cf. "AE-UC-4-Software Component Allocation") by mapping high-level business requirements to the low-level technical description. As business experts still require technical assistance for consuming cloud services and allocating the software components, the DMN-to-CAMEL-Mapper aims to create a way to semi-automatically handle the software component allocation and configuration based on high-level parameters. This approach enables the modelling of cloud applications by using non-technical business values, which will be mapped to a technical description, namely CAMEL, by using the *Business Knowledge Model (BKM)* in combination with *Decision Tables (DT)* of the *Decision Model and Notation (DMN)* standard (OMG 2015).

Figure 44 depicts a high-level view on the architecture of the DMN-to-CAMEL-Mapper. The component is accessible via a REST API and requires a set of business requirements. These requirements will be mapped to predefined and reusable DMN tables and the DMN Execution Engine processes the respective DMN tables by transforming the business requirements into technical CAMEL fragments. As a CAMEL model comprises of multiple sub-models, the CAMEL Composer assembles the CAMEL fragments into one CAMEL model, which is returned as result.

[1] See D3.3 BPaaS Allocation and Execution Environment Blueprints, chapter 3.4

Copyright © 2015 BOC and other members of the CloudSocket Consortium

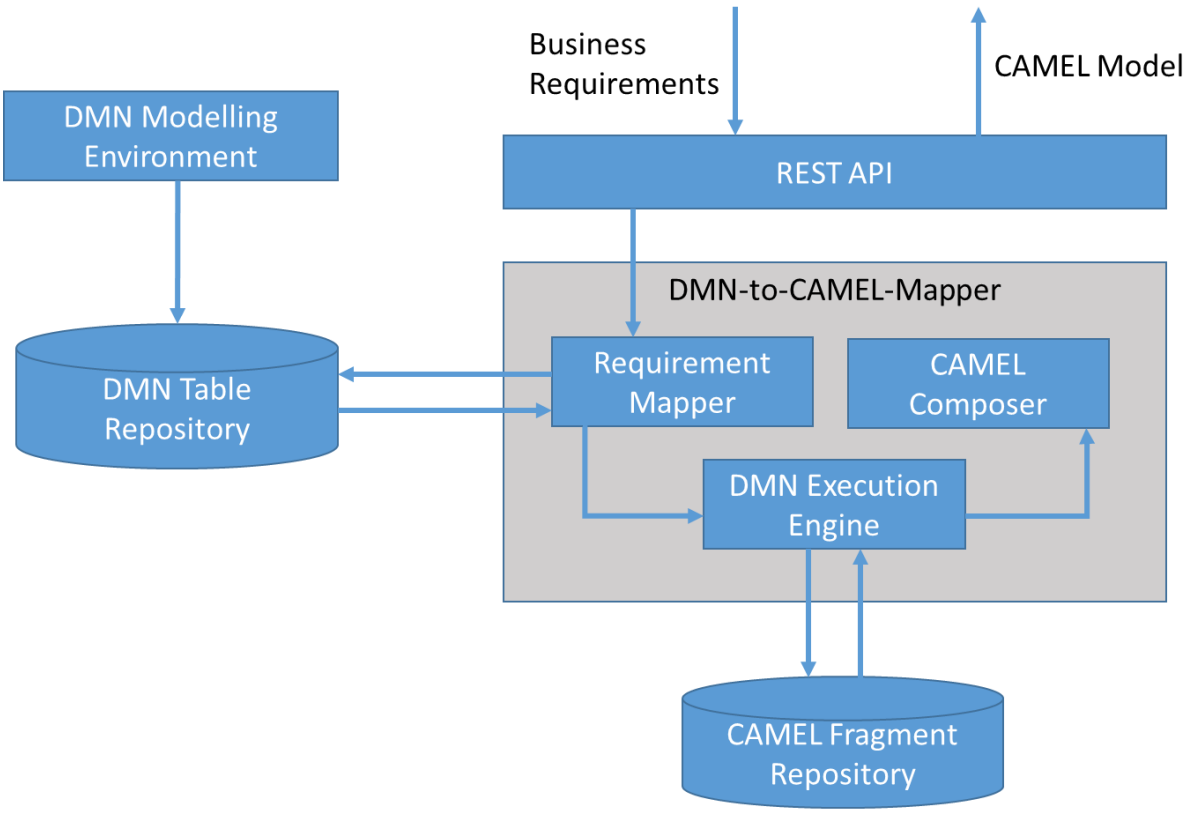


Figure 44 - DMN-to-CAMEL-Mapper Architecture

6.4 Roles

The CloudSocket Broker is an organization whose business involves creating BPaaS Bundles and selling them to CloudSocket Customer via the Marketplace. The CloudSocket Broker (actually the users of this organisation or a technical consultant hired) creates the bundles starting from a BPaaS Design Package selected from the BPaaS Design Environment.

If a CloudSocket Broker does not find a convenient BPaaS Design Package to create a BPaaS Bundle, then he/she could involve one or more consultants and delegate to them the creation of a new BPaaS Design Package by means of the BPaaS Design Environment.

The CloudSocket Broker gets paid by the BPaaS Customer who purchases the BPaaS Bundles that have been published in the Marketplace. It is responsibility of the CloudSocket Broker to make sure that each BPaaS Bundle published in the Marketplace has a Pricing Model covering the costs for executing the bundle, and a SLA to be met by the bundle, taking into account pricing and respective SLAs of all external resources (concrete Atomic Services and VM offerings) used by the bundle.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

6.5 Data Interface

The input of the Allocation Environment is a BPaaS Design Package described in paragraph 4.2.2.1 and 5.6.

The output of the BPaaS Allocation Environment is a BPaaS Bundle, represented as a JSON file containing a section for each of the following information (if it is executable):

- Business Process Model
- Executable Workflow Model
- Atomic Service Allocation
- Software Component Allocation
- KPI Model
- SLA
- Pricing Model
- Marketing Metadata

In case of abstract BPaaS Bundle, the output contains the following information:

- Business Process Model
- Abstract Workflow Model
- Pricing Model
- Marketing Metadata

Allocation-related information and metrics are represented according to the CAMEL (CAMEL 2015) language. The eligibility/adaptation rules are expressed in the SRL (Scalability Rules Language) (SRL 2015) language which is also part of CAMEL. These two languages have been defined in the context of the PaaSage project (PaaSage 2015). They have been extended in order to express extra information in different aspects that is needed for supporting the deployment, allocation, and adaptation of BPaaS.

With respect to the research prototpyes, the SLA can conform to a combination of WS-Agreement (WS-Agreement 2015) and OWL-Q (OWL-Q 2015), where OWL-Q is used for semantically annotating (and thus providing indirectly their formal specification) the metrics referenced in the conditions of SLOs in the WS-Agreement specifications. Alternatively, OWL-Q can be exploited in a standalone manner as it is able to specify SLAs. In this latter case, if existing SLA Management components are in place to be used in the Execution Environment which operate over WS-Agreement, we could have transformation code which is able to transform OWL-Q SLA specifications to WS-Agreement ones.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7 BPAAS EXECUTION ENVIRONMENT

7.1 Introduction

The execution environment consists basically of:

- the deployment/cloud provider engine of BPaaS
- the execution engine of BPaaSs
- the monitoring engine covering the different levels (business, workflow, service, infrastructure)
- the SLA engine to assess the conformance to agreements
- the adaptation engine to adapt the BPaaS to still fulfil the requirements posed to it
- the associated Web-GUI environments to cover the different components

Besides these execution and follow-up mechanisms, the execution environment has:

- The adaptation mechanisms to maintain the different levels of the quality for BPaaSs.

The environment is responsible to manage, monitor and adapt the execution of the BPaaS bundles generated during the allocation phase, which have been published via the Marketplace. Hence, the main functional capability is to guarantee the execution and the suitable behaviour of the deployed BPaaSs. To this end, the environment is responsible to deploy and configure all components required to execute the bundles, which comprise the workflow, SLA, adaptation rules, and details of the third-party services involved. When a BPaaS workflow bundle is deployed, the environment will allow to manage an end-user's workflow instances and to visualize the conformance levels to associated agreements and the respective monitoring data. Besides, based on the monitoring data, the violations incurred as well as the BPaaS bundle adaptation rules, the environment will be able to adapt the BPaaS instances to maintain the promised service level through executing particular adaptation actions, including component scaling, component/workflow migration and service substitution, possibly across different levels.

The main roles involved in the execution phase and thus to this environment are: i) the **CloudSocket Operator** who is providing and maintaining the environment to manage BPaaSs; ii) the **CloudSocket Broker** who is responsible for designing and publishing BPaaS bundles in the CloudSocket; This broker will also be able to check the SLA agreement status as well as find root causes of problematic situations in order to adapt a BPaaS and of course it is responsible for designing suitable adaptation rules in order to address such problematic situations; iii) **CloudSocket Customer**, a user of an end-user organisation which is assigned with the task to find, purchase and manage the different BPaaS bundles to be used by his/her organisation, iv) **Knowledge Worker** which represents a certain employee of an end-user organisation that is mainly involved in the interaction with the BPaaS workflow instance, when a manual task is assigned to him.

As there is also a need to compute and assess KPIs in the context of checking overall BPaaS goals as well as for optimizing BPaaS, the monitoring and contextual data produced by the BPaaS Execution Environment should be transferred on demand to the BPaaS Evaluation Environment which covers the next phase in the BPaaS life-cycle. Such data, apart from KPI assessment, can be used to discover interesting patterns or optimisation suggestions in the context of one or more BPaaS.

The following components can be distinguished to fulfil the aforementioned functional capabilities of the BPaaS Execution Environment: i) graphical interface components which can be exploited in order to view the status and manage BPaaS workflow bundles as well as inspect the conformance level of service level agreements and visualize respective monitoring information, ii) a Workflow Engine, to manage the execution of deployed BPaaS

Copyright © 2015 BOC and other members of the CloudSocket Consortium

workflows, iii) an SLA Engine to check the conformance levels of the agreements between the CloudSocket Brokers and the CloudSocket Customer and produce respective violation events to be consumed by the component responsible for BPaaS adaptation, iv) a Monitoring Engine to monitor the BPaaS quality at all levels as well as sense the respective contextual information, v) an Adaptation Engine to trigger rules based on the contextual monitoring data and SLO violation events and perform the respective adaptation actions required to maintain the quality of service of a BPaaS, vi) a Mediator to map and transform the output data of one service to the input data of the subsequent service for a BPaaS workflow to guarantee the normal continuation of the BPaaS workflow in case of service substitutions or workflow re-concretisation, vii) the Cloud Provider Engine to deploy the BPaaS according to the respective bundle's deployment plan possibly in multiple clouds and to manage the lifecycle of the components deployed.

7.2 Functional Capabilities

The following scenarios identified in the BPaaS Execution Environment detail the high level functionalities that are needed to cover the complete lifecycle in the execution phase. These scenarios are depicted using the UML nomenclature.

Based on the different scenarios, we can identify the following list of functionalities:

- Deployment of predefined BPaaS bundles according to their enclosing deployment plan, which have been published via the Marketplace.
- Management of the deployed BPaaSs, through the graphical user interface.
- Execution of the deployed BPaaSs.
- Monitoring of the BPaaS execution environment across all levels.
- Management and assessment of the service level agreements.
- Evaluation of the adaptation rules based on BPaaS monitoring and contextual data.
- Multi cloud reconfiguration in the context of adaptation rules for IaaS/PaaS[2]/SaaS and BPaaS levels.

7.2.1 Deployment of BPaaS

Use Case id	EE-UC-1-Deployment of BPaaS
Title & Description	Deployment Business process as a Services. The CloudSocket Customer has identified the BPaaS bundle that fulfils his/her needs. The user selects and purchases it in the Marketplace, and after some possible user and account management steps, the BPaaS deployment is started. Then, the environment is configured to support this BPaaS bundle and all necessary components are deployed. Afterwards, the organization can access and manage the underlying workflow of the BPaaS bundle. During the BPaaS deployment according to the respective deployment plan in the bundle, the CloudSocket Customer can follow up the status of their selected BPaaS deployment in terms of which deployment task is currently executed and which ones have already been completed, where such tasks can include the deployment of VMs and software components and their configuration and the deployment of monitoring sensors. Besides, the CloudSocket Broker can also monitor the status of all deployments and the environment to guarantee the correct behaviour.

[2] See D3.3 BPaaS Allocation and Execution Environment Blueprints, chapter 4.1

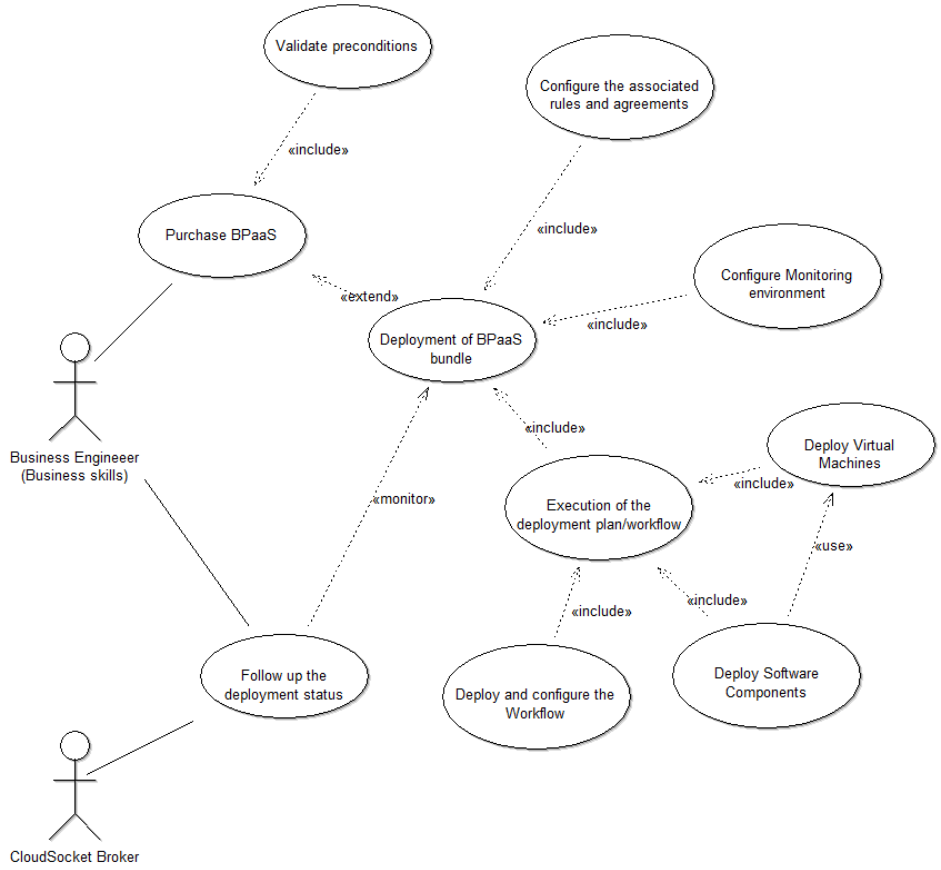
Copyright © 2015 BOC and other members of the CloudSocket Consortium

Actors	CloudSocket Customer and CloudSocket Broker
Use Case Objective	Deploy the selected BPaaS bundle in the BPaaS Execution Environment.
Pre-Conditions	The CloudSocket Customer and his/her organization have already been registered in the environment. All accounts and agreements with third parties whose services are to be exploited in the context of the current BPaaS have been previously generated and achieved, respectively. It is the responsibility of the Marketplace to manage or validate the associated prerequisites of the BPaaS bundle. The user identification is delegated to the authentication mechanism of the Marketplace in order to allow the single sign on of these users across the different components with which the user will interact in the same or different environments. The CloudSocket Customer has already purchased the bundle and as all pre-requisites are fulfilled, he/she can initiate the bundle deployment in the BPaaS Execution Environment.

Process Dialog	• The CloudSocket Customer initiates the BPaaS bundle deployment and accept the conditions of the service.
	• The deployment operation starts configuring the whole BPaaS environment and deploying all necessary components.
	  ○ Deploy the associated software components included in the workflow of the BPaaS by first creating the respective VM(s) in the selected Cloud Providers and then deploying and running the necessary component software.
	  ○ Configure the monitoring environment, by deploying the different probes/sensors across the appropriate VMs (e.g., software component VM, Workflow Engine VM, etc.) in which they should be situated.
	  ○ Configure the associated rules and conditions, which are defined in the BPaaS bundle, and distribute them or their corresponding parts to suitable components of the BPaaS Execution Environment.
	  ○ If all the previous actions are successfully performed, such that all the internal and external services are reachable and the environment is configured correctly to monitor and follow up the agreements defined for this BPaaS at the different levels, then the environment deploys and configures the BPaaS workflow defined in the bundle.
	  ○ Create the agreement associated to this BPaaS Bundle, which has been previously accepted by the customer, and start to follow up it.
	• The BPaaS bundle is deployed, the agreement is confirmed and they are available for the CloudSocket Customer.

Variations	If something happens along the deployment process: • The failure semantics, included in the deployment plan of the respective BPaaS bundle, are followed. This can mean rolling back to the initial state or dealing with local deployment problems through adaptation (e.g., select another VM from a different provider to instantiate). • The CloudSocket Broker can analyse the problem in order to resolve it and notify it to the BPaaS Customer.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	During the deployment process, the different actors can see the status of the BPaaS bundles in order to follow them up. The environment includes different levels of details depending on the technical skills of the roles: • A technical role in the CloudSocket Customer Organization can just see what is the overall status (e.g., pending) of the deployment, especially if he/she does not have the appropriate knowledge to understand the respective details. • The CloudSocket Broker needs to see all details (for example, which step is currently executed, what were the previous steps, what was the final state of these steps) in order to check if a problematic situation not previous anticipated has been reached. If the failure actions cannot handle this situation, then the CloudSocket Broker needs to go to the allocation phase in order to modify the corresponding deployment plan of the bundle.
Post-Conditions	The BPaaS bundle is ready to be used.
Diagrams	 **Figure 45 Use Case Diagram – EE-UC-1-Deployment of BPaaS**

Copyright © 2015 BOC and other members of the CloudSocket Consortium

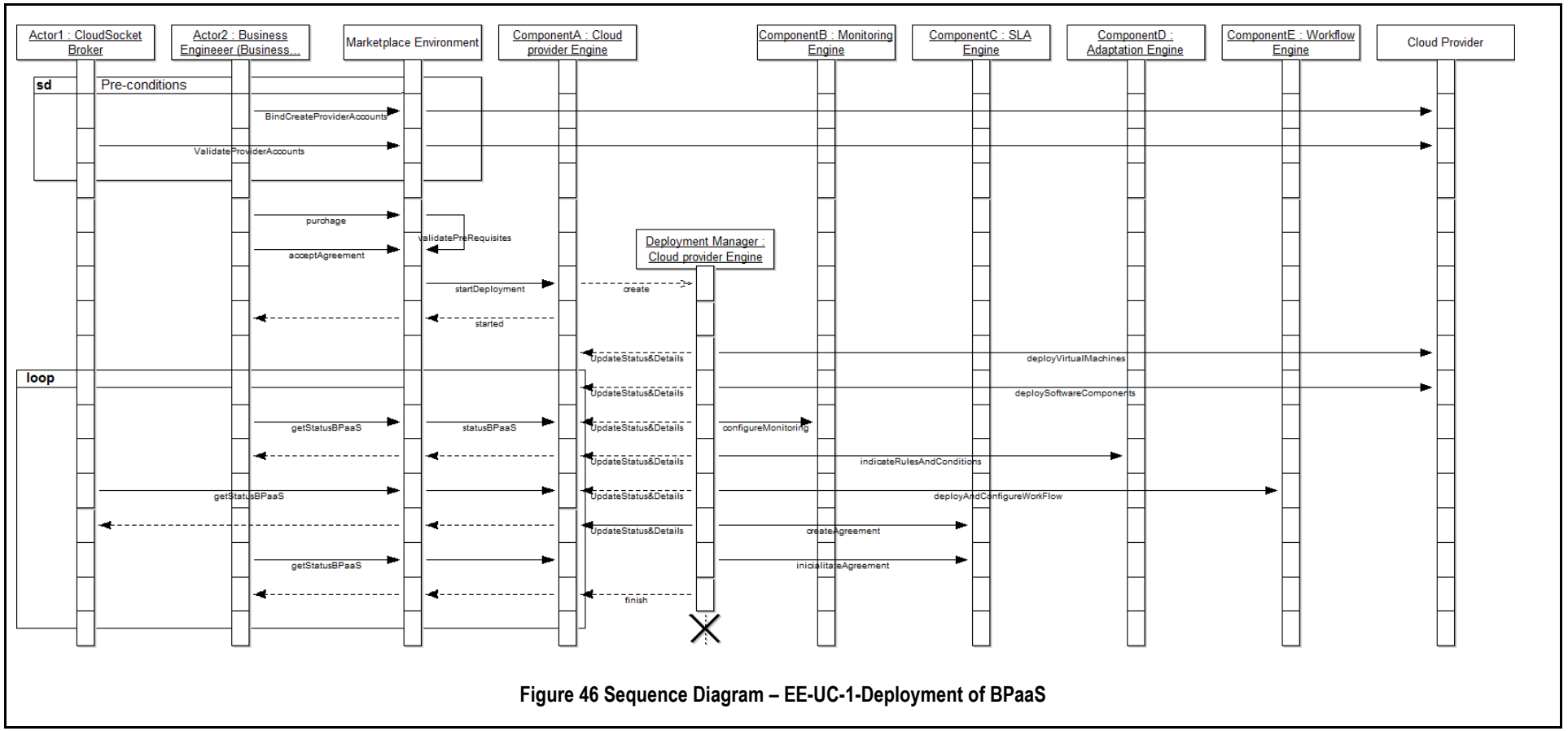


Figure 46 Sequence Diagram – EE-UC-1-Deployment of BPaaS

Table 18 BPaaS Execution Environment – Use Case 1- Deployment of BPaaS

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.2.2 Execution of the BPaaS

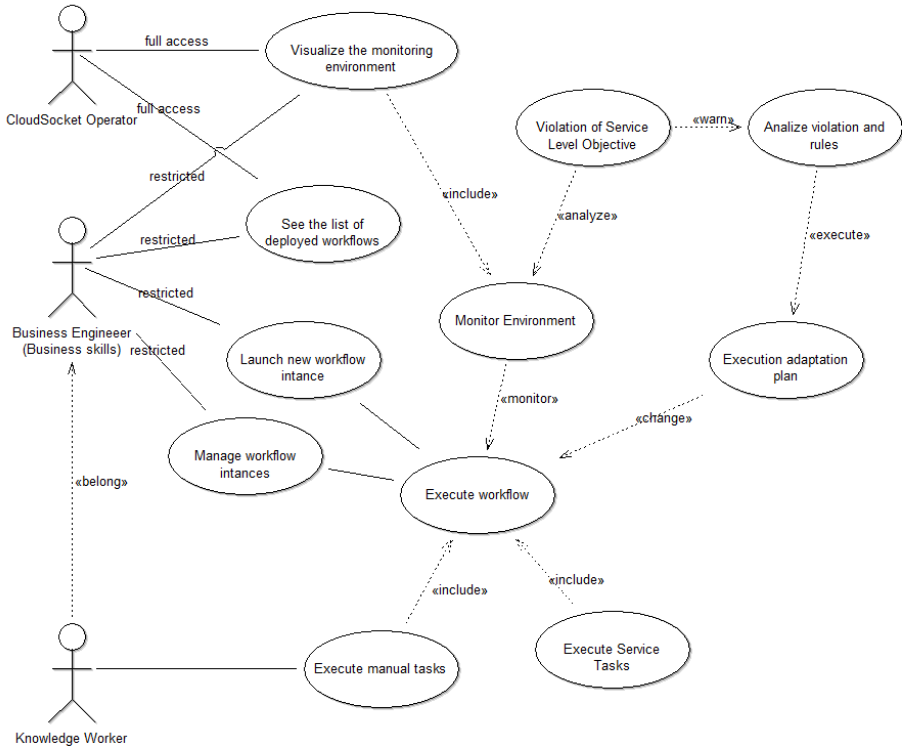
Use Case id	EE-UC-2 – Execution of the BPaaS
Title & Description	Execution of the Business Process as a Service The CloudSocket Customer, who is responsible for managing the business process of the BPaaS for his/her organization, desires to launch a new business process. He/she identifies it on the Marketplace and selects it in order to create an instance and launch it. Then, the business process instance is executed step by step, following the definition of the respective workflow in the BPaaS bundle. The BPaaS Execution Environment and the Workflow Engine in particular, is responsible for managing all the tasks and orchestrate the workflow. There are two main types of tasks: • Automatic tasks are managed by the environment, without any human intervention. They can be processed internally or through external services (atomic services), which are provided by third party cloud providers (registered previously in the registry). The system doesn't need to interact with Knowledge Workers in order to fulfil the respective task execution. • Manual tasks managed by Knowledge Workers, belonging to the CloudSocket Customer organisation. The environment just have to know when the task has been executed and what was the output produced. This involves the respective interaction between the Knowledge Workers and the UI (in alignment with the Workflow Engine) to indicate the task end and upload the respective output, if needed. The tasks assignment could correspond to employees of different departments of the end-user organization which are mapped to the role of Knowledge Worker. Different types of manual task assignment are planned to be supported. During the workflow execution, the environment monitors the workflow at the different levels, namely IaaS, PaaS, SaaS and BPaaS. If the environment detects any SLO violation, it informs the Adaptation Engine which analyses such violations along with any other contextual information and decides if it is needed to act through executing the adaptation plan of a certain adaptation rule leading to the execution of some adaptation actions, including the substitution of services or the scaling of software components, in order to maintain the quality of service level promised.
Actors	CloudSocket Broker and CloudSocket Customer each one involving technical roles (e.g., System Architects and Knowledge Workers, respectively).
Use Case Objective	Execute and orchestrate the deployed BPaaS, making use of the authorisation policies, which depend on the organizations and their security permissions (operational, technical, human interaction)
Pre-Conditions	The user identification is delegated to the authentication mechanism of the Marketplace to enable single sign on for the users desiring to interact with the components of the BPaaS Execution Environment. The workflow of the BPaaS has been correctly deployed and the environment has been accurately and properly configured.
Process	• The CloudSocket Customer desires to see the deployed business processes in the

Copyright © 2015 BOC and other members of the CloudSocket Consortium

Dialog	Workflow Dashboard • The Workflow Engine Dashboard identifies the CloudSocket Customer with the assistance of the Marketplace (by e.g. exploiting authentication tokens or any kind of user authentication proof) and returns the list of deployed business processes for only this company. • The CloudSocket Customer launches a new instance of the deployed workflow. • The environment starts a new instance for this workflow, which has been previously deployed. • The CloudSocket Customer can manage the workflow instances generated (e.g., suspend, resume and cancel). • The workflow is executed in a step by step according to the control flow of its specification in the BPaaS bundle by differentiating between the running of the two aforementioned types of tasks as follows o For service tasks, the following two alternative cases apply: ▪ Internal execution, where software or scripts have to be executed. It is managed by the Workflow Engine in order to cover the task's behaviour, without calling third party services. ▪ Running Exposed services (external or internal) to cover the functionality of the task, which are identified by an endpoint. The Workflow Engine does some internal activities to finalize the handling of the task and then moves to the next task in execution o Manual Tasks: These tasks should be managed by Knowledge Workers as follows: ▪ Depending on the type of assignment, the Knowledge Worker either receives, in the inbox, the pending tasks to be dealt with or selects a task not yet assigned to him and any other worker. ▪ The Knowledge Worker manages the manual actions and finalizes the task. ▪ The Workflow Engine finalises the task, also processing the user-generated data (which have been uploaded) and goes to the next step at the defined workflow. • Monitoring Engine collects metric measurements so as to monitor the BPaaS instance across different levels. • The collected monitoring data can be visualized through the dashboard; depending on the actor the information shown will be more detailed. o The CloudSocket Broker can access all the information related with the executed workflow instances in the environment. This level of detail is necessary for analysis and improvement reasons, covering cases of identifying problems and correcting them or optimizing a BPaaS bundle based on the previous adaptation history. o The CloudSocket Customer can only access information of the associated instances of his/her end-user organization. He/she needs to know, at a high level, the workflow instance status and the origin of the problems (if any). • The SLA Engine analyses these measurements in order to identify possible SLO violations which are part of the agreed SLAs when the purchase is done. • The SLA Engine communicates SLO violations to the Adaptation Engine. • Moreover, the Adaptation Engine also receives the notifications for the subscriptions of the metric conditions or SLO, directly from the Monitoring Engine.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	• The Adaptation Engine analyses the conditions of adaptation rules included at this BPaaS bundle and executes the respective adaptation actions, when these rules are triggered, in order to maintain the quality of the service offered.
	  ○ If the Adaptation Engine has to perform any kind of adaptation action, it might need to:
	  ▪ Suspend the execution of the workflow instance
	  ▪ Execute the adaptation actions possibly with the cooperation of the Cloud Provider Engine (adaptation at the infrastructure level) and the Mediator (adaptation at the service level with data incompatibility involved).
	  ▪ Inform the Workflow & Monitoring Engine about modifications at the workflow, service and infrastructure level
	  ▪ Resume the execution of the possibly modified workflow instance.

Variations	There are variations depending on the type of adaptation actions to be performed: • The Cloud Provider Engine might not be called • The Mediator might not be called • The workflow specification for the suspended instance does not need to be modified • The instance might not even need to be interrupted and then resumed.
Post-Conditions	n.a.
Diagrams	 **Figure 47 Use Case Diagram – EE-UC-2 – Execution of the BPaaS**

Copyright © 2015 BOC and other members of the CloudSocket Consortium

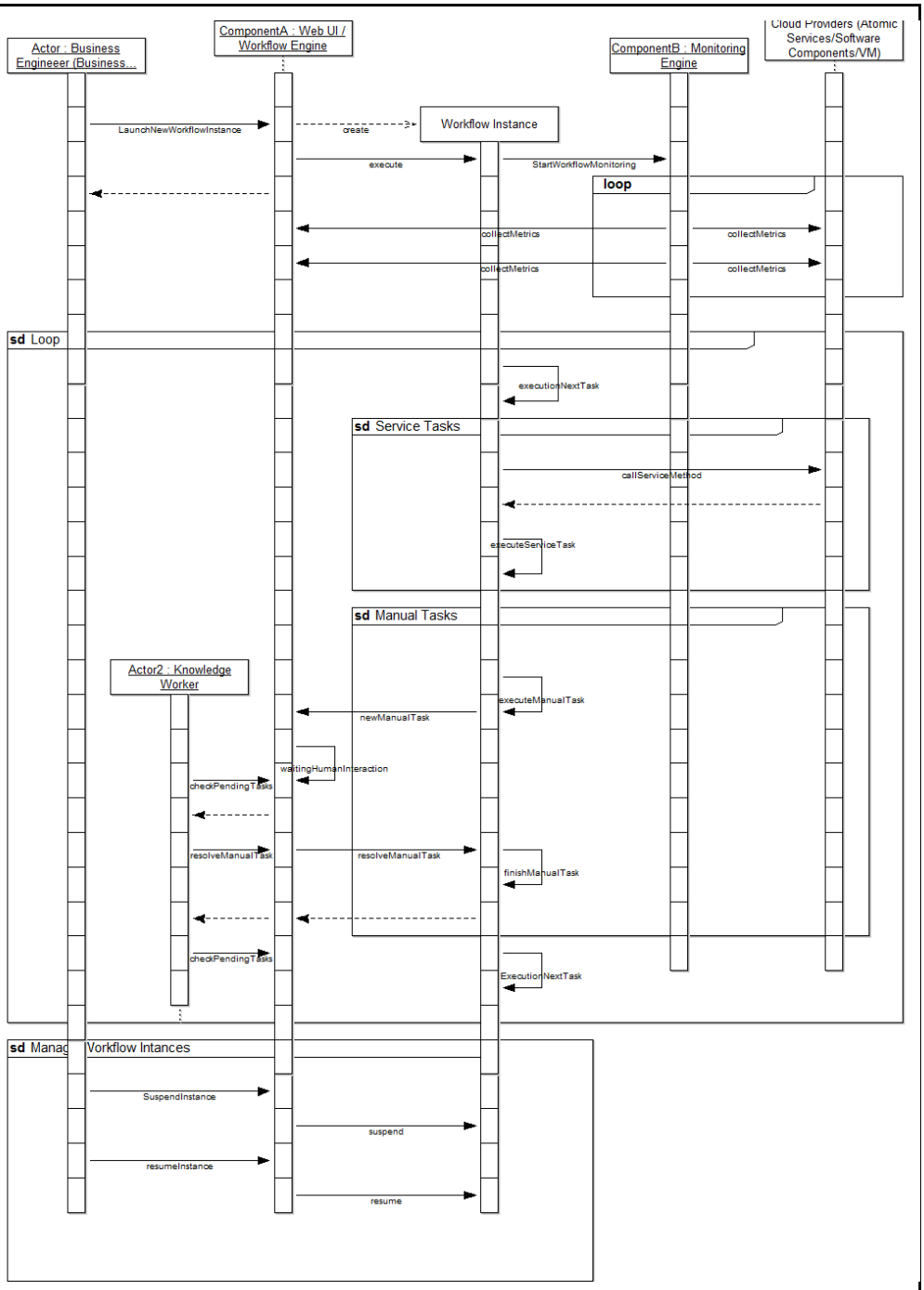


Figure 48 Sequence Diagram – EE-UC-2 – Launch/Execute BPaaS instance

Copyright © 2015 BOC and other members of the CloudSocket Consortium

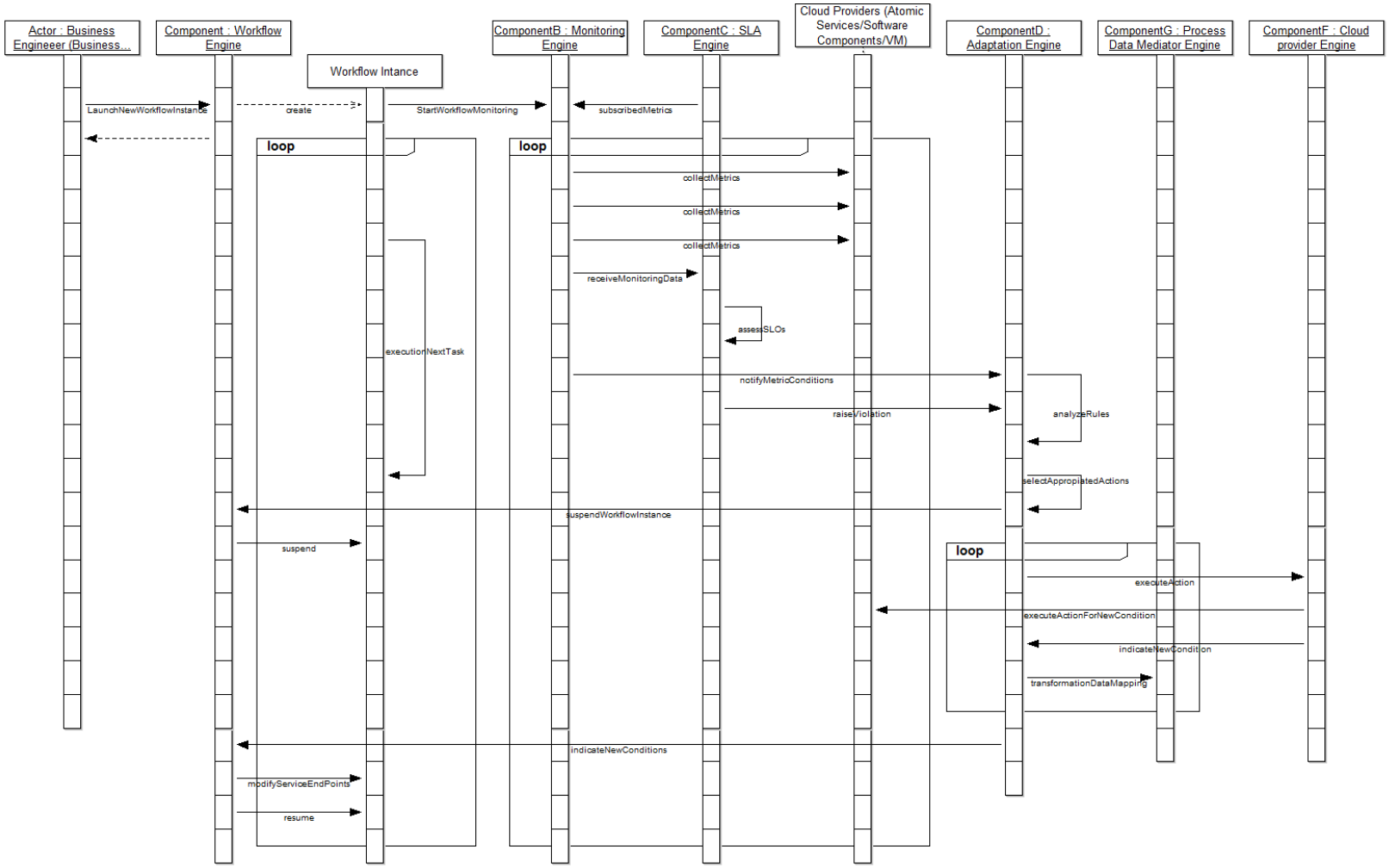


Figure 49 Sequence Diagram – EE-UC-2 – Reconfiguration environnement

Table 19 BPaaS Execution Environment – Use Case 2 – Execution of the BPaaS

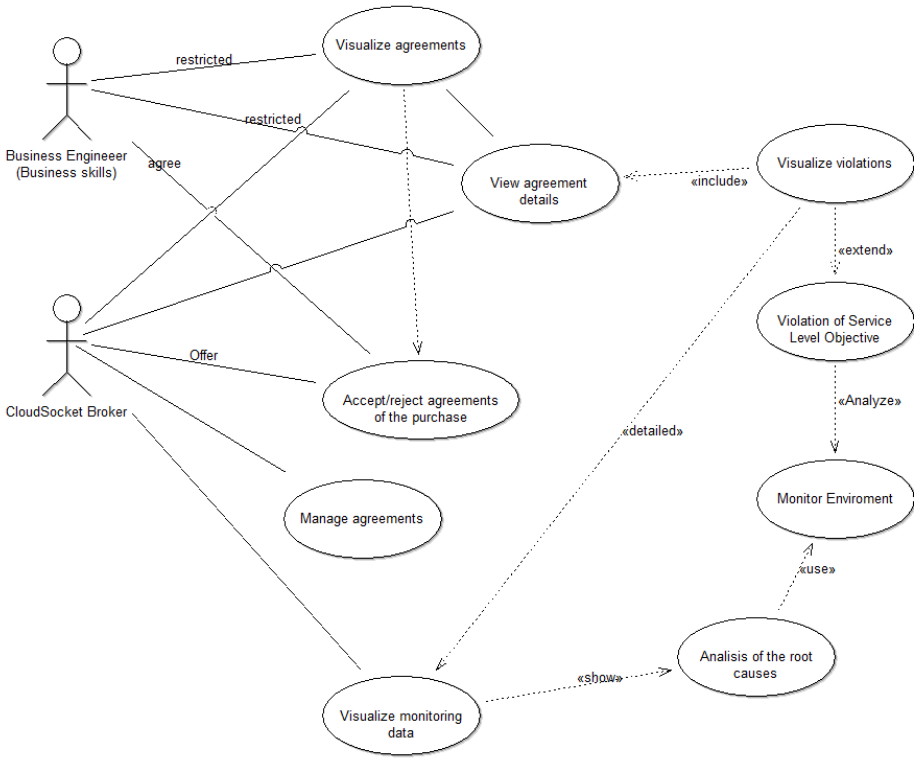
Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.2.3 Monitoring of Agreement Status

Use Case id	EE-UC-3 – Monitoring of Agreement Status
Title & Description	***Monitoring Agreement*** This scenario enables both the CloudSocket Customer and CloudSocket Broker to check the status of their agreements. Besides, they can see the incurred violations and the details of the metrics associated to the SLOs.
Actors	CloudSocket Customer (Business Engineer) and CloudSocket Broker (Technical Consultant, Business Consultant).
Use Case Objective	The aim of this use case it is to detail mechanisms provided to CloudSocket Customers and CloudSocket Brokers in order to follow up existing agreements in terms of service levels obtained and to get information about incurred violations, if any. Different levels of details will be visualized to these actors. While the CloudSocket Customer will have access solely to information specific to the agreements/SLAs related to BPaaS bundles purchased in terms of metrics directly referenced by these SLAs, the CloudSocket Broker will be able to get detailed information of all available agreements of all BPaaS bundles owned across different levels spanning KPIs for business roles and SLOs for technical roles, allowing to identify what are the root causes of violations.
Pre-Conditions	The SLAs are created by the BPaaS Allocation Environment as part of the BPaaS bundle which has been purchased by the CloudSocket Customer. Through the purchasing, the customer agrees with the content of the BPaaS SLAs.

Process Dialog	• During the execution of the BPaaS bundle, it is possible to see the status of the agreement and the associated monitoring data.
	• The CloudSocket Customer desires to check current status of all its active SLAs through the SLA Dashboard or the CloudSocket Broker wants to check current status of all active agreements for all BPaaS bundles owned.
	• First a list of all existing and active agreements is presented. Below in this section a detailed definition of what it is considered “active” agreement is provided. The CloudSocket Customer only can see the active agreements, over which have visibility, and the CloudSocket Broker can see all of them for all its clients.
	• For each agreement the CloudSocket Customer and the CloudSocket Broker can check the conformance to established agreement terms, after the purchase has been performed. In case of violations:
	  ○ The CloudSocket Customer can see the violation, the respective penalty and the associated metric information, in a high level description.
	  ○ The CloudSocket Broker can also see the violations but in a more detail level obtaining monitoring information related to the different term values and analysing more deeply the information to find the root causes. He/She might also be able to see summaries and statistics over the penalties that had to be paid.
	The SLA implements the following steps:
	• When the BPaaS bundle is purchased, the CloudSocket Customer reviews the agreement and decides if he/she accepts or rejects it.
	  ○ In case of rejection, the purchase is cancelled.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	○ In case of acceptance, the respective SLA is considered as agreed and the process continues. • When the BPaaS bundle is deployed, the monitoring for the associated SLA is initiated. • The SLA can be terminated when the respective completion conditions hold. As a result, the monitoring is also ended in the context of this SLA.
Variations	n.a
Post-Conditions	n.a
Diagrams	 **Figure 50 Use Case Diagram – EE-UC-3 – Monitoring of Agreement Status**

Copyright © 2015 BOC and other members of the CloudSocket Consortium

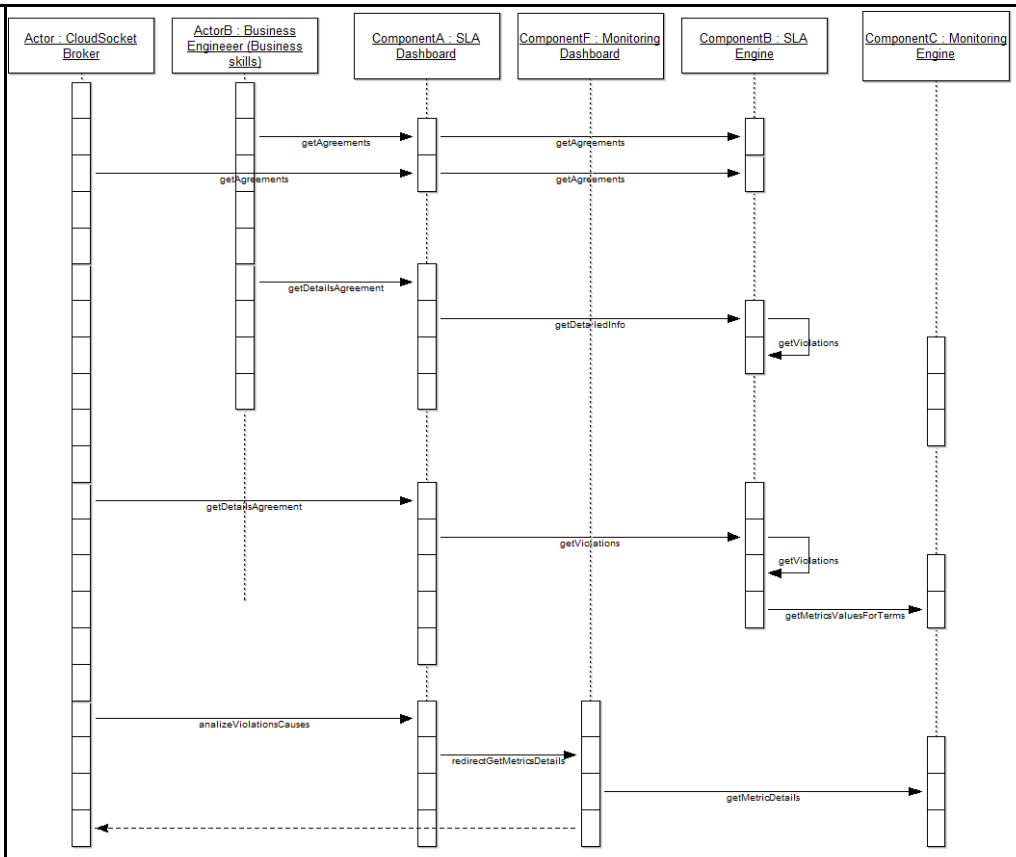


Figure 51 Sequence Diagram - EE-UC-3 – Monitoring of Agreement Status

Table 20 BPaaS Execution Environment – Use Case 3 – Monitoring of Agreement Status

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.2.4 Workflow Environment Management

Use Case id	EE-UC-4 – Workflow Environment Management

Title & Description	This scenario details the complete management of the workflow engine, including the deployed workflows and their instances.
	
	This scenario considers covering: i) the management of the own workflow engine for the CloudSocket Broker; ii) managing lifecycle of the deployed workflows, which have been defined at BPaaS Bundles, iii) and their instances. These features are not accessible for all the involved actors, as the CloudSocket Broker will have full access to them while CloudSocket Customer will have partial access to them:
	
	• The administration aspects related with the Workflow Engine for both CloudSocket Broker and CloudSocket Customer. While the CloudSocket Customer will only be able to administer details about its organization, the CloudSocket Broker will be able to administer such details for any existing organization. The details managed/administered include
	  o Edit Organisation associated information
	  o Management of users registered to a particular organisation
	  The broker will also be able to manage the data base and its entities.
	• The necessary operations to manage and configure the workflows and their instances for the execution phase. The CloudSocket Customer can manage the instances of the workflows purchased, while the CloudSocket Broker has full control over all workflows and instances.
	  o Workflows management: It consists of managing the lifecycle of deployed workflows. This management is the responsibility of the CloudSocket Broker.
	• Manage Workflow Instances (launch / stop / destroy). This includes the definition of tasks which can be both manual and automatic. Both actors can manage this, but the CloudSocket Customer only can see and manage instances of his/her organization.

Actors	CloudSocket Broker and CloudSocket Customer
Use Case Objective	Allow to perform necessary operations to configure/manage the Workflow Engine. These include: • Deployed workflows (included in the BPaaS Bundles) lifecycle management. • Workflow Instances management (lifecycle and configuration) Database, users and organizations management.
Pre-Conditions	A number of BPaaS Bundles and workflows exist for the CloudSocket Broker. These bundles/workflows have been already been deployed in the BPaaS Execution Environment. The users identification is delegated to the authentication mechanism of the Marketplace in order to enable single sign on for users when requiring to interact with platform/environment components.
Process Dialog	This process will happen through interactions with the Workflow Engine GUI. The administration aspects related with the Workflow Engine: • CloudSocket Broker will be presented initially with a list of existing organizations.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

	Once an organisation is selected, the process will be the same for both actors. • Edit Organisation details will allow to modify data associated with a specific organisation. This process will be synchronized with Authentication Engine. • Management of Users will allow adding, removing and modifying existing users for a particular organisation. This process will be synchronized with authentication and authorization, which is covered by the Marketplace. • In addition to these, it will be possible that the CloudSocket Broker checks the Database entities that the Workflow Engine is using internally for Workflow definition and execution. The necessary operations to manage and configure the workflows and their instances: • The CloudSocket Broker accesses the Workflow Engine UI where all defined workflows (defined in BPaaS bundles) and all workflow instances are presented, for any organization. The CloudSocket Customer can only see the information related with respect to his/her organisation (and is of course not able to manage the workflows but just see their information mainly in terms of their specification). • For any workflow of the BPaaS Bundle, the CloudSocket Broker is able to deploy and undeploy it. The automatic workflow deployment is part of the overall bundle deployment as identified in the aforementioned use case. The CloudSocket Broker can also move workflows from one Workflow Engine to another one, when needed, provided that a distributed version of the BPaaS Execution Environment exists. • Regarding workflow execution, the CloudSocket Broker can access the list of all active workflow instances as well as all details of workflow deployments (process definitions, images, business rules, etc.). The CloudSocket Customer can only see the information related with respect to his/her organization. • The CloudSocket Broker can manage workflow instance lifecycles (launch / stop / resume/ destroy). The CloudSocket Customer can only manage the instances of BPaaS bundles purchased.
Variations	n.a
Post-Conditions	n.a

Copyright © 2015 BOC and other members of the CloudSocket Consortium

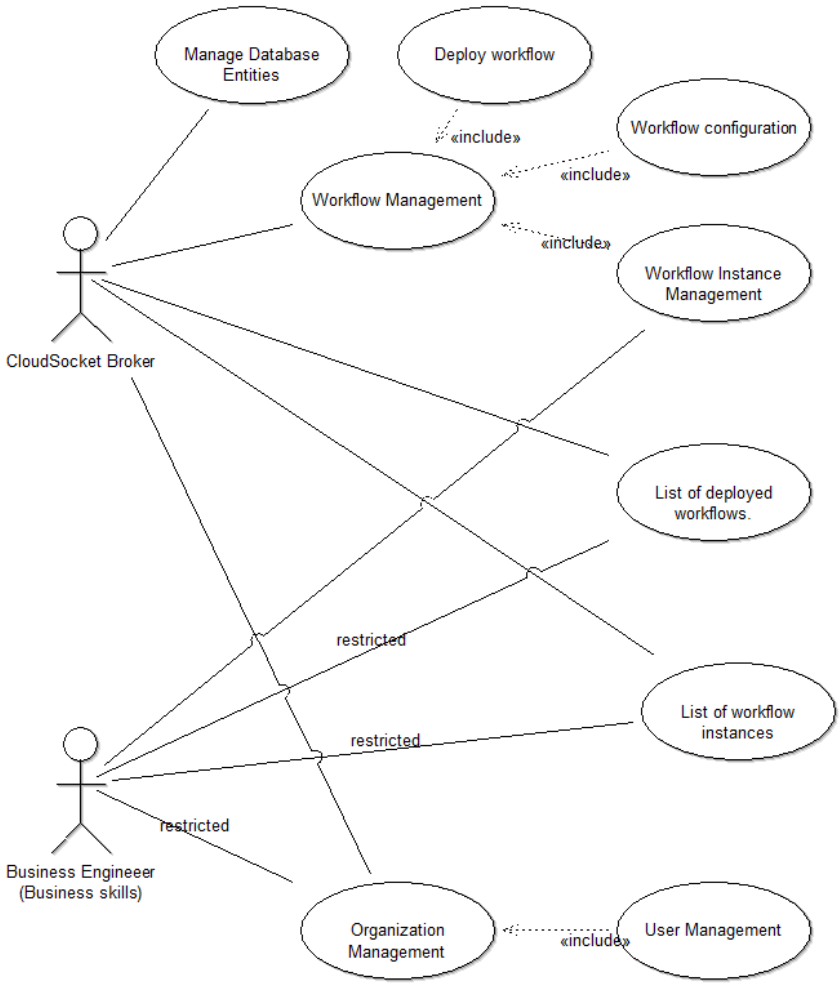
Diagrams	

Figure 52 Use Case Diagram - EE-UC-4 – Workflow Environment Management

Copyright © 2015 BOC and other members of the CloudSocket Consortium

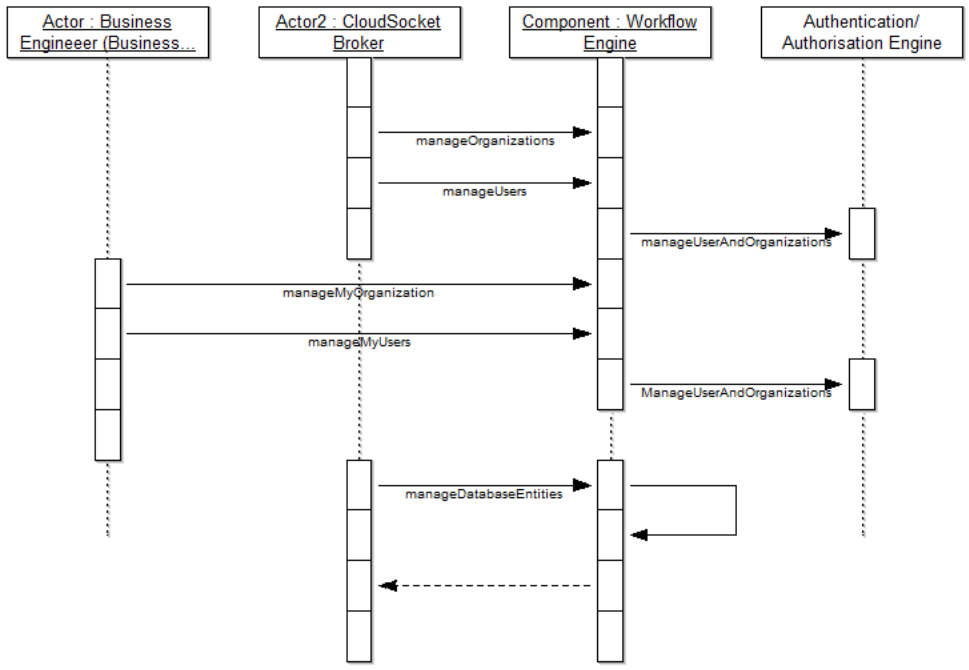


Figure 53 Sequence Diagram – EE-UC-4 – Workflow Environment Management

Table 21 BPaaS Execution Environment – Use Case 4 - Workflow Environment Management

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3 Components

The architecture supports a multitier topology to decouple the user interface from the business logic core. Nevertheless, it depends on the specific component whether to apply more tiers or patterns to create a loosely coupled design. In the following subsections, the two main layers and their components are analysed, beside their interactions with the rest of environments.

7.3.1 User Interface workspace

This layer contains all the graphical interfaces components to interact with the users (CloudSocket Brokers, and CloudSocket Customers). The interface should be mainly web-based, but it could contain other interface approaches for example for backend configuration. The following components are included at this level.

7.3.1.1 Web UI / Workflow-Engine

Web UI / Workflow-Engine is responsible for interacting with the different actors involved in the lifecycle of a workflow. The CloudSocket Customer, considered as a tenant, can purchase the business processes, follow up the different instances and assign their tasks to its employees. Besides, the CloudSocket Broker can manage the business process for the different tenants by using the graphical user interface. However, the respective information needed for realizing the underlying functionalities is not stored and managed by this component but by the Workflow Engine core.

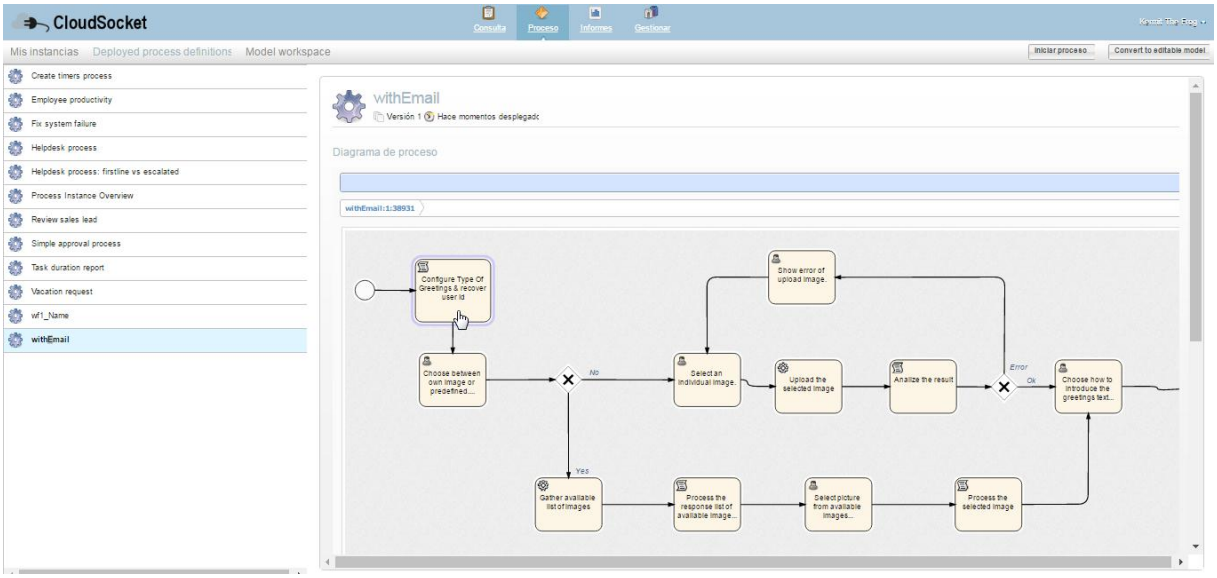


Figure 54 BPaaS Execution Environment – Workflow Engine User Interface

Figure 54 indicates the user interface of the Workflow Engine based on YourBPM.

7.3.1.2 SLA Dashboard

SLA Dashboard is responsible to present summaries of the status of the different agreements between providers and the consumers (e.g., CloudSocket Brokers and CloudSocket Customers) with respect to how well the service levels agreed are maintained. The information on agreements and respective violations is collected through the SLA Engine and just visualized. As indicated above, the SLA Engine realizes the respective logic for SLA information collection.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

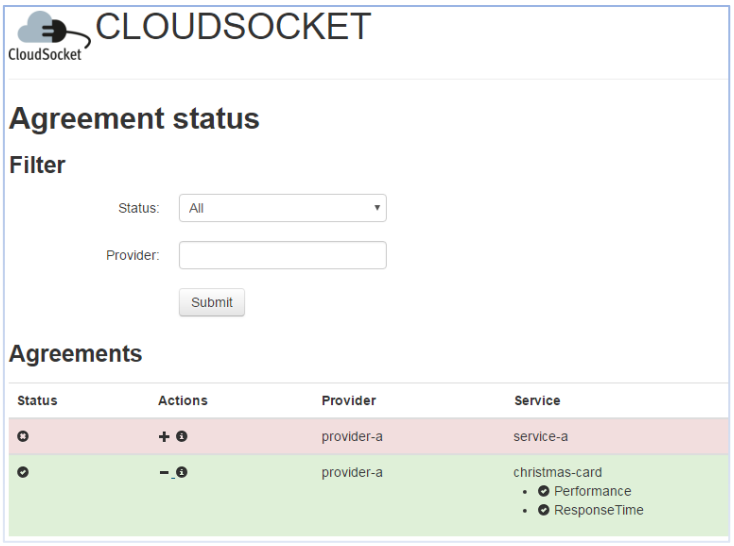


Figure 55 BPaaS Execution Environment – SLA Monitoring Dashboard User Interface

Figure 55 indicates the user interaction of the SLA monitoring.

7.3.1.3 Cloud Provider Engine Dashboard

The Cloud Provider Engine Dashboard provides technical details about the currently running and finished deployments of the software components and VMs across multiple clouds. This includes cloud specific details such as location, image or hardware and software component details such as the lifecycle commands and the communication between multiple software components. The Cloud Provider Engine Dashboard can be accessed by the CloudSocket Broker to retrieve more technical insights in the actual software component deployments.

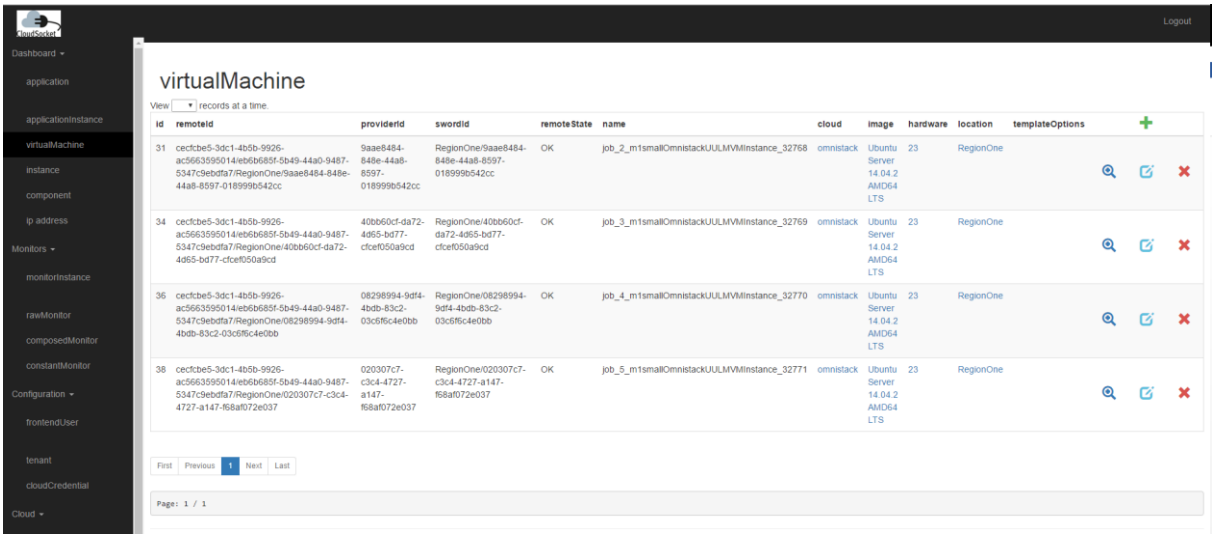


Figure 56 BPaaS Execution Environment - Cloud Provider Engine Dashboard

7.3.1.4 Monitoring Dashboard

Monitoring Dashboard is able to show the monitoring data associated to different BPaaS artefacts, such as atomic services, VMs, workflow engines, and instances of already specified and running workflows (plus their activities as another type of artefact that can be monitored). Moreover, when something goes wrong and a violation is generated, the user can drill down to see the performance of the underlying components and identify

Copyright © 2015 BOC and other members of the CloudSocket Consortium

which ones to blame for this violation. Thus, the SLA Dashboard redirects to the Monitoring Dashboard to see more fine-grained analysis of the monitoring information but only in the context of the CloudSocket Broker.

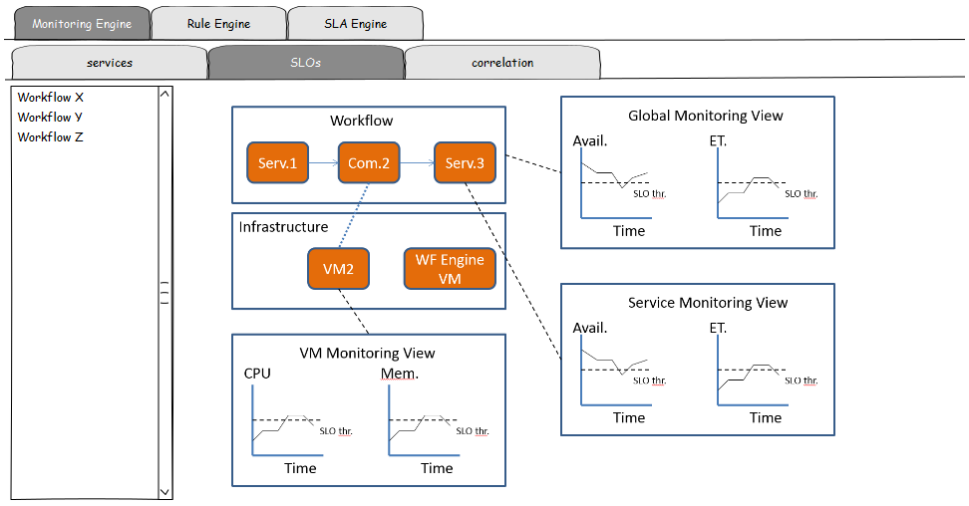


Figure 57 BPaaS Execution Environment – Monitoring Dashboard User Interface Mockup

Figure 57 shows the indicated user interface of the monitoring of a workflow on several levels and the corresponding SLA monitoring. The raw measurements can also be directly viewed in the Cloud Provider Engine Dashboard along with the technical assets, see Figure 58.

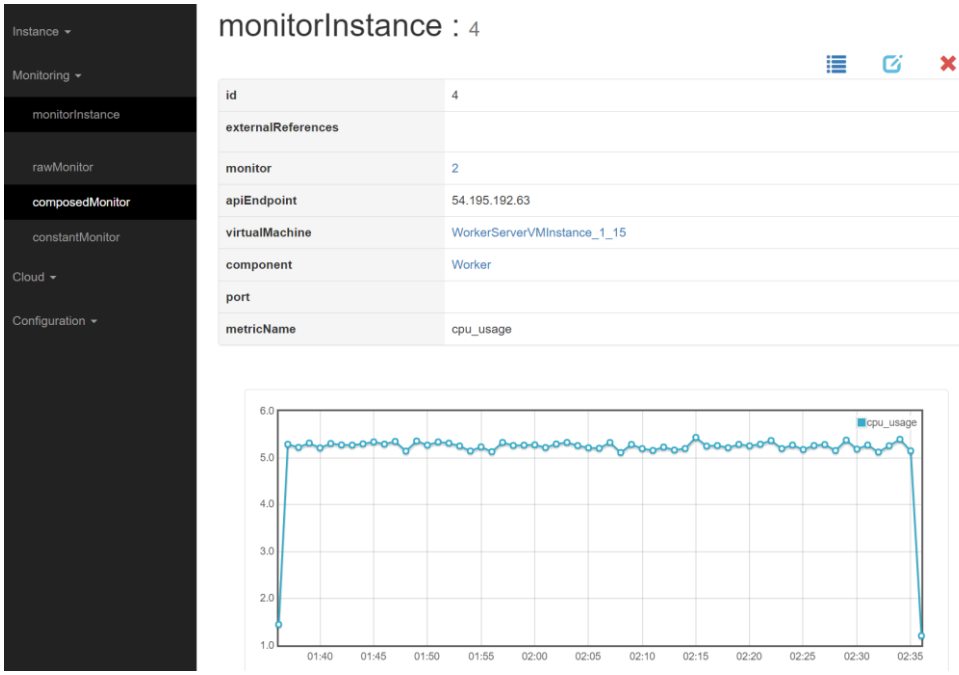


Figure 58 The unprocessed monitoring data in the Cloud Provider Engine Dashboard

7.3.2 BPaaS Middleware

This layer is in charge of managing all business logic and the data bases involved and exposing the functionalities via respective APIs through which interaction with other components and environments is enabled. This layer constitutes the core of the BPaaS Execution Environment. There is not a common data layer for each component in the core, due to the components' complexity and heterogeneity. Hence, each component will contain its own definition of the data base, internal sublayers and design patterns in order to have a completely decoupled and scalable environment.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.2.1 Workflow-engine

It is responsible for managing the deployment, execution and management of the different workflow instances. It will be multi-tenant leading to executing a workflow instance on behalf of one organisation by also taking care of the corresponding workflow and organisation data level. This component will expose a REST API interface allowing programmatic access to the different types of functionalities offered. This interface will be used both to support user interaction through the Web UI / Workflow Engine component as well as enable interaction with the rest of environments, e.g., to allow persisting, deploying and executing the business process.

- Tasks:
  - Deploy/redeploy a workflow in the workflow-engine, instantiate it and execute it.
  - Manage and follow the workflow instances, according to the workflow description in BPMN.
  - Interact with manual tasks of the workflow.
  - Manage the workflow engine environment.
- Main interfaces:
  - Deployment workflow phase:
    - Input: BPMN executable file (including the service endpoints for external service tasks), which is part of the BPaaS bundle defined in the allocation phase, and the Organization, who has purchased the bundle.
    - Output: when creating a new workflow deployment, the identification of the deployment and its status.
  - Execution workflow instance phase:
    - Input: actions to be executed (CRUD); for example the identifier of the workflow instance to be modified (to be suspended/resumed, change the service endpoints).
    - Output: Status message indicating the status/result of API method call, for example if the suspension of the instance request has succeeded or if the service task endpoint has been changed.

7.3.2.2 SLA Engine

It is responsible to follow up the different agreements created and assigned to the execution of purchased BPaaS bundles. This component should interact with the Monitoring Engine to obtain the measurements needed to enable assessment of SLOs. The interaction will be managed by a publish/subscribe mechanism or a pull model to obtain only the measurements for those metrics that directly map to the SLOs to be assessed. The defined SLOs will always be mapped to the actual metrics to be collected and supplied by the Monitoring Engine. The definition of metrics is done in the BPaaS Allocation Environment, where the Business KPIs have to be translated to the actual metrics pertaining to the monitoring of SLOs. Hence, the BPaaS Allocation Environment generates and configures the agreements while it is developing the BPaaS bundle. Afterwards, the CloudSocket Customers can select the appropriated BPaaS bundle in the Marketplace and accept the associated agreement. Then, the BPaaS Execution Environment needs to enforce the agreement, since the SLA is included in the BPaaS bundle that the environment receives during the BPaaS deployment.

If any violation arises, the SLA Engine will notify/broadcast to all registered components that are interested to be informed about the violation of the agreements, like the Adaptation Engine component which requires analysing adaptation rules and executing the respective actions. Moreover, such violations would be communicated to external components, such as the accountability and billing components, which would manage the necessary actions including the enforcement of penalties, discounts and charges.

- Task:
  - Subscribe to metrics directly related to SLOs of SLAs
  - Generate and store violation events following the boundaries associated to the guarantee terms/SLOs.
  - Manage the agreements and violations.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

    - Notify/broadcast to all registered components any violation of the agreements on which they have subscribed.
    - Access to the agreement and violation details, including historical data.
- Main interfaces.
    - Purchase phase:
        - Input: Template details for the BPaaS bundle.
        - Output: Consolidated agreement.
    - Deployment phase
        - Input: the agreement, included in the BPaaS bundle.
        - Output: the confirmation of the agreement enforcement and the status.
    - Execution phase:
        - Input: The agreement identifier in order to identify its associated metrics.
        - Output: The notification of SLA violations, which is composed by the identifier of the agreement plus what has been violated, how and when (i.e. SLO identifier and measurement).

7.3.2.3 Monitoring Engine

It is responsible to monitor a BPaaS and correlate/aggregate monitoring data from different levels, from atomic services or cloud components up to the level of workflows. It will expose an API in order to allow components, such as the SLA Manager and the BPaaS Evaluation Environment, to draw the information monitored by subscribing to particular metrics and perform the respective tasks assigned to them. In the case of the SLA Manager, this will involve performing an SLO evaluation, while, in the case of the BPaaS Evaluation Environment, this will involve performing background analysis of the monitored information in order to discover interesting patterns in the context of one or more business processes. The component will cover both raw metrics (direct measurements provided by deployed sensors or external measurement systems like PaaSs) and aggregated metrics (formulas to exploit metrics already implemented and produce the respective aggregated measurements). This component will also handle the monitoring of contextual information which will be handed over (through exposing a particular API) to the Adaptation Engine to enable it to completely assess adaptation rules.

As metrics are involved in SLO and contextual conditions, it is essential that they need to be defined beforehand in order to allow the Monitoring Engine to measure them and thus enable the evaluation of such conditions. The BPaaS Evaluation Environment has an interface to this component via the publish-subscribe mechanism.

The measurement database to store and execute continuous evaluation on monitoring data, will be managed by a unified API that can be implemented for any time-series database (TSDB). It will be engaged as an abstraction layer that helps to be able to use the technology, one needs for a specific job. The system can then use on specific TSDB for e.g. write-intensive activities and another one e.g. for activities the TSDB supports certain mechanisms, like continuous queries on InfluxDB (D3.3 2016, section 4.2.3.4).

- Task:
    - Allow implementation and description of user-defined metrics;
    - Metric specification modification leading to changing the monitoring infrastructure/environment, and new metrics definition for a BPaaS.
    - Monitor and aggregate metrics;
    - Inform interested parties about fresh metric values through the publish/subscribe mechanism
    - Inform interested parties about historical metric values through the use of the REST-API
    - Inform interested parties about contextual information
- Main interactions.
    - Input: Description of metrics (Sensor configuration for raw metrics and metric formula description to allow aggregation); Configuration of user-defined metrics (which includes how to download the implementation, for example as a rar file);

Copyright © 2015 BOC and other members of the CloudSocket Consortium

- Output: Measurement DB (which can be realized by a time series database or a semantic database or even both); Context DB (storage & update of contextual information); notification to subscribers.

7.3.2.4 Adaptation Engine

This component is responsible for the change of the BPaaS deployment (different services, different service configuration and workflow structure, new services) to resolve the problematic situations identified by adaptation rules. Different types of adaptations will be performed at different levels of abstraction. In particular adaptation actions on VMs (deploy, migrate), services (substitute, rebind), workflow tasks (e.g., re-execute, map to different service composition) and the workflows themselves (recompose workflow) are provided.

The management of the adaptation rules will be covered by a subcomponent called Rule Engine, which is responsible to take decisions based on the environment variables and the performance levels encountered. So, it should interact with the Monitoring Engine and the SLA Engine to collect the needed data required for the evaluation of the rules exploited. The required data include SLO violations communicated by the SLA Engine as well as contextual information produced by the Monitoring Engine (in which the Rule Engine needs to subscribe). Such data are required in order to assess SLO and contextual conditions which constitute the left part of adaptation rules. In case that one or more rules are triggered, the respective adaptation actions will be executed - based on the settings from the BPaaS Allocation Environment - to maintain the quality of the service promised and of the experience of the stakeholders by, e.g., migrating services to other providers and deploying software components over different clouds. Adaptation rules and their respective will be described as workflows of adaptation actions, that can be processed by the Cloud Provider Engine (D3.3 2016, section 4.3.2).

The need for adaptation will be indicated by the Rule Engine as it possesses the knowledge of the adaptation rules (covering scalability and the fault-tolerance) and be supported by the Adaptation Engine which includes an adaptation library of actions that can be exploited to perform the different types of adaptation needed at the different levels. The Adaptation Engine will need to perform either one or more adaptation actions. In the first case, it will be responsible for just executing or delegating this action to another component (e.g., Cloud Provider Manager). In the second case, the actions to be executed will be described in a form of a workflow which will also dictate the sequence in which the adaptation actions have to be run, hence, there is a need for a (possibly internal to the Adaptation Engine) workflow engine able to execute the adaptation workflows.

- Task:
  - Evaluate adaptation rules which involves assessing contextual and SLO conditions (where the latter are already evaluated and sent in the form of SLO violations to the Adaptation Engine)
  - Manage the execution of an adaptation strategy (in the context of triggering a specific rule)
  - Adapt the BPaaS according to the adaptation strategy provided.
- Main interface:
  - Input:
    - To manage the rules: Violations, Metrics, description of adaptation rules (including event patterns that lead to their triggering) to adapt a BPaaS, policies for adaptation (e.g. max amount of VMs or service instances, cost limits), which are specified in the allocation environment. The following functionalities are covered:
      - register adaptation rules,
      - modify adaptation rules on demand,
      - get adaptation history for a certain BPaaS,
      - allow executing personalized/customized adaptations.
    - Referencing of the actual adaptation workflow (by the Rule Engine which is a sub-component of the Adaptation Engine) in rules stored in the Rule Engine, part of the Adaptation Engine, to allow its instantiation when a respective adaptation need arises.
  - Output:
    - Result of the executed adaptation.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.2.5 Cloud Provider Engine

This component is responsible for the complete deployment and lifecycle management of all the required components of the BPaaS, including software components and VMs across multiple clouds, with transactional semantics (at least for the deployment part). These capabilities will be managed by different subcomponents of the Cloud Provider Engine to provide a modular, flexible and scalable architecture. To exhibit these capabilities, the Cloud Provider Engine will build upon existing functionalities offered through the interfaces exposed by the cloud providers.

The BPaaS deployment with transactional semantics will be managed by the Deployment Engine, responsible to deploy, configure and operate all appropriate software on IaaS and PaaS infrastructure (D3.3 2016, section 4.1) before deploying a workflow in the Workflow Engine. In essence, the Deployment Engine will be responsible for executing the deployment plan included in the BPaaS bundle by orchestrating all necessary steps. This deployment plan will not only cover component deployment but also agreement registration and validation, monitoring infrastructure deployment and configuration, and workflow deployment in the Workflow Engine in order to guarantee that in the end the workflow of the BPaaS bundle will be ready for execution. If something goes wrong, then the transactional/failure semantics, which is defined as part of the deployment plan, will dictate what actions will have to be performed to remedy for this, which could involve rolling back the system or compensating previous deployment actions and performing new ones with the same goal.

The Cloud Provider Engine will expose two interfaces: (a) an interface to enable the performance of re-deployment actions in order to interact with the Adaptation Engine component, and (b) an interface to interact internally with the Deployment Engine for managing deployment transactionally.

The Deployment Engine sub-component will comprise different plug-ins to connect to the different clouds (by also exploiting previously generated end-user cloud credentials), allowing to interact with these clouds to execute a common action as, e.g., the concrete deployment actions for a VM will be different depending on the cloud provider, but the actual abstract action is the same: deploy a VM. Hence, this sub-component will allow providing an abstraction over the different specificities of cloud providers with respect to cloud management actions and it will be responsible for transforming abstract management actions to cloud-specific ones.

- Task:
  - Manage the complete BPaaS deployment.
  - Manage and abstract from current IaaS capabilities.
  - Manage the relationships with the different cloud providers and across different cloud service levels (IaaS/PaaS)
  - Offer scaling & migration capabilities to the Adaptation Engine
- Main interfaces
  - Input:
    - Deployable workflow (BPaaS bundle):
      - BPMN executable file (including the endpoints services of the services tasks)
      - SLA agreement definition (mapping the Service Level Objectives to measurable metrics)
      - List of applicable (adaptation) and alternatives rules to be applied
      - Semantic Metadata (describing metrics or adaptation actions) to allow taking decisions in the execution phase.
    - Deployment plan along with transactional semantics, deployment actions to be executed in the context of an adaptation rule which are dictated by the Adaptation Engine (as part of the BPaaS bundle)
  - Output: The result of executing the deployment plan.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.2.6 Process Data Mediator

The component is responsible for mapping output data of one component/service to the input data of a subsequent component/service in execution and performing the respective data integration/transformation in order to guarantee an error-free execution of the workflows. It will rely on ontology based mapping (when the IO of the process activities/components is annotated via ontology concepts during the design of the business process and workflow) as well as data mapping and transformation techniques (even in case where no ontology annotations are provided by relying on the schemata of the data involved - we can assume XML schemata here). The following scenarios through the use of such component are foreseen.

- At design time, when the complete BPaaS workflow has been generated, it can include the specification of data transformation tasks. The process data mediator offers an API through which the respective required transformation tasks can be realized while external data transformation services could also be alternatively exploited.
- At design time, when the BPaaS workflow has been completed but there is no specification of any transformation task for data incompatible services, there can be some annotations mapping data to ontology concepts and also indicating that the output data of one service has to be mapped to the input data to the next service. Such annotations can then be exploited by the workflow engine in order to interrupt the workflow execution, run the respective API method of the Process Data Mediator, and once this is done, use the transformed data as input to the next service after of course the respective workflow execution is resumed. The API method of the Process Data Mediator should be responsible for finding the mapping between the data of the pair of services involved and then run the respective data transformation.
- At runtime, the workflow of a BPaaS has to be adapted through the substitution of one service (B) with another one (B1). If the output of the previous service in execution (A) is not compatible with the new service (B1), then there is a need for data transformation. As such, the respective API method of the Process Data Mediator could be exploited which could inspect the (possibly ontology-based) definitions of the services involved (A and B1) in order to produce the mapping and then use it to perform the respective transformation. Then, as the workflow execution has been suspended, the workflow will be resumed with the execution of the new service (B1) which has now the appropriate input.

Based on the above analysis, we can identify the three main modes of usage for the Process Data Mediator: (a) at runtime to realize a particular data transformation task already specified in a BPaaS workflow; (b) at runtime to realize a data transformation functionality required between two services with incompatible data in a certain BPaaS workflow in an actually indirect manner (as no data transformation is prescribed directly in the workflow but there is only an annotation dictating it); (c) at runtime to check and adapt the data of a previously executed service with those needed by a new service which substitutes the one that was next in the execution order.

- Task:
  - Map output data of one component/service to the input data of another component/service;
  - Perform the respective transformation based on the mapping specification/result
- Main interfaces:
  - Input: data schema for a service/component or ontology annotation, specification of the mapping in a specific language (optionally transformation logic/specification)
  - Output: transformed data to be used as input to the next service in execution order

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.2.7 Component Diagram

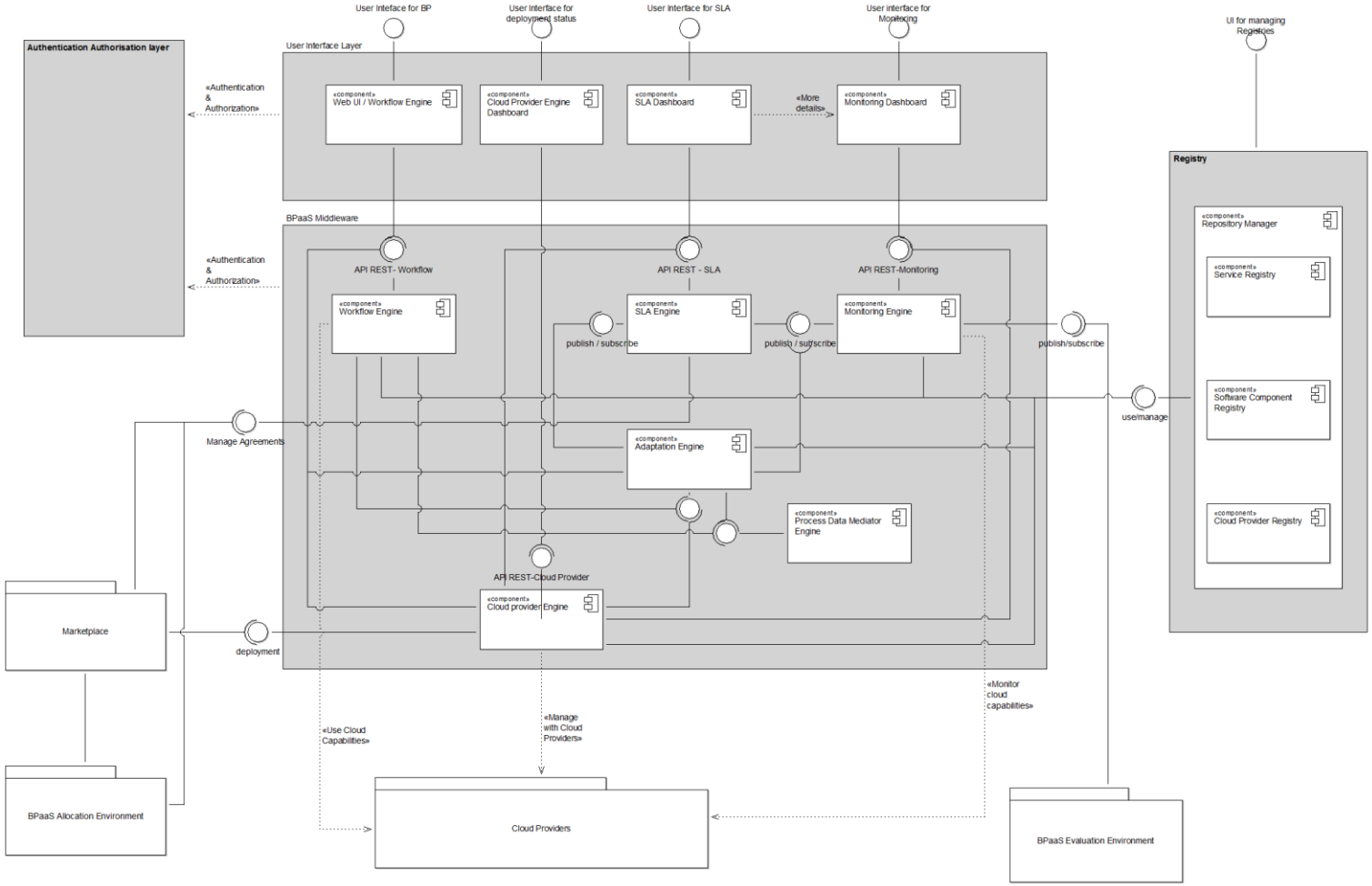


Figure 59 BPaaS Execution Environment – Component Diagram

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.2.8 Roles

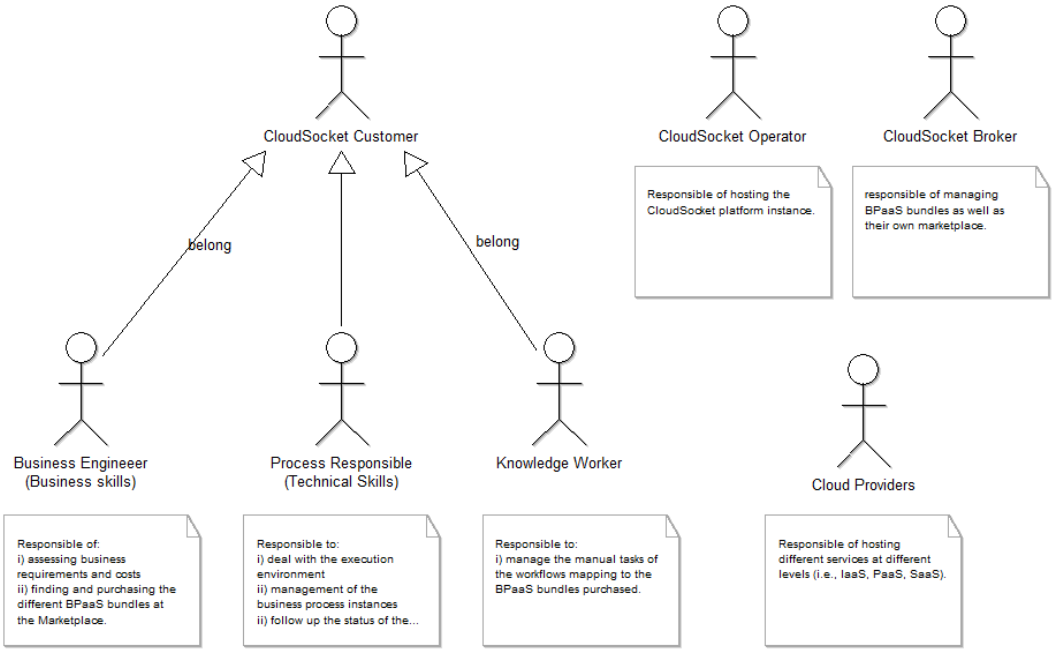


Figure 60 BPaaS Execution Environment – Actors

The following actors and roles will be involved in the BPaaS Execution Environment:

- CloudSocket Operator for hosting the CloudSocket instance
- CloudSocket Broker for managing BPaaS bundles as well as their own Marketplace (offered as service by the CloudSocket Operator).
- The CloudSocket Customer which needs to exploit a certain BPaaS. There are different roles that could be undertaken by different users of this actor. Each role covers different phases of the life-cycle management of the BPaaS, and the skills are different such as business, technical and operative, nevertheless every role can be cover by the same actor.

  One type of CloudSocket Customer role will be the Business Engineers (business skills), which are responsible of assessing business requirements and costs in order to find and purchase the different BPaaS bundles at the Marketplace. The Process Responsible (technical skills) will deal with the execution and management of the business process instances in order to follow up the status of the purchased BPaaS

  Another type of BPaaS Customer role will be the Knowledge Worker, which is responsible to manage the manual tasks of the workflows mapping to the BPaaS bundles purchased.
- The Cloud Providers offering different services at different levels (i.e., IaaS, PaaS, SaaS) enabling the creation of virtual machines to host software components as well as the direct SaaS call to realize a certain functionality, which may be required for the BPaaS.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

7.3.3 Data Interface

The Execution Environment exposes three main interfaces with the rest of the environments to manage the deployments and the SLA, besides exposing the monitored information. The following data is used to interact between the different environments.

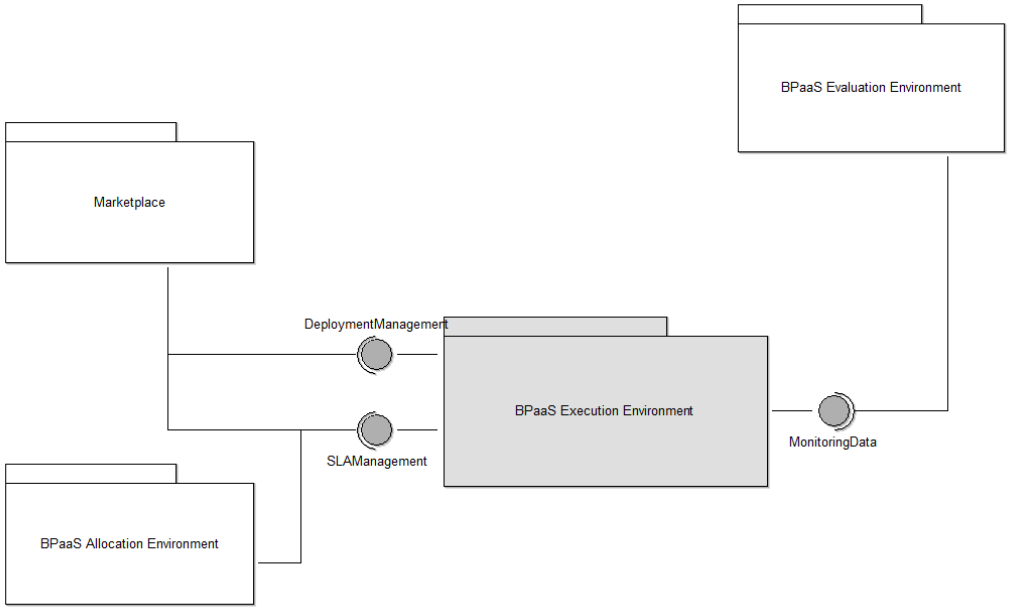


Figure 61 BPaaS Execution Environment – Interfaces

7.3.3.1 Interface to Deploy the BPaaS Bundles.

The Marketplace uses this interface to send the BPaaS bundle for deployment, which contains all the details to configure the BPaaS environment. Such data include the details of rules, adaptations, cloud providers, the software component, atomic services, the business process definition and the executable workflow. We can consider some standards to cover the different parts of the interchangeable data.

- Adaptation workflows/strategies: BPMN could be exploited
- Adaptation history: mapping of workflow instances to certain adaptation workflows that were executed for them, respective adaptation rules and reasons for adaptation - here the own format/language is deployed (e.g., defining mappings through IDs to respective information stored elsewhere)
- Adaptation rules in form of SRL (SRL 2015) as part of CAMEL (CAMEL 2015) will be used with extension to be able to specify adaptation strategies and contextual conditions: the form is the following:

  rule: event_pattern -> adaptation plan/workflow/strategy

  where rule can have an ID as well as an organisation/user owning it as well as other meta-data (e.g., date of creation). An event pattern maps to a combination of events through the use of logical or time operators, similarly to other event pattern description languages. We can have a pointer to the adaptation strategy/workflow/plan which is formally described by BPMN. If we desire not to use BPMN as adaptation plans can be quite simplistic and only sequence-based, then action specification constructs in SRL can be exploited along with some possible extensions to cover different types of adaptation actions (not only scaling but fail-over and other actions on higher-levels).
- User/Organisation information: Pointers to users and organisation information to be drawn through the SCIM (SCIM 2015) protocol/standard from the Marketplace.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

- Workflows are specified in BPMN.
- Data mappings: need to check the format/language to be used for their specification. There can be different languages depending on the task that has to be performed (e.g., map ontology concepts, XML document elements, etc.). Potential candidates are RML (RDF Mapping Language) (RML 2015), R2R (R2R 2015) and CIDOC (CIDOC 2015) mapping language. The latter has already been used for generating particular mappers which have been used in data integration tasks.
- Deployment plans CAMEL (CAMEL 2015) deployment meta-model (CloudML 2014) with the extensions for PaaS and SaaS modelling.[3]

The BPaaS Bundle of Sending Christmas Greeting Cards can be seen in the Annex.

7.3.3.2 Interface to Manage Service Level Agreements

Two potential approaches are foreseen to define the SLA specification language: a) to enrich WS-Agreement (WS-Agreement 2011) by using/referencing quality terms, such as quality metrics, defined in OWL-Q (OWL-Q 2015); b) extend OWL-Q to be able to specify SLAs.

The SLA Engine is based on the first approach relying on WS-Agreement specification (WS-Agreement 2011) and OWL-Q (which will be extended), so an agreement specification will be defined using the WS-Agreement schema (WS-Agreement 2009) and it will refer to metrics and other quality terms defined in OWL-Q (OWL-Q 2015) to complete the specification of SLOs. The main parts of the WS-Agreement schema are the following:

Figure 62 SLA Interface

A simple example of the agreement description, following this WS-agreement schema can be seen in the Annex.

7.3.3.3 Interface to publish the monitored information.

This interface semantically describes and aggregates the measures from the BPaaS Execution Environment to be further abstracted towards domain specific business process level in the BPaaS Evaluation Environment.

In OWL-Q (OWL-Q), there is a concept representing a measurement which is related to the measured value, the timestamp indicated when it was generated and to the respective metric measured. Metrics are defined in turn via respective concepts in OWL-Q in a semantically rich way. To this end, any kind of aggregated measurement which could exposed to different components via the publish/subscribe or the REST interface of the Monitoring Engine will be mapped to the description of instances of this measurement concept.

7.3.4 Research Prototypes

While it has been clearly shown that some research prototypes with respect to BPaaS monitoring and re-configuration have already been adopted by the current implementation, it is advocated that a more sophisticated

[3] See D3.3 BPaas Allocation and Execution Environment Blueprints, chapter 2.2

Copyright © 2015 BOC and other members of the CloudSocket Consortium

approach should be followed in the future which attempts to perform both types of activities in a cross-layer manner by taking into account the dependencies between the different layers as well as the possibilities of performing a series of adaptation actions, even in the context of a certain problematic situation, which span different layers. To this end, in the context of Deliverable D3.3 “BPaaS Allocation and Execution Environment Blueprints”, particular extensions to the BPaaS monitoring and adaptation research prototypes have been analysed which could be undertaken by the implementation once they are realised in full.

Concerning BPaaS monitoring, the respective prototype/blueprint is analysed in D3.3 (D3.3 2016,section 4.2.5). To summarize, the respective extensions concern the following: (a) incorporating layer-specific monitoring mechanisms/tools; (b) appropriately connecting adjacent-layer monitoring tools via a publish-subscribe mechanism in order to enable the propagation of measurements from lower to higher layers and thus be able to cover the missing gaps; (c) taking into account a hierarchical cross-layer metric model which indicates how such a propagation of measurements within the same and across layers can be performed.

Concerning BPaaS adaptation, the respective prototype/blueprint is analysed in D3.3 (D3.3 2016, section 4.3.4). The following extensions to the current prototype adopted are envisaged: (a) employing layer-specific services which implement the respective actions needed in each layer; (b) executing in a controlled and synchronized manner adaptation workflows, which include adaptation actions as service tasks in different layers, via a Workflow Engine; (c) capturing and processing of sophisticated adaptation rules which include combining via logical and temporal operators single metric events such that complex adaptation circumstances crossing different layers are properly detected; (d) feeding-up from the BPaaS Evaluation Environment of new adaptation rules in a semi-automatic manner in order to improve the adaptation behaviour of the BPaaS.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

8 BPAAS MARKETPLACE

8.1 Introduction

The foreseen Cloud Marketplace is based on the YMENS cloud marketplace, which is an online storefront through which customers may subscribe to and access native cloud applications provided by Ymens or other independent software vendors (ISVs). The Marketplace and its underlying components act as a Cloud Service Brokerage (CSB) platform acting as an intermediary between cloud providers and cloud consumer and assisting companies in choosing the services and offerings that best suits their needs.

The Marketplace, in the context of CloudSocket, has been selected as a dedicated environment to facilitate customer and provider on boarding, from customer ordering and procurement of BPaaS to provider registration of its atomic cloud services.

Its role is to link the BPaaS Allocation to the Execution Environment, giving the client the opportunity to buy and configure the BPaaS bundles received from the Allocation and to send the configured bundles to the Execution Environment for provisioning.

This Marketplace, in the context of CloudSocket, provides the following high-level features:

- BPaaS & SaaS Product Catalog
- Decision Support System for BPaaS procurement
- Customer & User Registration & On-boarding
- Identity Provisioning & Identity Lifecycle Management
- Cloud Service Provider registration of atomic cloud services
- Registry Services
- Authorization (at service level) & Authentication (at user level)

In order to support the aforementioned functionalities, the BPaaS Marketplace comprises two main components. These components are the following:

- Marketplace *(*yCONNECT*)* allows the customers to discover, analyse and purchase a BPaaS bundle in the cloud environment. Thereforem, this is the actual Marketplace which enables the customers to browse, analyze and buy the BPaaS bundles.
- Repository Manager is responsible for managing the information related to different entities, such external services, software components, cloud providers and so on. It is a transversal component, with respect to the rest of the Environments, allowing the population, browsing and searching of this information using standard web technologies.

The following major building blocks fulfil the functional capabilities of the Marketplace Environment:

i. **Customer UI Layer:** Frontend Portal used by Business User to purchase and administer services.
ii. **Product Management Layer:** Product Information System that manages product marketing data.
iii. **Identity Management System:** Provides identity federation, provisioning, authentication and authorization.
iv. **Service Orchestration Layer:** Manages service provisioning, ordering and billing.
v. **Cloud Provider Hub:** Gateway for registering and consuming atomic cloud services.
vi. **Service Repository Manager:** Set of registries for cloud services and software components.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

8.2 Functional Capabilities

The Marketplace is a software system used in production that already serves multiple customers and, as such, has objectives beyond the scope of CloudSocket. As a CloudSocket environment, the Marketplace will support its existing functionalities and will be extended to support additional scenarios for the CloudSocket objectives in BPaaS procurement.

The following scenarios were identified as part of the CloudSocket context. These scenarios are depicted using the UML nomenclature:

- Publish a BPaaS Bundle to Product Catalog
- Purchase a BPaaS Bundle
- Register New Customer and Users
- On-board Atomic Service Provides

8.2.1 Publish BPaaS Bundle to Product Catalogue

Use Case ID	**MP UC2 - Publish BPaaS Bundle to Product Catalogue**
Title & Description	A CloudSocket Broker publishes a BPaaS Bundle in the Marketplace Product Catalog. The Broker publishes a completed and hence final BPaaS Bundle in the Marketplace Product Catalog so that CloudSocket Customer can purchase, deploy and use it.
Actors	CloudSocket Broker
Use Case Objective	To publish a BPaaS Bundle into the Marketplace and enable CloudSocket Customer to buy it.
Pre-Conditions	The Bundle exists in the Bundle Repository, and it is in the Complete state.
Process Dialog	1. The Broker selects a BPaaS Bundle. 2. The Broker issues the command to publish the BPaaS Bundle in the Marketplace Product Catalog. 3. The BPaaS Allocation Environment invokes the Product API exposed by the Product Management Layer. 4. The Broker browses the bundles using the Catalog Management UI. 5. The Broker selects the published bundle and edits it description
Variations	5.a) The published Bundle is not found. The publish command is repeated.
Post-Conditions	The BPaaS Bundle is published in the Marketplace Product Catalog and the Business Process Users can find and buy it using the Marketplace.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

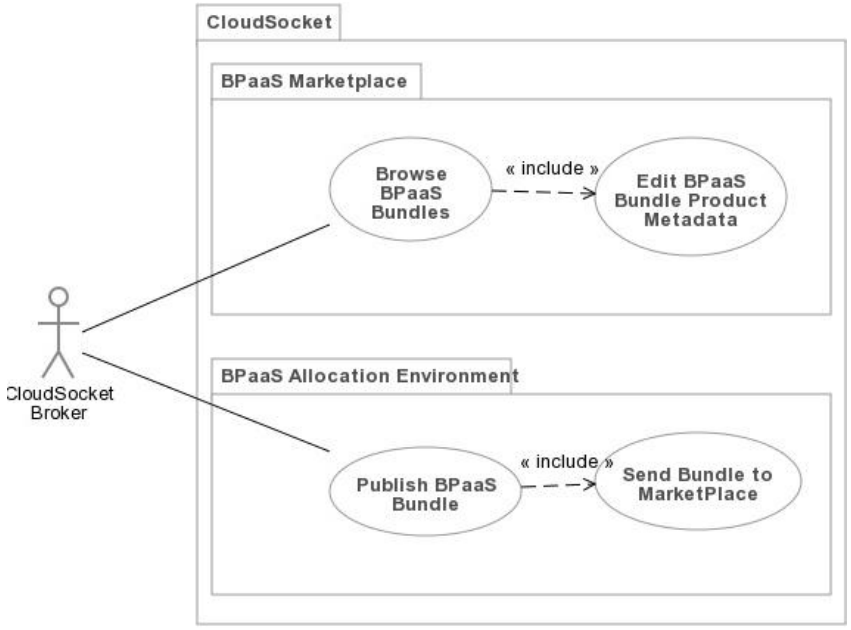
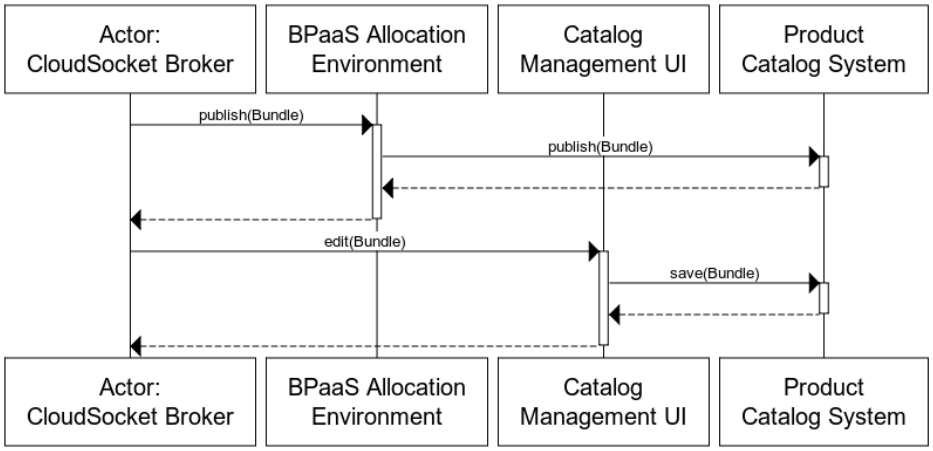
Diagrams	 **Figure 63 Use Case Diagram – MP UC2 - Publish BPaaS Bundle to Product Catalogue**  **Figure 64 Sequence Diagram – MP UC2 – Publish BPaaS Bundle to Product Catalogue**

Table 22 BPaaS Marketplace – Use Case 1 – Publish BPaaS Bundle to Product Catalogue

Copyright © 2015 BOC and other members of the CloudSocket Consortium

8.2.2 Purchase BPaaS Bundle

Use Case ID	MP-UC2-Purchase BPaaS Bundle
Description	The CloudSocket Customer browses the Product Catalog for Business Processes that fulfill his/her needs. He/she performs searches based on business criteria and is presented with BPaaS Bundles that match these criteria. After selecting a bundle, the user is presented with a guided decision wizard in which the user can further choose the most appropriate BPaaS Bundle. Making the final selection, triggers the ordering flow. Depending if the cusomter is a new or returning, the client has the chance to provision the organization he belongs to, the account and other organization user, and to add the billing & payment details. Upon final confirmation of the order, the Marketplace sends the deployment request to the BPaaS Execution Environment.
Actors	CloudSocket Customer and CloudSocket Broker
Use Case Objective	Select & Oder the most appropriate BPaaS Bundle in the environment.
Pre-Conditions	All accounts and agreements with third parties (for atomic services) and Cloud Providers (for cloud capabilities) have been previously generated and achieved, respectively. BPaaS Bundles have been defined in the BPaaS Design Environment and published to the Marketplace's Product Catalog.
Process Dialog	1. The CloudSocket Customer browses BPaaS bundles in the Marketplace. 2. The Marketplace presents to the user a decision wizard to enable choices based on business criteria. 3. The CloudSocket Customer selects the preferred BPaaS bundle according to domain specific business process properties. 4. The Marketplace collects identity information from the CloudSocket Customer and creates a new organization and account: a. Inside the Identity Management System b. With the Atomic Cloud Services associated with the BPaaS Bundle 5. The Marketplace manages or validates all the preconditions of the BPaaS. 6. If the preconditions are fulfilled, the Marketplace triggers the deployment via an Web Service call inside the BPaaS Execution Environment 7. The BPaaS bundle is deployed and available for the Business User.
Variations	If something happens along the ordering process: • The environment rolls back all the previous steps and settles all billing transactions. • The CloudSocket Broker can analyze the problem in order to resolve it and notify it to the Business Process User. During the ordering process, the different actors can see the status of the service orders so they can react on them: • The CloudSocket Customer can track the order fulfilment status and be notified upon completion.

Copyright © 2015 BOC and other members of the CloudSocket Consortium

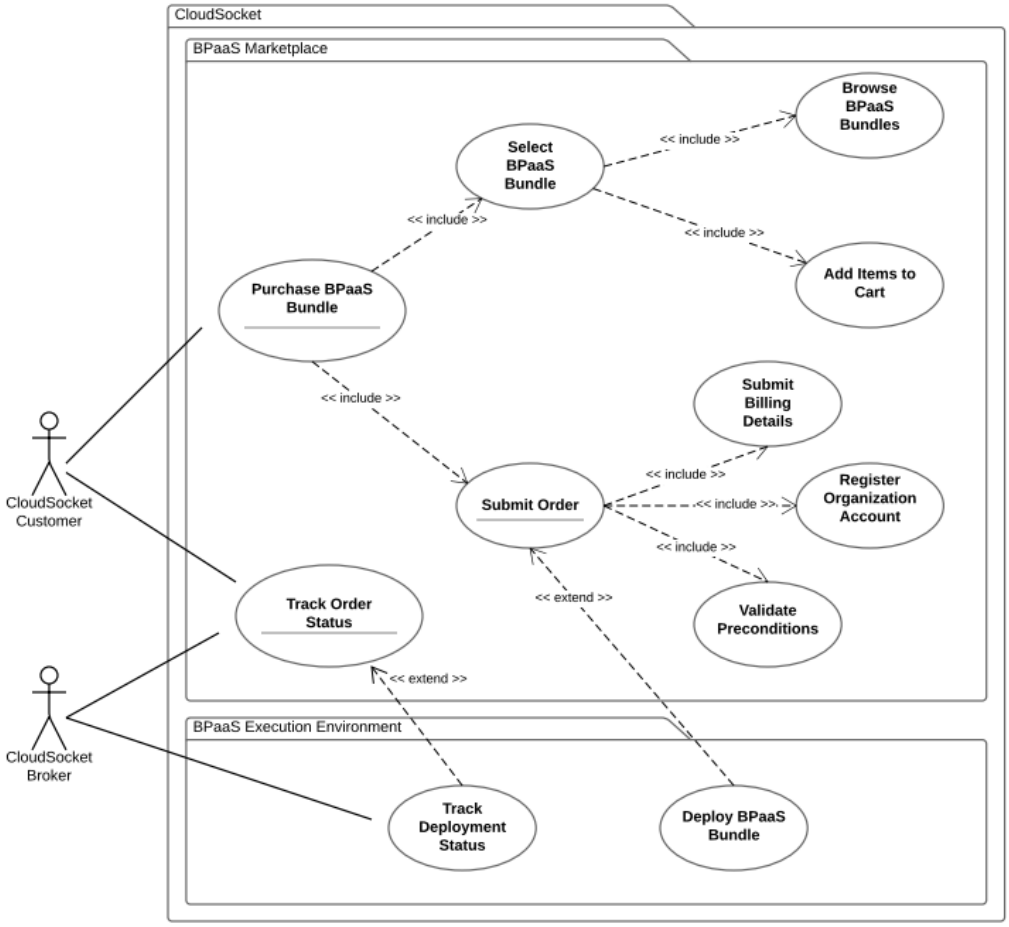
	• The CloudSocket Broker can see the history and status of all orders placed.
Post-Conditions	The BPaaS bundle deployment request is received by the execution environment.
Diagrams	

Figure 65 Use Case Diagram – MP-UC2-Purchase BPaaS Bundle

Copyright © 2015 BOC and other members of the CloudSocket Consortium

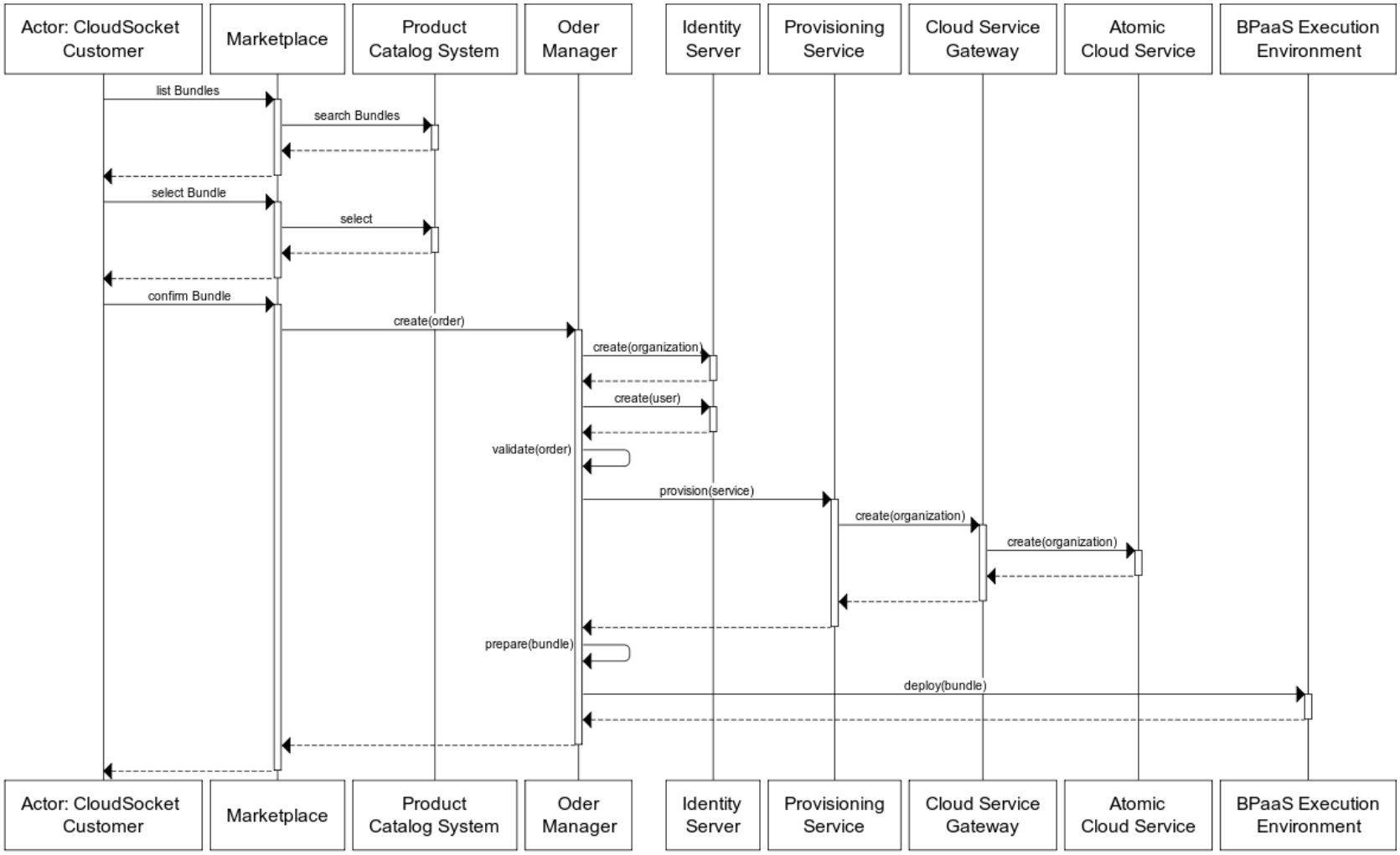


Figure 66 Sequence Diagram – MP-UC2-Purchase BPaaS Bundle

Table 23 BPaaS Marketplace – Use Case 2 - Purchase BPaaS Bundle

Copyright © 2015 BOC and other members of the CloudSocket Consortium

8.2.3 Register New Customer User

Use Case ID	**MP-UC3-Register New Customer User**
Description	The user, belonging to a CloudSocket Customer organisation, wants to create an organisational account in the Marketplace and register the organization's end-users so that he can order BPaaS bundles without the need for further registration.
Actors	CloudSocket Customer and CloudSocket Broker
Use Case Objective	Create organisation and user identity information
Pre-Conditions	N/A
Process Dialog	1. The CloudSocket Customer user accesses the Marketplace and clicks on register. 2. The user submits the identity data pertaining to the organization and the administrative account 3. The Customer Portal inside the Marketplace send the data to the Provisioning Service 4. The Provisioning Service orchestrates the data submission a. to the Marketplace Identity Server and b. if necessary to the Atomic Cloud Services 5. In case of success, an organizational account is created and the CloudSocket Customer User can add multiple end-user accounts 6. The end-user accounts are created through the same procedure via the Provisioning Services The accounts are created and confirmation emails are sent to the users.
Variations	If something happens along the identity registration process: • The environment rolls back all the previous steps and organisation/user accounts are marked for deletion. After the registration process, the different actors can see the status of organization and its users: • The CloudSocket Customer user can add/delete/update user accounts. • The CloudSocket Broker can add/delete/update user accounts and organizations.
Post-Conditions	A confirmation email is received by the Business User and.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

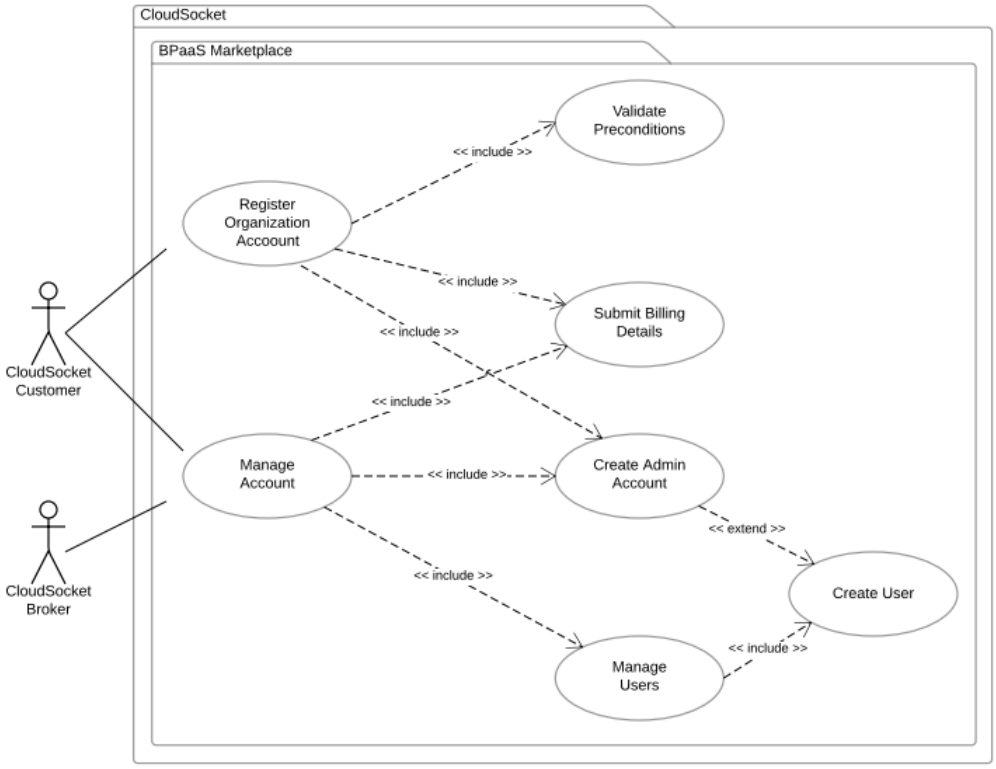
Diagrams	

Figure 67 Use Case Diagram – MP-UC3-Register New Customer User

Copyright © 2016 BOC and other members of the CloudSocket Consortium

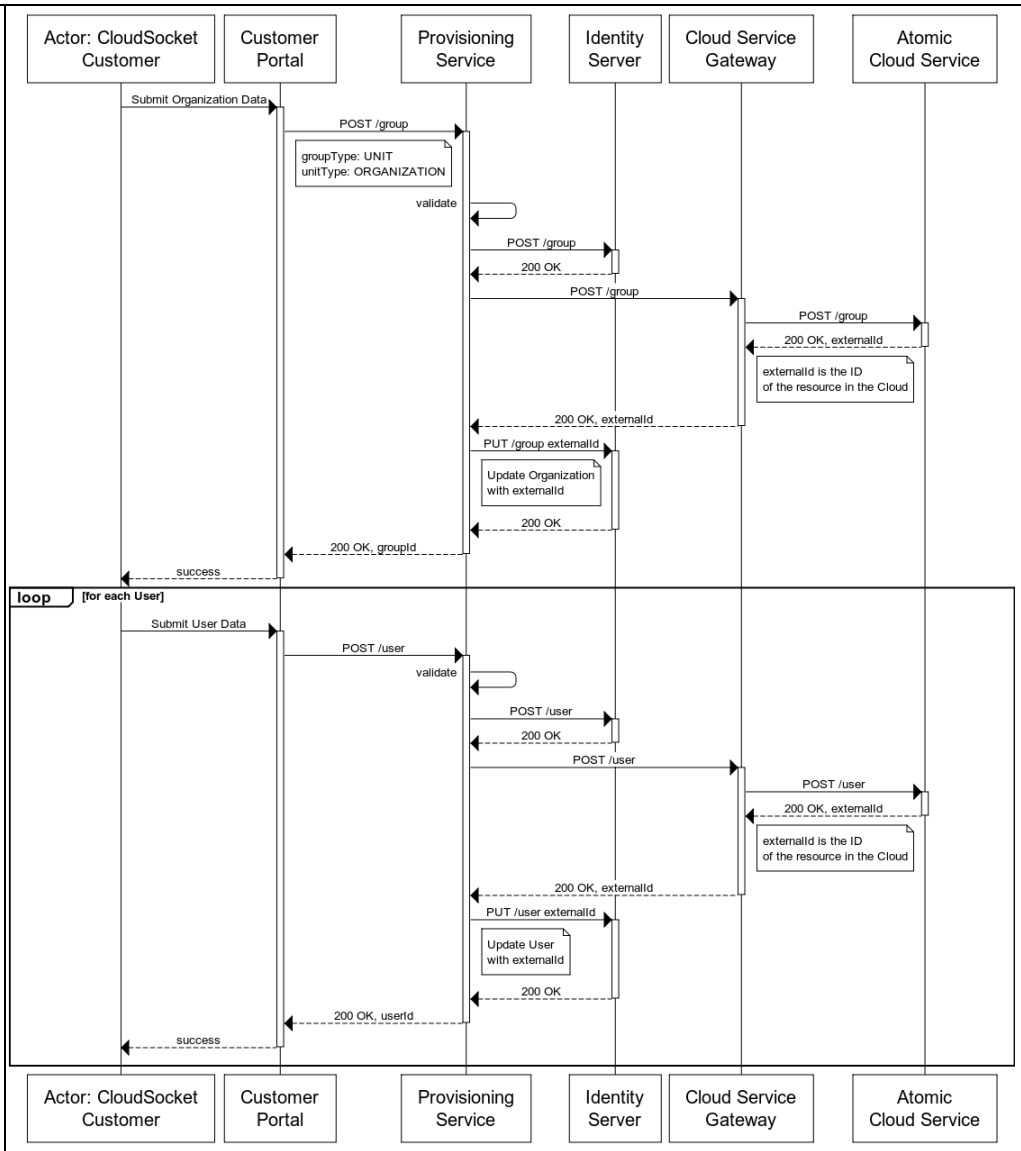


Figure 68 Sequence Diagram – MP-UC3-Register New Customer User

Table 24 BPaaS Marketplace – Use Case 3 - Register New Customer User

Copyright © 2016 BOC and other members of the CloudSocket Consortium

8.2.4 Onboard Cloud Service Provider

Use Case ID	**MP-UC-4-Onboard Cloud Service Provider**
Description	The Independent Software Vendor, providing an atomic cloud service, wants to publish its service in the Marketplace.
Actors	Service Provider
Use Case Objective	Register atomic cloud service and publish in Product Catalogue.
Pre-Conditions	A commercial agreement is reached between the CloudSocket Broker and the Service Provider.
Process Dialog	1. The Service Provider accesses the Cloud Service Hub and clicks on register. 2. The Service Provider submits service metadata: urls, service description, prices, …, 3. The registry metadata is submitted to the Service Registry 4. The product metadata is submitted to the Product Management System 5. The Service Provider tests its service for compliance. If: a. Success: The Product and Service are marked as active b. Fail: The Service Provider is notified and can update the data The Service is registered and marked as active.
Variations	If something happens along the registration process: • The environment rolls back all the previous steps and service metadata is marked for deletion After the registration process, the different actors can see the status of organization and its users: • The Service Provider can update the service and product metadata. • The CloudSocket Broker update/delete the service and product metadata.
Post-Conditions	The registered service is tested successfully for compliance

Copyright © 2016 BOC and other members of the CloudSocket Consortium

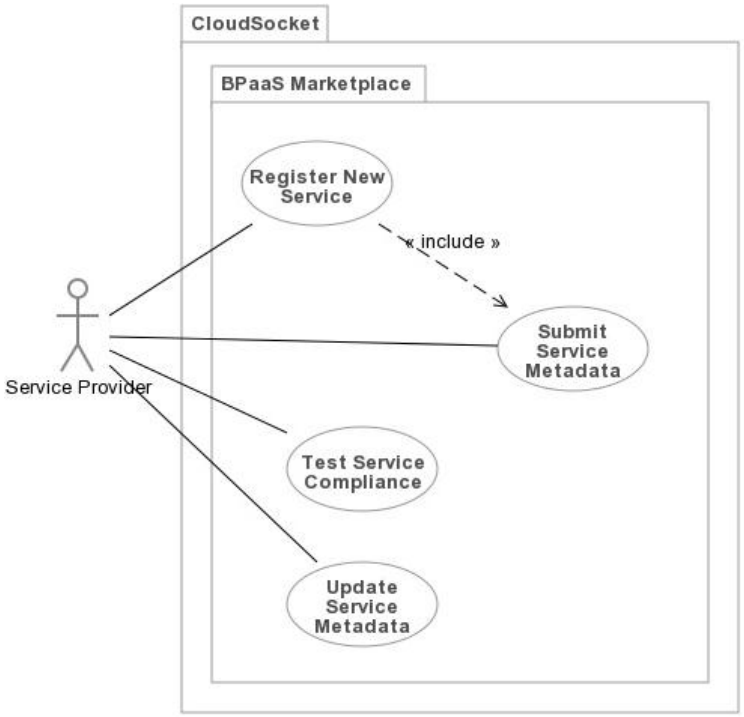
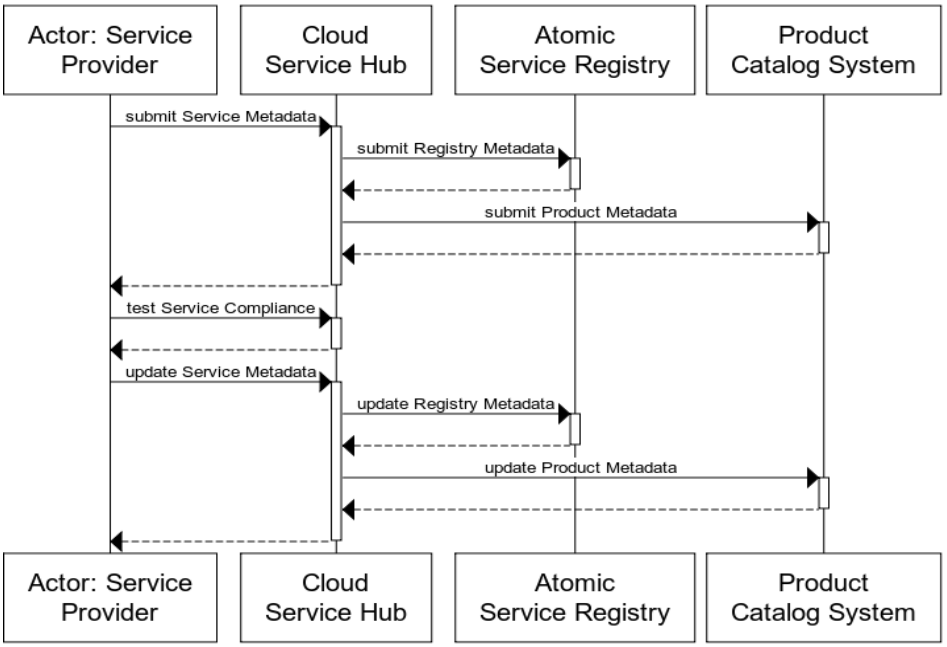
Diagrams	 **Figure 69 Use Case Diagram – MP-UC-4-Onboard Cloud Service Provider**  **Figure 70 Sequence Diagram – MP-UC-4-Onboard Cloud Service Provider**

Table 25 BPaaS Marketplace – Use Case 4 – Onboard Cloud Service Provider

Copyright © 2016 BOC and other members of the CloudSocket Consortium

8.3 Components

The Marketplace architecture supports multitier topologies in order to decouple the user interface and the business logic core. Nevertheless it is responsibility of every component to apply more tiers or patterns to create a loosely coupled design. The following sections the major components in the Marketplace and the interaction with the other BPaaS Environments.

8.3.1 Customer UI Layer

8.3.1.1 Marketplace

This frontend allows the users to browse, order, provision, and manage their resources. This Portal is the user's single pane of glass to remotely browse a storefront catalogue of cloud services and provision desired services. It has a **catalogue** of cloud services that users can browse (storefront), an e-commerce **Shopping Cart** integrated with PayPal and other Payment Gateways and an account registration module to support new customer onboarding.

Decision Wizard: Part of the Marketplace, the Decision Engine helps users automatically find the appropriate BPaaS bundle for their needs from among an extensive offer. Is will be a wizard like interface to enable service selection based on location, compliance, cost, SLAs and other domain specific business process criteria.

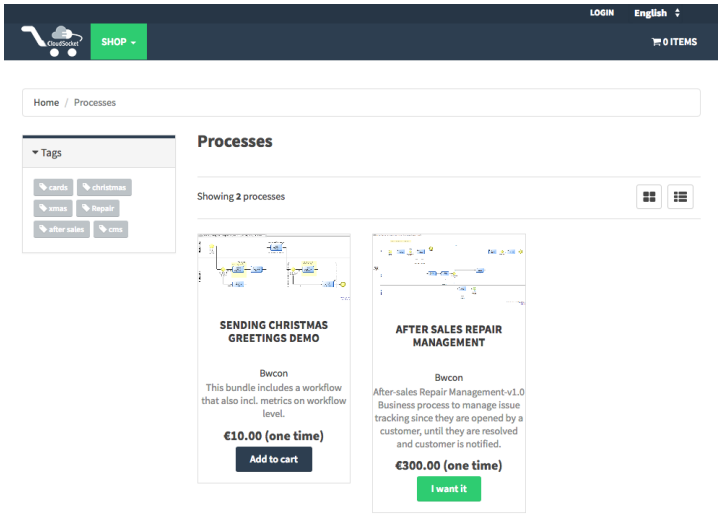


Figure 71 - Marketplace web page.

8.3.1.2 Customer Portal

The Customer Portal provides a single point of entry for CloudSocket Customer to login and manage system accounts, resources, access control and purchased cloud services (through the **User Management UI**). End-users are able to access their entitled services in a seamless manner, with web SSO, via the **Cloud Portal** and received feedback and statistics about service usage via the pre-configured **Dashboard(s).**

Copyright © 2016 BOC and other members of the CloudSocket Consortium

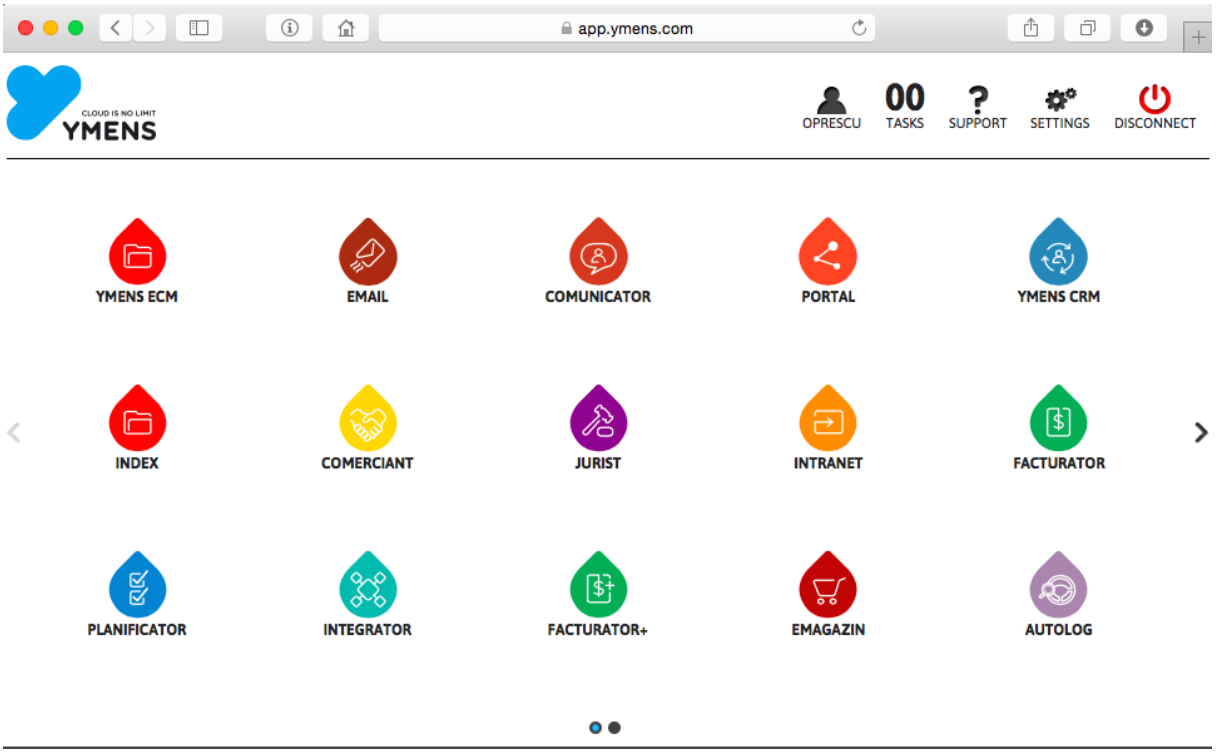


Figure 72 Marketplace – Customer Portal – User Interface Mockup

8.3.2 Cloud Broker Engine

The Cloud Broker Engine provides the core cloud automation capabilities and a business rules management. It ensures catalogue management, order orchestration, service provisioning and identity synchronization in an authorized manner.

Product Management Layer: The Product Information System manages product-marketing data. It is the core sales component that transforms a set of cloud services into products and bundles, with prices, discount and promotions, with sales description, product sheet and variants.

Decision Support System: Customized rule engine that supports the service selection process, with customer defined criteria that are linked to business requirements like cost, location, data protection level, etc.. The criteria and the decision matrix are extendable and customizable.

Service Orchestration Layer: The orchestrator supports managing service provisioning, ordering and billing and assures the fulfilment of cloud services provision.

8.3.3 Identity Management System

Identity Manager is the authentication component responsible for verifying the identity of a principal requesting authorization, as well as group membership inquiries. It ensures interoperability and Cross-Domain SSO by relying on standards such as SAML (SAML 2015), OpenID Connect (OIDC 2015) and SCIM (SCIM 2015). It ensures identity propagation and just-in-time provisioning. The Access Manager is responsible for authorization and access policy management, which serves as the basis for the decisions of the Policy Manager that either grants or denies the requests for capabilities.

The Identity Management System (IdM) will provide authentication services for all the components of the Cloud Socket platform. This means it will store the following types of user accounts:

Copyright © 2016 BOC and other members of the CloudSocket Consortium

1. User accounts for securing the integration between different components of the platform (Design, Allocation, MarketPlace, Execution)
2. User accounts for administrative tasks for each component of the platform (monitoring, administrative UI, and so on)
3. User accounts belonging to Cloud Brokers, Cloud Providers, Cloud Consumers (Customers) in order to use the platform

The Identity Management System will include an API, available to the other components of the Cloud Socket platform, for user/roles provisioning.

The standard deployment of Identity Management System will include the following default roles:

- Design Environment
  - design.broker.admin – has access to administrative functions of Design Environment from the Broker perspective for example to send BPaaS to Allocation Environment)
  - design.admin – has access to administrative functions of Design Environment
- Allocation Environment
  - alloc. broker.admin - has access to administrative functions of Design Environment from the Broker perspective for example to send BPaaS to Marketplace Environment)
  - alloc.admin - has access to administrative functions of Design Environment
- Marketplace Environment
  - mkp.customer.admin – has access to administrative functions of Marketplace – acting as customer procurement
  - mkp.provider.admin – has access to administrative functions of Marketplace – acting as provider service manager
  - mkp.broker.admin – has access to administrative functions of Marketplace – acting as broker of a specific BPaaS
  - mkp.admin – has to administrative functions of Marketplace like: user management for brokers and providers, authorization for customer_admins, manage the special offers and so on
- Execution Environment
  - exec.customer.admin – has access to users management for the bought BPaaS
  - exec.provider.admin – has access to the service usage dashboards
  - exec.admin – manage the access for the Execution Environment Components
  - exec.customer.user – belongs to the customer and interacts with the execution of the BPaaS bundles.
- Evaluation Environment
  - eval.broker.admin – has access to administrative functions of Evaluation Environment from the Broker perspective
  - eval.admin – has access to administrative functions of Evaluation Environment

Besides the default roles, each environment might manage through API the IdM in order to provision additional needed roles specific for their environments. For example, the specific user roles required by the execution of workflow should be deployed in IdM along with the BPaaS bundle deployment (exec.bpass_id.role1). In this context the mapping of users with specific BPaaS bundle roles will be managed by Workflow component of Execution Environment through IdM API.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

The supported functionalities of the Identify Management System are as follows:

- Identity Management (IM)
  - General Identity Management
  - Federated Identity Management (FIM)
- Authentication
  - General Authentication
  - Single Sign-On (SSO)
- Authorization
  - Account and Attribute Management
  - Account and Attribute Provisioning

Supported use cases include:

Title	Description
Identity Provisioning	Feature the need support and manage customer policies for identity decommissioning including transitioning of affected resources to new identities.
Identity Configuration	Feature the need for portable standards to configure identities in cloud applications and infrastructure (virtual machines, servers etc.).
Organization/Customer Administration	Feature the ability for enterprises to securely manage their use of the cloud provider's services (whether IaaS, PaaS or SaaS), and further meet their compliance requirements. Administrator users are authenticated at the appropriate assurance level.
Access to Enterprise Cloud Hosted Applications	Exhibit the need for seamless authentication and access privileges conveyance from an enterprise that is wishes to host their workforce applications on a public cloud.
Cloud Identity Management, SSO and Authentication	A user (or cloud consumer) is able to access multiple SaaS applications using a single identity.
Federated User Provisioning and Management	Show the need for provisioning, administration and governance of user identities and their attributes for organizations that have a distributed structure which includes many central, branch offices and business partners where each may utilize cloud deployment models.
Federated SSO and Attribute Sharing	Feature the need for Federated Single Sign-On (F-SSO) across multiple cloud environments
Enterprise to Cloud SSO	A user is able to access resource within their enterprise environment or within a cloud deployment using a single identity. Users expect and need to have their enterprise identity extend to the cloud and used to obtain different services from different providers. By accessing services via a federated enterprise identity, not only the user experience of SSO is to gain, but also Enterprise compliance and for control of user access
Offload Identity Management to External Business Entity	Show the need for federated identity management which enables an enterprise to make available cloud-hosted applications to either the employees of its customers & partners or its own institutional consumers and avoid directly managing identities for those users
Per Customer Identity Provider Configuration	Show the need for cloud tenants to securely manage cloud services using automated tools rather than navigating and manually configuring each service individually
Delegated Identity Provider Configuration	Show the need for cloud tenant administrators need to delegate access to their identity services configuration within a multi-tenant cloud service to their chosen IdP service

Table 26 Identify Management Use Cases

Copyright © 2016 BOC and other members of the CloudSocket Consortium

8.3.4 Cloud Provider Hub

This component represents the gateway for registering and consuming atomic cloud services. The integration of new Service Providers provides the following functionalities:

- Provisioning new instances of the service / application: purchase, modification or cancellation.
- Provisioning users who can make use of the application / service.
- Validation of user credentials in order to use application / service contracted.

Provider Management UI: The aim of this portal is to enforce the integration of every Service Provider to become a part of the Marketplace. It manages Service Provider integration and lifecycle.

Cloud Service Gateway: It enables atomic cloud services routing proxy and adaption layer that supports provisioning and SSO.

8.3.5 Repository Manager

This component offers access to the repositories and therefore to the meta-data of the services. Different options to manage the access are considered to the different registries (a) someone may indicate the repository/registry apart from the query itself and then let the Repository Manager to perform the respective call and return back the result from the registry, (b) an API is provided by the Repository Manager for each repository/registry - in this case, there is a need of calling the API methods of the corresponding registry with the respective query needed and, (c) direct access to repositories - in this case, the endpoint of the repositories has to be known in order to be able to execute a particular query.

Based on the following requirements, we consider the two first options in order to be more modular and flexible:

a) there is a need for a component to manage these repositories;
b) the access API can be uniform to hide specifies of the technologies used to realize these repositories - otherwise, different forms of queries will need to be run for different repositories in possibly different query languages which might be overwhelming for components which need to interact with more than one registry in the end;
c) this is needed for accounting reasons - otherwise, there should be a sophisticated mechanism which detects who is performing which query on what repository in order to proceed with the respective charge or to check if the access is allowed due to non-payment reasons (component owner did not pay this month to access the repository service).

Hence the Repository Manager is characterised by:

- Tasks: Handles access to the repositories;
- Main Interfaces:
    - Input: Endpoint of repositories (data to be collected) and credentials
    - Output: Information retrieved through calling the respective access method of the Repository Manager API.

The following three registries are part of the Repository Manager.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

8.3.5.1 Atomic Service Registry

The Atomic Service Registry describes the cloud services that a **Service Provider** offers. There are three essential components to the Service Catalogue:

- The data model (in a self-describing JSON format) holding the information for the Service Catalogue to describe standard and extended attributes of **Service Providers** and service offerings.
- The standard set of structures to describe the **Service Provider** and **Service Provider's** products and services.
- The API to interact with the Service Catalogue.

The different workflows use them to be part of the service tasks in the business process definition. Hence, these descriptions have to include both functional descriptions, as business details and technical specification (as the interface description). Besides, this is required certainly in case of adaptation reasons, as we might need to substitute one service with another one. Additionally, it can also be used to draw additional information about a particular service, which could then facilitate its (adaptive) execution.

8.3.5.2 Software Component Registry

This is required in the context of a BPaaS realization, deployment and adaptation as we need to know how to obtain a software component and deploy/run it in a particular VM. Such information can be drawn on demand (based on the component identifier and other references in the BPaaS bundle) when the respective task (deployment/adaptation) has to be performed. As in the case of the atomic service, the business process needs to know details about the functional descriptions, the business details and the technical specifications (as the interface description) in order to include and use them in the design of the BPaaS.

8.3.5.3 Cloud Provider Registry

This component will be responsible to store and describe the different cloud provider configuration (local or remotes) as the login, front-end, the definition of the APIs. This registry is not just a simple configuration description, it is also related with the complete management information; only the cloud providers included in the registry will be able to be used in the BPaaS bundle. This information will be use in the execution and the allocation phase.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

8.3.5.4 Component Diagram

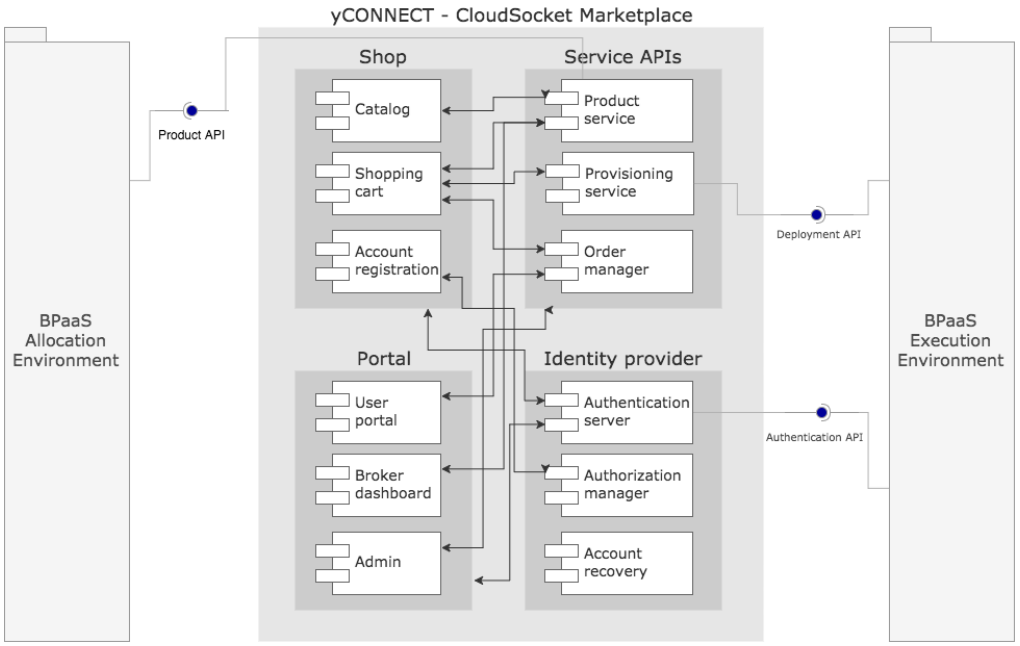


Figure 73 BPaaS Marketplace Components

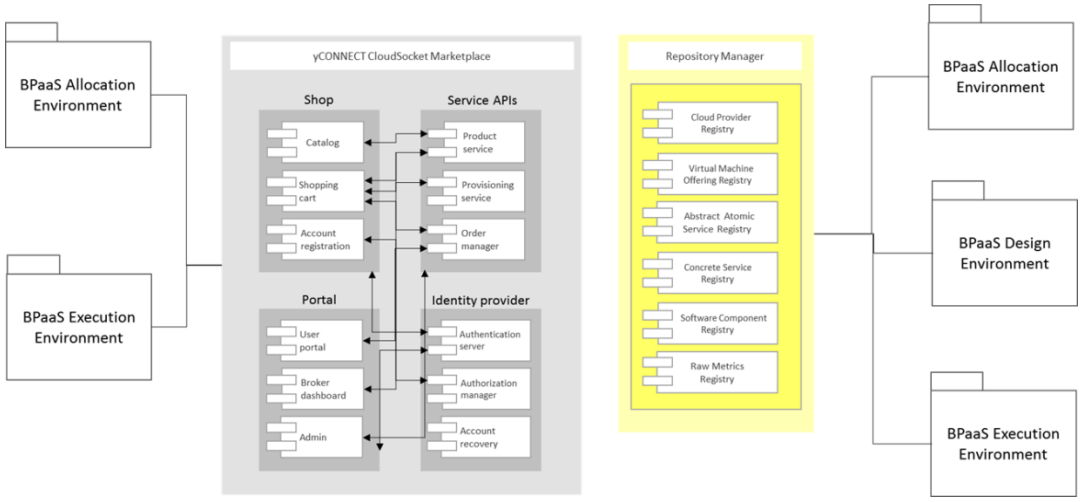


Figure 74 BPaaS Marketplace Component Interaction

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Deploy Diagram

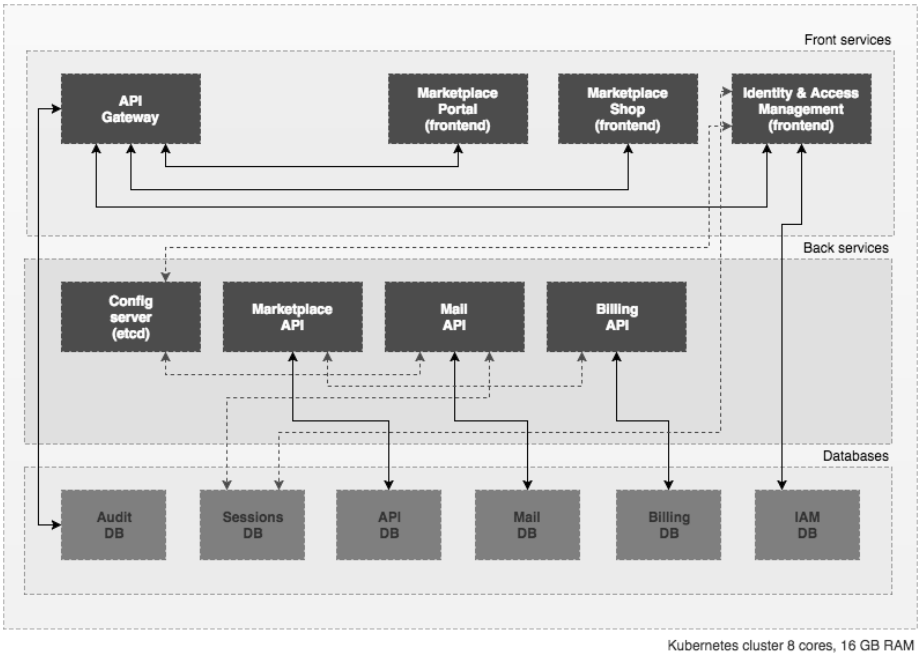


Figure 75 - Deploy diagram of the Marketplace

8.4 Roles

The main CloudSocket roles that interact with the Marketplace are:

i. **CloudSocket Operator** who is for providing and maintaining the environment to manage BPaaSs
ii. **CloudSocket Broker** who is responsible for designing and publishing BPaaS bundles in the CloudSocket
iii. **CloudSocket Customer**, a user of an end-user organisation which is assigned with the task to find, purchase and manage the different BPaaS bundles to be used by his organisation
iv. **Service Provider** who wants to make its atomic cloud services available in the Marketplace and orchestratable via BPaaS.

8.5 Data Interface

The interfaces supported by the Marketplace environment are REST based interfaces, working over http, derived from the SCIM protocol and extend to support additional functionalities. They support both JSON and XML data format and are extensible so that enrichment with metadata information is enabled for every operation. These are as follows:

- **Identity API:** SCIM extension that deals with user identity provisioning and management
- **Auth API:** OpenID Connect 1.0 single sign-on implementation (REST, HTTP based)
- **Product API:** SCIM inspired REST protocol that extends the concept of Resource to **Product** and manages product lifecycle
- **Registry API:** SCIM inspired REST protocol that extends the concept of Resource to Service and deals with Atomic Cloud Service registration and management
- **Cloud API:** set of REST APIs that support identity provisioning (SCIM inspired), SSO and service adaption for external cloud services

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9 EVALUATION ENVIRONMENT

9.1 Introduction

The BPaaS Evaluation Environment has the main aim to perform an analysis over the information produced during the execution phase of the BPaaS lifecycle in order to discover bottlenecks as well as optimisation points. Such information can then be exploited in a next cycle by the BPaaS Design and Allocation Environments in order to optimize the BPaaS such that it is able to exhibit a particular service level as well as improve the business benefits of the CloudSocket Broker offering it.

In order to support such evaluation, the BPaaS Evaluation Environment follows, similarly to the previous environments, a multi-layer architecture which comprises three main layers: (a) the UI layer which visualizes the results of the analysis; (b) the business logic layer which offers the analysis functionality to the layer above, the UI one; (c) the data layer which provides all information that is required in order to support the analysis.

In each layer, different components have been put in place. At the UI layer, we have the Hybrid Business Process Dashboard which visualises KPI evaluations as well as other findings from the analysis, such as business process bottlenecks. At the business logic layer, there is: (a) the Hybrid Business Process Management Tool which is responsible for the evaluation of KPIs; (b) the Conceptual Analytics Engine which is responsible for performing analysis over the underlying data in order to assist in the evaluation of the KPIs as well as provide useful design and allocation suggestions in the form of best deployments and adaptation patterns & rules; (c) the Business Process Miner which mines the execution history of the business process of the BPaaS in order to discover bottlenecks as well as discrepancies between what has been designed and what has been in place during production.

One main actor is involved in the Evaluation Environment and this is the CloudSocket Broker. Two main roles are foreseen in the broker's organisation to participate in the BPaaS evaluation phase, which maps to a business and a technical role. The business role is responsible for the evaluation at the domain specific business process level while the technical role is responsible for the evaluation at the workflow level. The cooperation between the roles is supported by the BPaaS Evaluation Environment as it enables the smooth switch between business and workflow levels in a controlled way. It has to be noted that the results of the BPaaS Evaluation Environment can be exploited by the same roles during the design and allocation phase of the BPaaS. In this sense, the CloudSocket will support the alternation between the different environments in order to enable the rapid optimisation of the BPaaS based on the findings of the evaluation phase. This means that the evaluation results could be directly exploited by the tools of the BPaaS Design Environment in order to load the respective artefacts and immediately apply on them in the BPaaS Allocation Environement the provided optimisation suggestions, once of course the respective roles agree on them.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.2 Functional Capabilities

The following main functional capabilities are envisaged to be exposed by the BPaaS Evaluation Environment:

- Aggregation and decomposition from technical logs to domain specific business KPIs.
- Conceptual Analytics providing results which enable root cause analysis, drill down of business indicators and propagation of technical indicators
- Discovery of best deployments for BPaaSs
- Derivation of adaptation rules or event patterns leading to critical events, such as violation of KPIs and SLOs. The latter can be transformed into adaptation rules to support pro-active BPaaS adaptation in a semi-automatic manner.
- Process mining analysis to support the discovery of bottlenecks and places for improvement inside business processes and corresponding workflows.

To support and highlight the benefits of the aforementioned main functionality, we have the following visualisation and data building functionalities:

- Bottom up mapping of process, service and monitoring logs with semantic descriptions
- Presentation of a monitoring dashboard that enables a management overview of the business processes that run in the cloud.

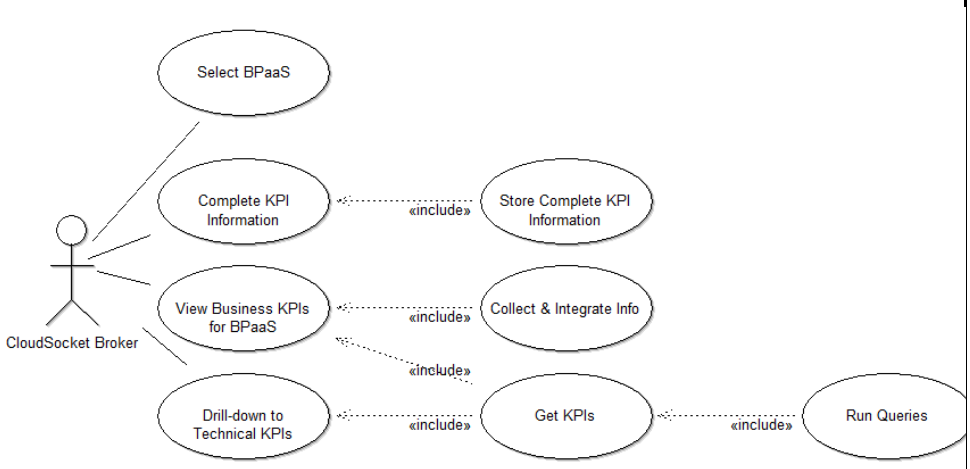
This high-level functionality is manifested through the following scenarios which are captured by use case description in conjunction with certain types of UML diagrams.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.2.1 KPI Analysis & Visualisation

Use Case id	EvE-UC-1-KPI Analysis and Visualisation
Title & Description	While the BPaaS execution phase focuses on the adaptive provisioning of a BPaaS, the aim of the evaluation phase is to discover those optimisation points which will enable to optimise a BPaaS in order to better fulfil the business and technical requirements posed to it as well as decrease costs and increase the business assets of the Cloud Socket Broker. One form of optimisation checking is through the use of KPIs which span the business and technical/workflow level. Such KPIs indicate how well a BPaaS behaves according to particular functional and quality aspects. In this respect, their evaluation can show whether the BPaaS behaviour is suitable or has to be modified in order to better support the requirements posed to it. To support the KPI evaluation, the BPaaS Evaluation Environment collects information from the BPaaS Execution Environment spanning process, service and monitoring logs and links it together in a semantic manner, building in this way a semantic repository. This semantic repository can then be exploited in order to derive new knowledge and pose queries on it which could lead to the evaluation of KPIs. The main actor in the BPaaS Evaluation Environment is the CloudSocket Broker, who would exploit the Hybrid Business Dashboard in order to select a BPaaS and its respective business process. Once this is done, then the dashboard should, by performing particular queries over the semantic repository, present to the broker the high-level business KPIs in different colours indicating which KPIs are met and which have been violated. The CloudSocket Broker then can select a violated business KPI, see how much it has been violated and request a drill-down into more related technical KPIs which will then be visualised, in the same manner, by the dashboard. In the background, a set of queries will be performed over the semantic repository in order to draw the information of the technical KPIs. Through this drill-down, the broker has the opportunity to discover what exactly the root cause of a KPI violation is and then act upon it.
Actors	CloudSocket Broker
Use Case Objective	Top-down browsing of KPIs with semantic analysis and querying at the background
Pre-Conditions	The BPaaS has been already purchased and be executed in the context of one or more users. Otherwise, it is not meaningful to provide any evaluation result as the BPaaS would not have been used. The existence of a BPaaS Execution Environment which produces the appropriate information that is used to build up the semantic repository. This environment should also provide an API (REST-API) through which this information can be retrieved. Business and technical KPIs have been specified for the BPaaS in the previous phases, indicating what they represent and what are their main components (target value, metric and evaluation period). Of course, there is the flexibility to describe the KPI in the context of the BPaaS Evaluation Environment. In this case, the business and technical roles are supported to complete the specification of KPIs and then they will be able to see the results of the respective semantic analysis.
Process	• The broker selects a particular BPaaS/business processes from those offered. In

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Dialog	particular, he is able to see a list of business processes along with any indicative visual element which shows whether there is a violation of one or more KPIs for each business process. In this manner, he/she will be more supported as he/she can select only to check interesting BPaaSs (e.g., having KPIs violated). • The CloudSocket Broker is prompted or decides to complete the information of some KPIs defined in the previous phases in order to be able to see assessment results also for them. • The CloudSocket Broker is able to see a high-level overview of the business process with a set of business KPIs coupled with their current status (i.e., respected or violated). • The CloudSocket Broker can select a KPI and see how well it is satisfied or how much it has been violated. • The broker can drill-down from a business KPI to a set of technical KPIs from which the business KPI is derived in order to enable him/her to understand what the root cause of a business KPI violation is. • For each KPI assessment requesting step, the Hybrid Business Dashboard perform requests on the Conceptual Analytics Engine, which performs the necessary computations for KPIs whose definitions have been completed during the current user session and the appropriate queries in order to obtain the respective values of KPIs already computed. This information is then sent back to the dashboard in order to visualize it by a so-called "business process model assimilation" technique in order to show the CloudSocket Broker the results within the business process model.
Variations	• If KPIs are already complete, then there is no need to complete their information. This can be checked by processing all KPIs posed for a particular BPaaS • KPI completion can be performed even after the overview for a BPaaS is graphically represented. In this case, the KPI would be shown with an indication that it cannot be evaluated. As such, the broker, if he/she desires to assess this KPI, will have to complete its information.
Post-Conditions	Measurements and respective evaluations for KPIs are produced. For some KPIs, there can be a completion of their specification.
Diagrams	 **Figure 76 Use Case Diagram – EvE-UC-1-KPI Analysis and Visualisation**

Copyright © 2016 BOC and other members of the CloudSocket Consortium

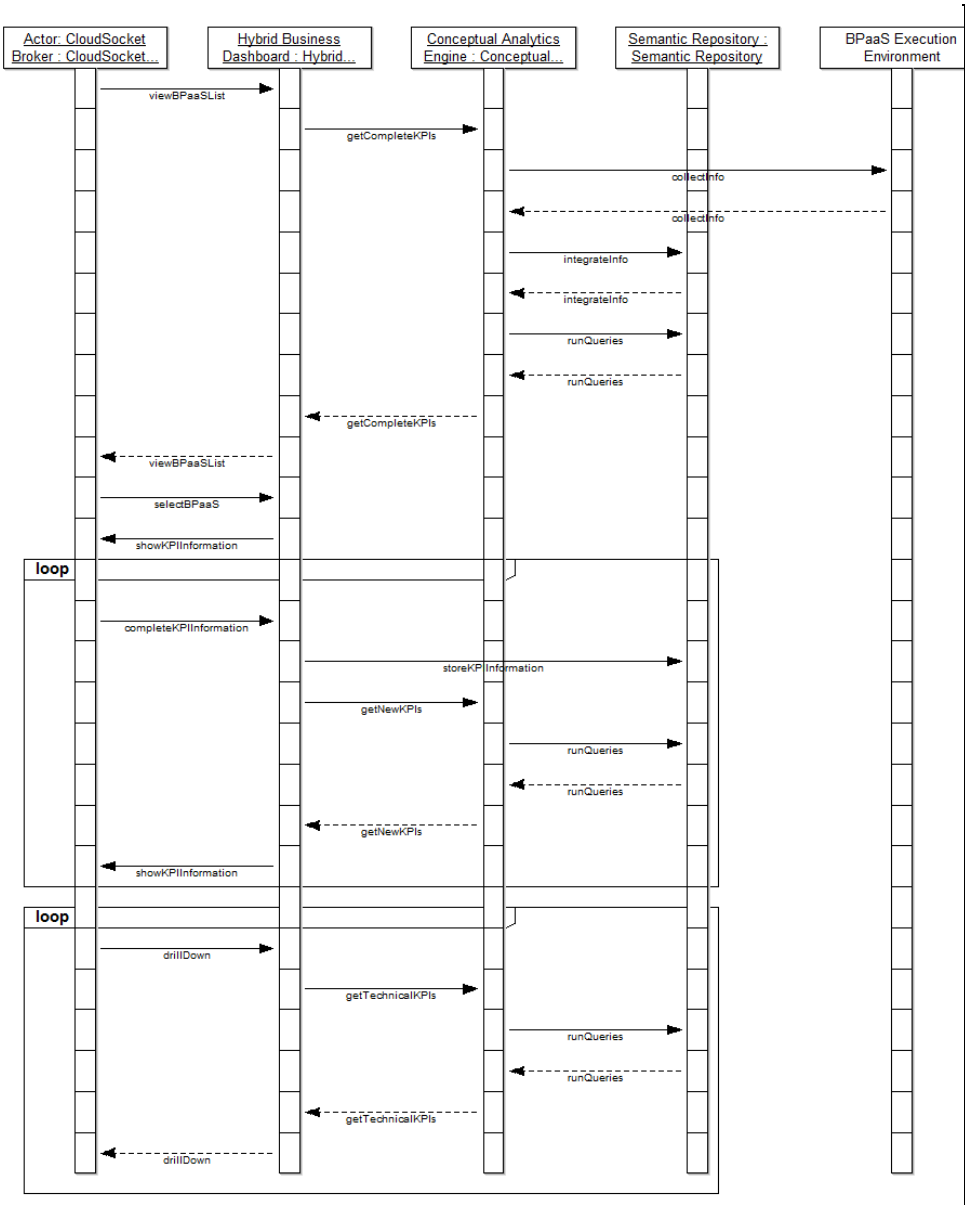


Figure 77 Sequence Diagram – EvE-UC-1-KPI Analysis and Visualisation

Table 27 BPaaS Evaluation Environment – Use Case 1 - KPI Analysis and Visualisation

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.2.2 Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules

Use Case id	EvE-UC-2-Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules
Title & Description	The goal of this use case is to enable the generation of particular knowledge in the form of best deployments and adaptation rules for BPaaS which can be used as suggestions for improvement in the BPaaS Design and Allocation Environments. Such knowledge can only be produced if the respective data are available in the semantic repository. Such data include process, service and monitoring logs through which we can for instance know for which process structure and respective deployment, we have the best possible performance for a set of KPIs. The CloudSocket Broker can exploit best deployment hints during the allocation phase in order to modify the deployment plan of a BPaaS based on the respective execution history of this BPaaS or similar BPaaSs. Of course, such deployment hints are just suggestions - the CloudSocket Broker should have the appropriate expertise in order to follow or modify the respective hint such that it suits a certain BPaaS. The CloudSocket Broker can also exploit adaptation patterns/rules in order to incorporate them as they are in a BPaaS bundle or modify them based on his/her expertise. Depending on the form of the adaptation suggestion, the respective modification effort could be different. If deployment rules are suggested, then the broker can just inspect them and make slight modifications. If event patterns leading to violation of KPIs/SLOs are suggested, then the broker should be able to complete the remaining part of the adaptation rule pertaining to the specification of an adaptation strategy. The CloudSocket Broker will exploit the Hybrid Business Dashboard in order to select the respective business process/BPaaS and perform an analysis on it. Then, he/she can see the analysis results and export them in the appropriate form in order to exploit them in the next cycle of the BPaaS.
Actors	CloudSocket Broker
Use Case Objective	Derivation & visualisation of deployment and adaptation suggestions
Pre-Conditions	The BPaaS and/or similar BPaaSs have been already purchased and have been executed in the context of one or more users. To obtain deployment suggestions, there is the need that at least similar BPaaS have already been executed. To obtain adaptation suggestions, there is the need that at least the BPaaS has been executed. The existence of an BPaaS Execution Environment which produces the appropriate information that is used to build up the semantic repository on which the analysis takes place. This environment should also provide an API (REST-API) through which such information can be retrieved. Obviously, there should be some violations for KPIs/SLOs in order to be able to derive adaptation event patterns or rules. Similarly, for a BPaaS, there should obviously be not one but many deployments. If this BPaaS is similar to other BPaaSs, then at least one deployment per similar BPaaS in order

Copyright © 2016 BOC and other members of the CloudSocket Consortium

	to have a meaningful comparison which will lead to a suitable result is required. As the best deployment analysis considers a set of KPIs, then obviously the KPIs of this set should be fully specified.
Process Dialog	• The CloudSocket Broker selects a particular business process/BPaaS from those offered. He might select an interesting BPaaS which has critical or important KPIs. • The CloudSocket Broker is prompted or decides to complete the information of some KPIs defined in the previous phases in order to be able to perform the analysis over the set of KPIs desired. • The CloudSocket Broker selects to perform deployment or adaptation analysis: ○ The user selection is forwarded in the form of a respective request to the Conceptual Analytics Engine. This engine might have to first to adapt the execution history of the BPaaS (and possibly also for similar BPaaSs) in case that one or more KPIs have their specification completed during the current user session. Then, it executes the respective analysis algorithm(s) based on the user selection over the semantic repository which should have been properly set up by drawing the appropriate information from the Execution Environment. • Depending on the CloudSocket Broker's choice, the dashboard shows the analysis results. • In case of adaptation analysis, the CloudSocket Broker can select incomplete rules in the form of event patterns (leading to SLO/KPI violations) which can then be finalized through the mapping of these patterns to specific adaptation actions or strategies. If strategies are represented by BPMN workflows, then the broker should be presented with a BPMN design tool to perform the adaptation workflow specification. • The CloudSocket Broker selects some (automatically generated or manually completed) analysis results and exports them in the suitable format for further processing in the next cycle.
Variations	• If KPIs are already complete, then there is no need to complete their information. This can be checked by processing all KPIs posed for a particular BPaaS • We could by default perform all sorts of analysis without requiring from the broker to select the one sort or the other. • The option to complete adaptation rules can be communicated to the users working with the BPaaS Allocation Environment.
Post-Conditions	Adaptation and/or deployment suggestions are derived. They might also be exported in case that the broker selects this option.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

Diagrams

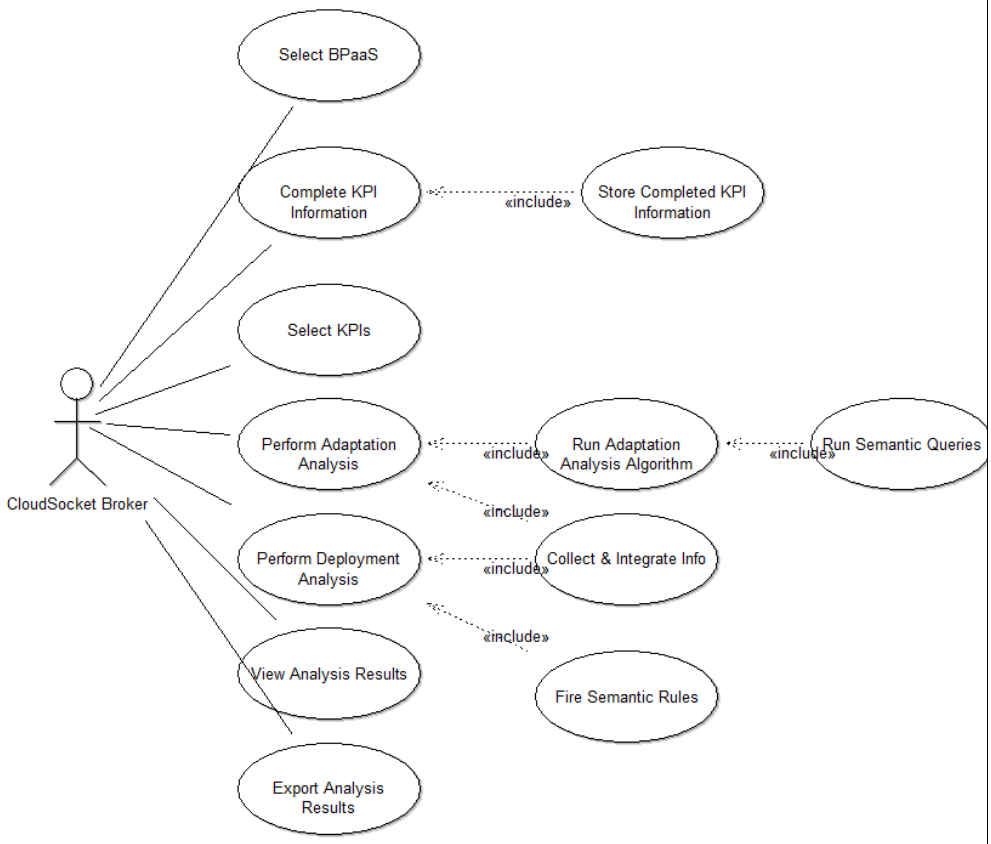


Figure 78 Use Case Diagram – EvE-UC-2-Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules

Copyright © 2016 BOC and other members of the CloudSocket Consortium

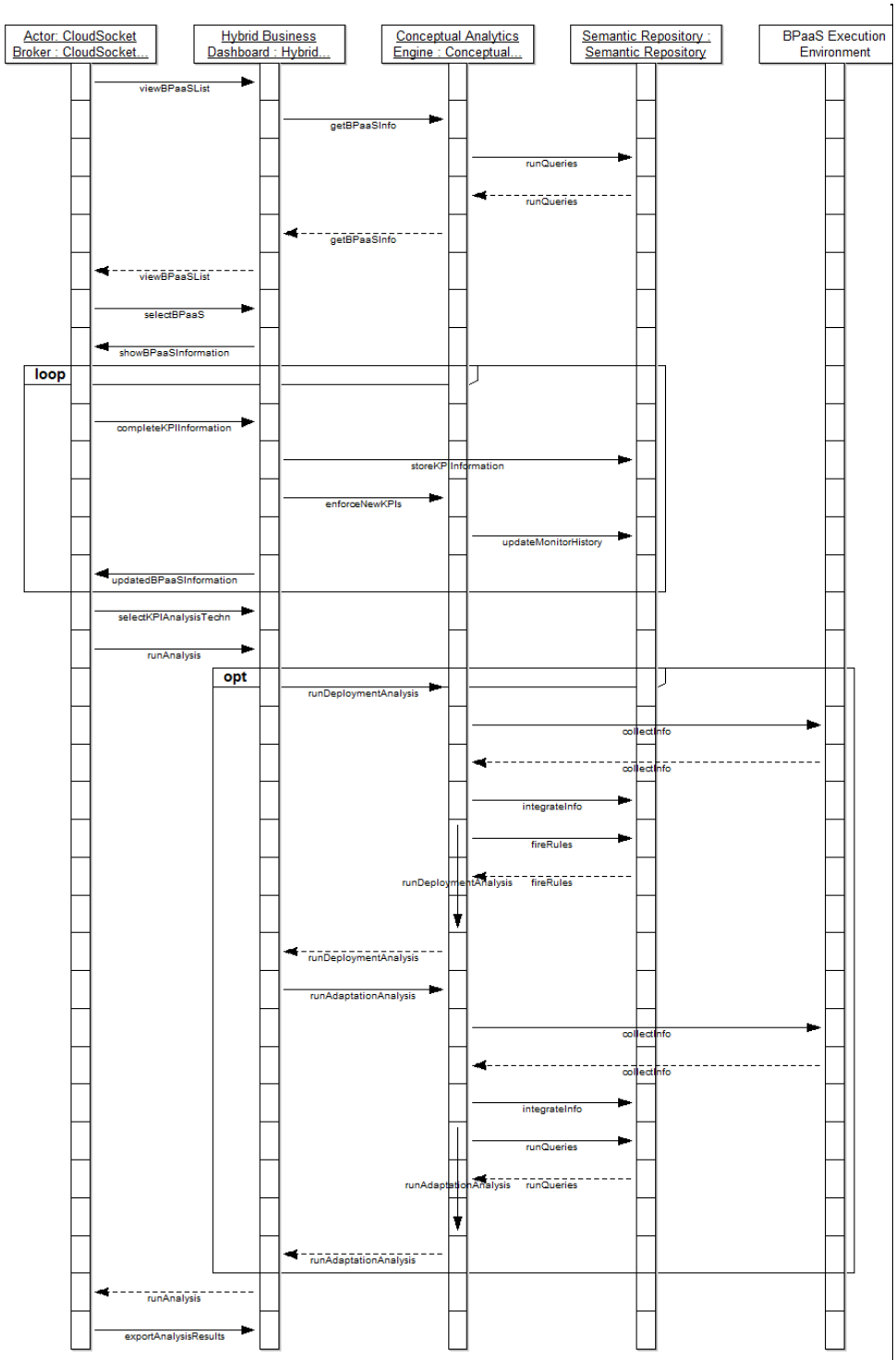


Figure 79 Sequence Diagram – EvE-UC-2-Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules

Table 28 BPaaS Evaluation Environment – Use Case 2 – Derivation & Visualisation of Best Deployments and Adaptation Patterns/Rules

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.2.3 Process Mining Analysis & Graphical Representation

Use Case id	EvE-Uc-3-Process Mining Analysis & Graphical Representation
Title & Description	The goal of this use case is to perform process mining over the workflow logs in order to discover any kind of optimisation finding. Such findings can take the following form: (a) process bottlenecks; (b) paths that are never followed; (c) new paths that were not initially anticipated; (d) process discrepancies with respect to a path that has been designed and the actual path that is followed during process execution. All these findings can then lead to optimising the BPaaS on different levels. To this end, they can be used as a guide for the next cycle of the BPaaS. We envisage that the CloudSocket Broker selects a particular BPaaS, and then requests to perform a process mining over its logs in order to find bottlenecks or discrepancies. The mining results are then visualized in order to enable the broker to see what is going wrong and what the main discrepancies are by using appropriate interaction elements in conjunction to the normal ones that are used to graphically represent the BPaaS business process. In the end, the CloudSocket broker can select to export the mining results in order to exploit them in the next cycle.
Actors	Cloud Socket Broker
Use Case Objective	Perform process mining & visualise respective results
Pre-Conditions	The BPaaS has been already purchased and be executed in the context of one or more customers. The existence of a BPaaS Execution Environment which produces the appropriate information that is used to build up the semantic repository on which the mining takes place. This information should at least comprise process and workflow log data. The BPaaS Execution Environment should provide an API through which this information can be retrieved.
Process Dialog	• The CloudSocket Broker selects a particular BPaaS from those offered. Depending on the context of the CloudSocket Broker, different characteristics of the BPaaS business process may be of interest. The hybrid dashboard also presents meta information of the business process, such as KPIs, human interaction and the like to enable the CloudSocket Broker to take an informed decision of which BPaaS to select for the analysis. • The CloudSocket Broker demands to perform process mining on the selected BPaaS ○ The user selection is forwarded in the form of a respective request to the Process Mining Engine. This engine first collects the appropriate information and stores it in the semantic repository. Then, it executes one or more state-of-the-art process mining algorithms. • The dashboard shows the analysis results over the so-called “assimilation” technique, where information is “assimilated” into business process models. • The CloudSocket Broker selects to export the process mining results to the suitable format for further processing in the next cycle.
Variations	• The CloudSocket Broker might be able to choose the mining algorithms for the analysis in order to focus on certain aspects.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

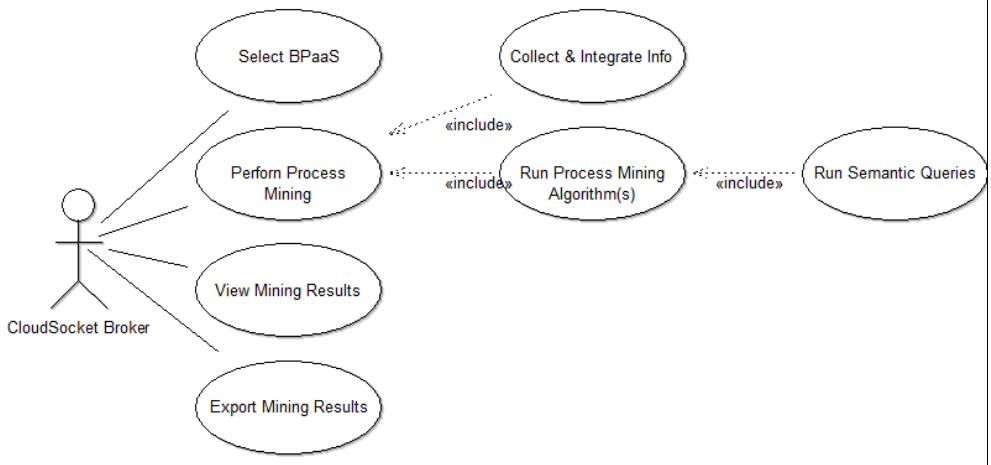
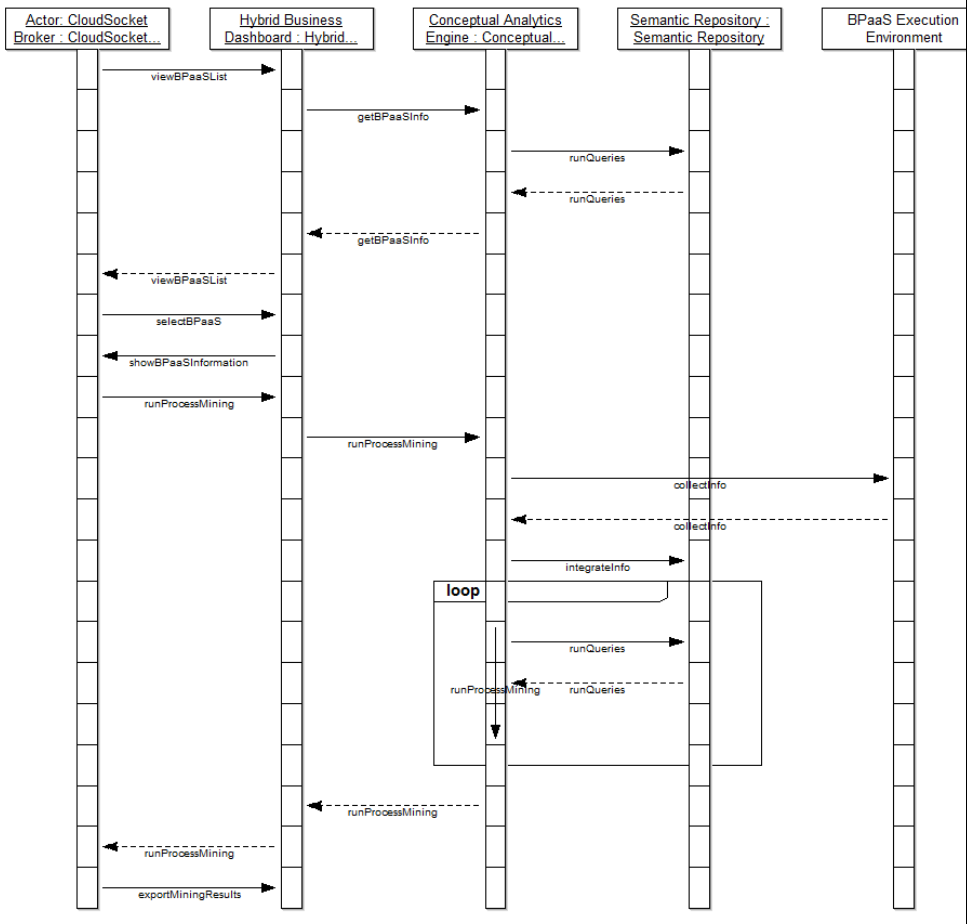
Post-Conditions	Process mining results are produced and possibly exported
Diagrams	 **Figure 80 Use Case Diagram - EvE-Uc-3-Process Mining Analysis & Graphical Representation**  **Figure 81 Sequence Diagram - EvE-Uc-3-Process Mining Analysis & Graphical Representation**

Table 29 BPaaS Evaluation Environment – Use Case 3 - Process Mining Analysis & Graphical Representation

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.3 Components

As already indicated, there are three layers involved in the architecture of the BPaaS Evaluation Environment. To this end, we present the components involved in this architecture based on the layer in which they exist.

9.3.1 User Interface Layer

9.3.1.1 Hybrid Business Dashboard

There is currently only one component in the UI layer, namely the Hybrid Business Dashboard. This component is responsible for enabling the user to select the appropriate analysis technique as well as visualize the respective analysis results. Depending on the type of analysis, the user has also the ability to export the results in an appropriate form. In addition, different visualisations and flows of interactions are involved.

In particular, when KPI analysis is involved, then the visualisation concentrates on depicting the assessment results over high-level business KPIs. The user is then enabled to select an e.g. problematic business KPI and inspect the assessment of technical KPIs from which the business KPI is derived. This form of top-down interaction enables to perform a root cause analysis for problematic performance situations. As a side-effect of having incomplete KPIs, the dashboard also enables the completion of the KPI specification in order to enable assessment for it over the Semantic Repository.

In case of process mining, the dashboard enables the user to select a BPaaS and perform analysis over it. It then graphically represents the mining results by “assimilating” analysis findings into the usual business process model. The analysis results can also be exported as was the case of the adaptation & deployment analysis in order to be exploited in the next cycle.

The dashboard consists of three major views:

(c) The Business Indicator view enables to see business and technical goals and KPIs represented in different categories. Those goals and KPIs are represented as a business dashboard using common representations like traffic lights to indicate if a KPI has been successfully reached or violated. Different drill down mechanisms or search mechanisms enable to identify violating business and technical (e.g., deployment, processing) indicators by supporting the CloudSocket Broker in finding cause and effect relations.

(d) The Process Deployment/Adaptation Analysis view consists of tables showing improvement potentials by listing the findings in a similar way like process analysis reports or process simulation results are represented. Additional features to use those reports are provided such as sort, search or the export in an appropriate format.

(e) The Process Mining view represents the original business process model, but “assimilates” process log findings into the business process model to indicate process mining results within the corresponding business process model.

The Hybrid Business Dashboard, depending on the type of analysis, has to invoke the respective component in the business logic. This means that in case of KPI assessment and deployment/adaptation analysis, the Conceptual Analytics Engine is to be involved, while in case of process mining, the Process Mining Engine needs to be invoked.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

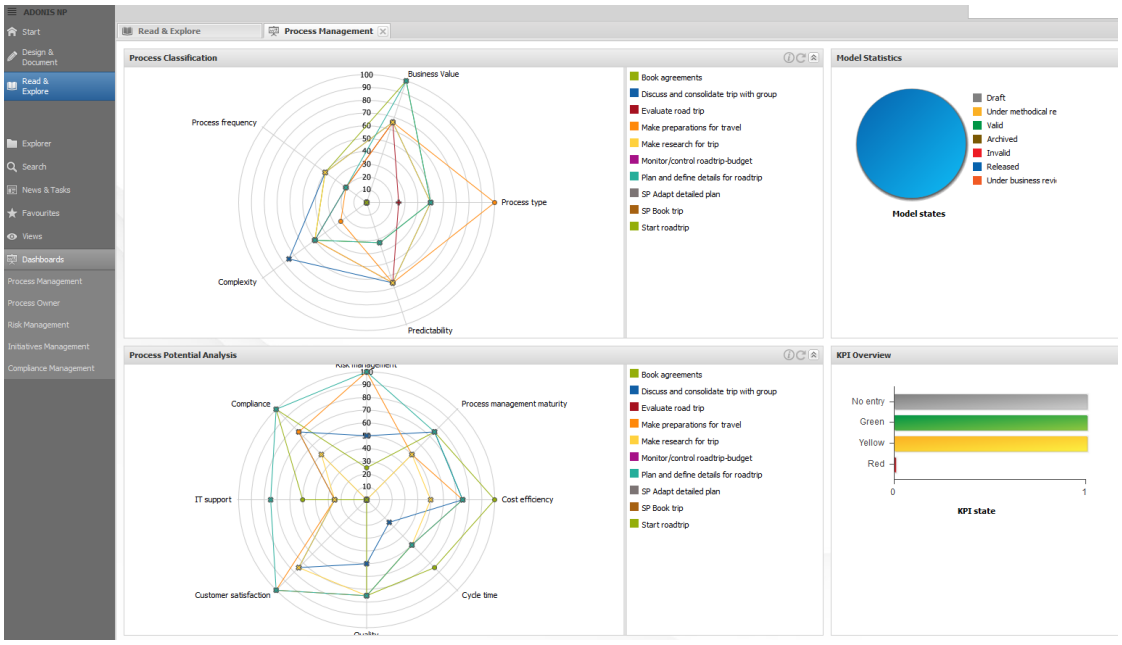


Figure 82 BPaaS Evaluation Environment - Business Dashboard User Interface Mockup

Figure 82 indicates a business dashboard, where KPIs are presented according their monitoring information.

9.3.2 Business Logic Layer

This layer comprises two components that perform different types of evaluation: (a) the Conceptual Analytics Engine and (b) the Process Mining Engine which provide all the core analysis functionality of the BPaaS Evaluation Environment. Both components exploit the REST API of the BPaaS Execution Environment in order to retrieve and link information in a semantic manner into the Semantic Repository in the Data Layer. Such information is then exploited in order to perform the respective analysis.

Additionally there is the Hybrid Business Process Management Tool that integrates the findings of the two engines, and orchestrates the communication from user interface to the analysis engines and the corresponding business process models.

9.3.2.1 Conceptual Analytics Engine

The Conceptual Analytics Engine presupposes that all information that can be used to assess KPIs already exists and can be drawn and integrated. It then enables to pose semantic queries over the Semantic Repository in order to derive the measurement value of a KPI. We envision that such semantic queries could involve the formulas over which the KPI measurement value can be produced as well as the required input parameters for those formulas. The completion of KPI information would then map to just specifying these semantic queries.

In case of deriving best deployments and adaptation rule suggestions, the Conceptual Analytics Engine can employ semantic rules which are able to reason over the existing knowledge in the Semantic Repository. As a similar approach has been developed in the PaaSage project, such an approach will be used and enriched to become semantically enhanced. The rules specified in the PaaSage KnowledgeBase could be exploited, extended and transformed into semantic rules.

For adaptation rule derivation, the engine relies on posing semantic queries over the Semantic Repository which will be used to draw all necessary information that is required in order to run the respective event pattern discovery algorithm exploited for a certain set of KPIs/SLOs. Such knowledge can of course be drawn from the

Copyright © 2016 BOC and other members of the CloudSocket Consortium

existing adaptation rules of the BPaaS which could have been manually modelled by the technical consultant in the BPaaS Allocation Environment or be produced in previous executions of the adaptation analysis.

- Tasks:
  - Perform adaptation rule/event pattern derivation
  - Perform best deployment analysis
  - Perform KPI assessment
  - Collect appropriate information from Execution Environment
- Main interfaces:
  - Evaluation phase:
    - Input: Process logs, monitoring logs, service logs, KPI specifications, SLO specifications, BPaaS (on which the analysis/assessment will be applied)
    - Output: assessment and analysis results to be visualized by the Hybrid Business Dashboard.

9.3.2.2 Process Mining Engine

The Process Mining Engine needs to have access to workflow logs and monitoring logs in order to be able to perform its analysis. The main analysis logic will focus on executing state-of-the-art process mining algorithms possibly extended with the capability of exploiting semantic information. One or more algorithms can be executed depending on their capabilities and the required focus aspects. This is due to the fact that each algorithm has different capabilities and performance, makes different assumptions on the content of a process log, and can focus on different aspects or different parts of a business process.

- Tasks:
  - Perform workflow mining by executing one or more state-of-the-art algorithms
  - Collect appropriate information from BPaaS Execution Environment
- Main interfaces:
  - Evaluation phase:
    - Input: Process logs, monitoring logs, service logs, KPI specifications, SLO specifications, BPaaS (on which the analysis/assessment will be applied)
    - Output: process mining results to be visualized by the Hybrid Business Dashboard.

9.3.3 Data Layer

The Semantic Repository will contain, after drawing it from the BPaaS Execution Environment, all the information that is required for the execution of the two core components of the Business Logic Layer. This information, which spans process, workflow, service, infrastructure and monitoring logs, will be semantically described and integrated such that it enables a meaningful browsing over it as well as the posing of smart queries which allow the derivation of information that can be invaluable for the required analysis on the next higher layer. The semantic information will be specified in OWL. The Semantic Repository will be realized in the form of a Triple Store which provides facilities to perform queries over a SPARQL endpoint. Various different Triple Stores could be exploited for this purpose with different functional and non-functional characteristics. On top of the Triple Store, there will be a reasoner which will enable validating the information stored as well as the derivation of new knowledge via the firing of semantic rules.

The Meta Model Platform stores all business process, workflow and KPI models that are needed to view dashboards, business processes or select workflows for analysis. It hence provides all necessary models and their corresponding functionality such as export, transformation or graphical representation.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

9.3.4 Component Diagram

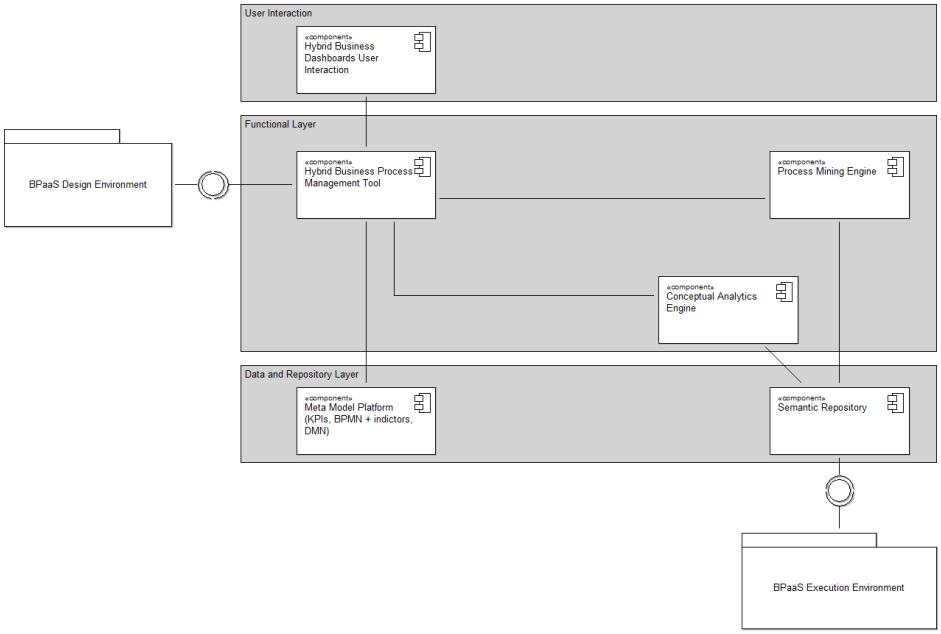


Figure 83 BPaaS Evaluation Environment – Components

9.4 Roles

The **Cloud Socket Broker** can involve users with different roles from the CloudSocket Broker organisation for conducting the evaluation and analysis for BPaaSs. The roles involved can be the following:

- Business role (e.g., busines goal designer and business process designer) who is familiar with balanced scorecard-like approaches and interested in the assessment of business KPIs and the discovery of optimisation hints at the business process level.
- Technical role (e.g., cloud infrastructure designer, workflow designer, and technical consultant) that has competence in technical rule management, and can thus configure the technical deployment and monitoring, as well as expertise in workflow design. This role will be interested in the assessment of technical KPIs and in obtaining optimisation hints at the technical/workflow level in the form of best deployments, adaptation rules and workflow discrepancies/bottlenecks. It will certainly have the expertise in completing the specification of adaptation rules by connecting event patterns to certain adaptation actions or strategies.

9.5 Data Interface

The following data are envisioned to be exchanged internally in this environment or externally with respect to the interaction with other CloudSocket environments:

- KPIs as well as SLOs can be exported from the dashboard in XML format and further translated – required the corresponding modelling of the KPIs - to formats like OWL-Q, or WS-Agreement.
- Process/workflow logs can rely on a standardized format or can be specific to the Workflow Engine implementation exploited. In either case, a transformation of logs to semantic information will be required. Monitoring logs will be specified in OWL-Q. This means that aggregated measurements are semantically enriched at the BPaaS Execution Environment. In Section 6, we have explicated the form that the measurements contained in the logs will take based on the OWL-Q specification.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

10 SUMMARY AND CONCLUSIONS

The first Architecture of CloudSocket has been developed based on the use case and evaluation criteria analysis as well as based on the original idea of the BPaaS Environment, corresponding to each phase of the Business Process Management System Paradigm (BPMS).

The high level architecture describes the core functional capabilities of each BPaaS Environment to encourage the development of alternative environments and hence enable a flexible configuration of different CloudSockets depending on the needs of the CloudSocket Broker.

This intension is stressed by the fact that the so-called BPaaS Bundle, the configuration file that enable the Business Process as a Service to become operative, allows missing sections to avoid vendor lock.

The conceptual bridge from domain specific business processes towards workflows that are in production in the cloud, is established by a layers model-based approach, using extensions to standards that enable the linkages and hence alignment between the different layers.

The description of each BPaaS Environment is not only understood as a technical specification but also as a contribution to the terminology WIKI of CloudSocket that is seen as a living document.

The first iteration of the CloudSocket Architecture builds on tools and prototypes that need to be linked together to a complete CloudSocket. The second iteration will integrate research findings to not only improve each individual BPaaS Environment but to improve the CloudSocket as a whole.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

11 REFERENCES

(ADONIS 2015) ADONIS Community Edition, www.adonis-community.com/, access: 2015.07.24

(ADOxx 2015) ADOxx, www.adoxx.org, access: 2015.07.24

(Apache 2016) Apache License Version 2.0: http://www.apache.org/licenses/LICENSE-2.0, access : 30.09.2016

(APQC 2014) Process Classification Framework Version 6.1.1. Available at http://www.apqc.org/pcf [Access December 23, 2015]

(BPMN 2015), OMG, http://www.bpmn.org, access: 16.03.2014

(CAMEL 2015), Rossini, A., Nikolov, N., Romero, D., Domaschka, J., Kritikos, K., Kirkham, T., Solberg, A.: D2.1.2 – CloudML Implementation Documentation. Paasage project deliverable (April 2014)

(CIDOC 2006) H. Kondylakis, M. Doerr and D. Plexousakis. Mapping Language for Information Integration. Technical Report 385, ICS_FORTH, 2006.

(CloudML 2014) Ferry, N., Song, H., Rossini, A., Chauvel, F., Solberg, A.. CloudMF: Applying MDE to Tame the Complexity of Managing Multi-Cloud Applications. In: Bilof, R., editor. UCC 2014: 7th IEEE / ACM International Conference on Utility and Cloud Computing. IEEE Computer Society; 2014, p. 269–277. doi:10.1109/UCC.2014.36.

(CSBT 2015) CloudSocket Broker Tools, CloudSocket Project, Web Page: Broker Entry Point, https://cloudsocket.eu/web/guest/brokercloudsocket, access: 31.07.2015

(C-SIG SLA 2014) Cloud Service Level Agreement Standardisation Guidelines. C-SIG on Service Level Agreement. Available at https://ec.europa.eu/digital-agenda/en/news/cloud-service-level-agreement-standardisation-guidelines [Accdess December 30, 2015).

(D2.1 2015) CloudSocket Project, D2.1 Use Case Analysis, www.cloudsocket.eu, access: 2015.05.18

(D2.2 2015) CloudSocket Project, D2.2 BPaaS References, Definitions & Common Terms, www.cloudsocket.eu, access: 2015.07.01

(D2.3 2015) CloudSocket Project, D2.3 Cloud Transformation Framework Demonstrator, https://cloudsocket.eu/transformation/, access: 31.07.2015

(D2.3a 2015) CloudSocket Project, D2.3 Cloud Transformation Framework Report, https://www.cloudsocket.eu/deliverables, access: 31.07.2015

(D3.1 2015) CloudSocket Project, D3.1 Modelling Framework for BPaaS, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D3.2 2016) CloudSocket Project, D3.2 Modelling Prototypes for BPaaS, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D3.3 2016) CloudSocket Project, D3.3 BPaaS Allocation and Execution Environment Bluprints, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

Copyright © 2016 BOC and other members of the CloudSocket Consortium

(D4.1 2015) CloudSocket Project, D4.1 First CloudSocket Architecture , https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D4.2_4.3_4.4 20916) CloudSocket Project, D4.2, 4.3, 4.4 Explanatory Notes: First Prototype, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D5.1 2016) CloudSocket Project, D5.1 Initial CloudSocket Setup Report, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D5.2 2016) CloudSocket Project, D5.2 CloudSocket BPaaS Reference Models, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D5.3 2016) CloudSocket Project, D5.3 CloudSocket BPaaS Allocations, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(D8.1 2016) CloudSocket Project, D8.1 First Exploitation and Business Plan, https://www.cloudsocket.eu/deliverables, access: 31.08.2016

(DMN 2015) OMG, 2015, Decision Model and Notation, http://www.omg.org/spec/DMN/1.0/Beta2/, access: 31.07.2015

(GNU 2016) GNU AGPL v3.0 - http://www.gnu.org/licenses/agpl-3.0.html, access : 30.09.2016

(GPL 2016) GPL v2: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html, access : 30.09.2016

(Kang 1990) Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feature-oriented domain analysis (FODA) feasibility study (No. CMU/SEI-90-TR-21). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

(Kar 1996) Karagiannis, D., Junginger, S. and Strobl, R.,1996, Introduction to Business Process Management System Concepts, in: B. Scholz-Reiter, E. Stickel (Eds.): Business Process Modelling, Lecture Notes in Computer Science, Springer.

(Kar 2002) Karagiannis, D.; Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2-6, 2002, LNCS 2455, Springer-Verlag, Berlin, Heidelberg, p. 182.

(Kar 2006) Karagiannis, D., Höfferer, P. 2006: Metamodels in Action: An overview, In: J. Filipe, B. Shishkov, M. Helfert, ICSOFT 2006 - First Int. Conf. on Software and Data Technologies:IS27-36. Setúbal: Insticc Press.

(Mozilla 2016) Mozilla Public License (MPL) : https://www.mozilla.org/en-US/MPL/2.0/, access: 30.09.2016

(OAuth 2.0 2015) OAuth, www.oauth.net/, access: 25.08.2015

(OIDC 2015) OpenID, openid.net/connect/, access:25.08.2015

(OMG 2015) Decision Model and Notation, Object Management Group (OMG), http://www.omg.org/spec/DMN/1.1/, 2016, access 27.09.2016

(OWL 2012) OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition) Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C Recommendation, 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/. Latest version available at http://www.w3.org/TR/owl2-syntax/.

Copyright © 2016 BOC and other members of the CloudSocket Consortium

(OWL-Q 2006) K. Kritikos and D. Plexousakis, “Semantic QoS Metric Matching,” in *ECOWS*. IEEE Computer Society, 2006, pp. 265–274.

(PaaSage 2015) PaaSage Project, http://www.paasage.eu/, access: 17.04.2015

(R2R) 1.Bizer, C., Schultz, A.: The R2R Framework: Publishing and Discovering Mappings on the Web. 1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai, November 2010.

(RDF 2014) Guus Schreiber, Yves Raimond. RDF 1.1 Primer. W3C Working Group Note, 25 February 2014. The latest version is available at http://www.w3.org/TR/rdf11-primer/.

(RML) A. Dimou and M. Vander Sande. RDF Mapping Language (RML). Unofficial Draft, iMinds Multimedia Lab, Ghent University. Available at: http://semweb.mmlab.be/rml/spec.html

(SAML 2015) OASIS Security Assertion Markup Language, www.oasis-open.org/standards#samlv2.0, access: 25.08.2015

(SCIM 2015) System for Cross-domain Identity Management, tools.ietf.org/wg/scim/, access: 31.08.2015

(SRL 2014) Kyriakos Kritikos, Jörg Domaschka, Alessandro Rossini: SRL: A Scalability Rule Language for Multi-cloud Environments. CloudCom 2014: 1-9.

(Str 1996) Strahringer S (1996) Metamodellierung als Instrument des Methodenvergleichs: eine Evaluierung am Beispiel objektorientierter Analysemethoden. Shaker, Aachen

(USDL 2015) J. Cardoso and C. Pedrinaci. Evolution and Overview of Linked USDL. In *IESS*, 2015.

(WS-Agreement 2009) Open Grid Forum, WS-Agreement Schema, http://schemas.ggf.org/graap/2007/03/ws-agreement, access: 25.08.2015

(WS-Agreement 2011) Open Grid Forum, WS-Agreement, https://www.ogf.org/documents/GFD.192.pdf, access: 25.08.2015

(yourBPM 2015a) yourBPM, Installation Manual: forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Light_Semantic_Composition_-_Installation_and_Administration_Guide, access: 2015.07.24

(yourBPM 2015b) yourBPM, User Manual, forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Light_Semantic_Composition_-_User_and_Programmer_Guide, access: 2015.07.24

Copyright © 2016 BOC and other members of the CloudSocket Consortium

12 ANNEX

12.1 BPaaS Bundle Sending Christmas Greeting Cards

```
camel model BundleCamelModel {
    application ChristmasGreetingsCard {
        version: 'v0.3'
        description: 'This is the CAMEL file for the ChristmasGreetingsCard BPaaS Bundle'
        owner: Organization.Owner
        deployment models [
            BundleCamelModel.DeploymentPlan
        ]
    }

    organisation model Organization {
        organisation Company {
            www: 'www.example.com'
            postal address: 'postal-address'
            email: 'info@example.com'
        }

        user Owner {
            first name: 'firstname'
            last name: 'lastname'
            email: 'firstname.lastname@example.com'
            paasage credentials 'Not necessary for CloudSocket'
            cloud credentials [
                OmistackCredential {
                    username: 'tenant:user'
                    password: 'topsecret'
                    cloud provider: BundleCamelModel.OmistackOrganisation.omistack
                }
            ]
        }

        security level: HIGH
    }

    deployment model DeploymentPlan {
        requirement set ChristmasGreetingsCardRequirement {
            os: Requirement.UbuntuOS
        }

        vm ChristmasGreetingsCardVM {
            requirement set ChristmasGreetingsCardRequirement
            provided host ChristmasGreetingsCardCompononentHost
        }

        internal component ChristmasGreetingsCardComponent {
            provided communication WebServiceCommunication { port: 2181 }
            required host ChristmasGreetingsCardHostReq

            configuration ChristmasGreetingsCardConfiguration {
                download: 'sudo apt-get install -y curl'
                install: 'curl -o ChristmasGreetingsCard_checkstart_ubuntu.sh https://omi-gitlab.e-technik.uni-
ulm.de/cloudsocket/prototype_v1_files/raw/master/ChristmasGreetingsCard_checkstart_ubuntu.sh && chmod +x
ChristmasGreetingsCard_checkstart_ubuntu.sh && curl -o ChristmasGreetingsCard_install_ubuntu.sh https://omi-gitlab.e-technik.uni-
ulm.de/cloudsocket/prototype_v1_files/raw/master/ChristmasGreetingsCard_install_ubuntu.sh && chmod +x
ChristmasGreetingsCard_install_ubuntu.sh && ./ChristmasGreetingsCard_install_ubuntu.sh'
                configure: ''
                start: '/apache-tomcat-7.0.65/bin/catalina.sh run'
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
        upload: 'source /ChristmasGreetingsCard_checkstart_ubuntu.sh'
                        }
                }

                hosting ChristmasGreetingsCardToChristmasGreetingsCardVM {
                        from ChristmasGreetingsCardComponent.ChristmasGreetingsCardHostReq to
ChristmasGreetingsCardVM.ChristmasGreetingsCardCompononentHost
                }

                vm instance ChristmasGreetingsCardOmistackSmallInstance typed
BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardVM {
                        vm type: BundleCamelModel.OmistackProvider.Omistack.VM.VMType
                        vm type value: BundleCamelModel.OmistackType.VMTypeEnumeration.m1.small
                        provided host instance ChristmasGreetingsCardComponentHostInstance typed
ChristmasGreetingsCardVM.ChristmasGreetingsCardCompononentHost
                }

                internal component instance ChristmasGreetingsCardComponentInstance typed
BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent{
                        required host instance ChristmasGreetingsCardInstanceHostReq typed
ChristmasGreetingsCardComponent.ChristmasGreetingsCardHostReq
                        provided communication instance ChristmasGreetingsCardWebServiceCommunication typed
ChristmasGreetingsCardComponent.WebServiceCommunication
                }

                host ChristmasGreetingsCardComponentInstance.ChristmasGreetingsCardInstanceHostReq on
ChristmasGreetingsCardOmistackSmallInstance.ChristmasGreetingsCardComponentHostInstance typed
BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardToChristmasGreetingsCardVM
        }

        requirement model Requirement {
                slo CPU_HIGH_SLO{
                        service level: BundleCamelModel.MetricModel.CPU_HIGH_RawMetricCondition
                }
                slo CPU_AVG_2MIN_SLO {
                        service level: BundleCamelModel.MetricModel.CPU_AVG_2MIN_Condition
                }
                slo CPU_AVG_DAILY_SLO{
                        service level: BundleCamelModel.MetricModel.CPU_AVG_DAILY_Condition
                }
                slo RAM_MAX_DAILY_SLO{
                        service level: BundleCamelModel.MetricModel.RAM_MAX_DAILY_Condition
                }

                os UbuntuOS {os: 'Ubuntu' 64os}
        }

        metric model MetricModel {
                property CpuUtilization{
                        type: MEASURABLE
                }
                property RamUtilization{
                        type: MEASURABLE
                }
                sensor CpuSensor{
                        configuration: 'cpu_usage;de.uniulm.omi.cloudiator.visor.sensors.CpuUsageSensor'
                }
                sensor RamSensor{
          configuration: 'memory_usage;de.uniulm.omi.cloudiator.visor.sensors.MemoryUsageSensor'
        }
        schedule Schedule10Seconds{
                        type: FIXED_RATE
                        interval: 10
                        unit: BundleCamelModel.UnitModel.Seconds
                }
        schedule Schedule2Minutes{
                        type: FIXED_RATE
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
		interval: 2
		unit: BundleCamelModel.UnitModel.Minutes
}
schedule ScheduleDaily{
		type: FIXED_RATE
		interval: 24
		unit: BundleCamelModel.UnitModel.Hours
}
window Window4Minutes {
		window type: SLIDING
		size type: TIME_ONLY
		time size: 4
		unit: BundleCamelModel.UnitModel.Minutes
}
window WindowDaily {
		window type: FIXED
		size type: TIME_ONLY
		time size: 24
		unit: BundleCamelModel.UnitModel.Hours
}

raw metric CPU_RawMetric {
		value direction: 0
		layer: IaaS
		property: BundleCamelModel.MetricModel.CpuUtilization
		unit: BundleCamelModel.UnitModel.CpuUnit
		value type: BundleCamelModel.TypeModel.Between_0_100
}

composite metric CPU_AVG_CompositeMetric {
		value direction: 0
		layer: IaaS
		property: BundleCamelModel.MetricModel.CpuUtilization
		unit: BundleCamelModel.UnitModel.CpuUnit
		value type: BundleCamelModel.TypeModel.Between_0_100

		metric formula Formula{
			function arity: UNARY
			function pattern: REDUCE
			MEAN(BundleCamelModel.MetricModel.CPU_RawMetric)
		}
}

raw metric context CPU_RawMetricContext{
		metric: BundleCamelModel.MetricModel.CPU_RawMetric
		sensor: MetricModel.CpuSensor
		component: BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent
		schedule: BundleCamelModel.MetricModel.Schedule10Seconds
		quantifier: ALL
}

composite metric context CPU_AVG_2MIN_CompositeMetricChristmasGreetingsCardComponentContext {
		metric: BundleCamelModel.MetricModel.CPU_AVG_CompositeMetric
		component: BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent
		window: BundleCamelModel.MetricModel.Window4Minutes
		schedule: BundleCamelModel.MetricModel.Schedule2Minutes
		composing metric contexts [BundleCamelModel.MetricModel.CPU_RawMetricContext]
		quantifier: ALL
}

composite metric context CPU_AVG_DAILY_CompositeMetricChristmasGreetingsCardComponentContext {
		metric: BundleCamelModel.MetricModel.CPU_AVG_CompositeMetric
		component: BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent
		window: BundleCamelModel.MetricModel.WindowDaily
		schedule: BundleCamelModel.MetricModel.ScheduleDaily
		composing metric contexts [BundleCamelModel.MetricModel.CPU_RawMetricContext]
		quantifier: ALL
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
		}

		metric condition CPU_HIGH_RawMetricCondition {
			context: BundleCamelModel.MetricModel.CPU_RawMetricContext
			threshold: 80.0
			comparison operator: < =
		}

		metric condition CPU_AVG_2MIN_Condition {
			context:
BundleCamelModel.MetricModel.CPU_AVG_2MIN_CompositeMetricChristmasGreetingsCardComponentContext
			threshold: 20.0
			comparison operator: < =
		}

		metric condition CPU_AVG_DAILY_Condition {
			context:
BundleCamelModel.MetricModel.CPU_AVG_DAILY_CompositeMetricChristmasGreetingsCardComponentContext
			threshold: 40.0
			comparison operator: < =
		}

		raw metric RAM_RawMetric {
			value direction: 0
			layer: IaaS
			property: BundleCamelModel.MetricModel.RamUtilization
			unit: BundleCamelModel.UnitModel.RamUnit
			value type: BundleCamelModel.TypeModel.Between_0_100
		}

		composite metric RAM_MAX_CompositeMetric
		{
			value direction: 0
			layer: IaaS
			property: BundleCamelModel.MetricModel.RamUtilization
			unit: BundleCamelModel.UnitModel.RamUnit
			value type: BundleCamelModel.TypeModel.Between_0_100

			metric formula Formula{
				function arity: UNARY
				function pattern: REDUCE
				MAX(BundleCamelModel.MetricModel.RAM_RawMetric)
			}
		}

		raw metric context RAM_RawMetricContext {
			metric: BundleCamelModel.MetricModel.RAM_RawMetric
			sensor: MetricModel.RamSensor
			component: BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent
			schedule: BundleCamelModel.MetricModel.Schedule10Seconds
		}

		composite metric context RAM_MAX_DAILY_CompositeMetricContext{
			metric: BundleCamelModel.MetricModel.RAM_MAX_CompositeMetric
			component: BundleCamelModel.DeploymentPlan.ChristmasGreetingsCardComponent
			window: BundleCamelModel.MetricModel.WindowDaily
			schedule: BundleCamelModel.MetricModel.ScheduleDaily
			composing metric contexts [BundleCamelModel.MetricModel.RAM_RawMetricContext]
			quantifier: ALL
		}

		metric condition RAM_MAX_DAILY_Condition {
			context: BundleCamelModel.MetricModel.RAM_MAX_DAILY_CompositeMetricContext
			threshold: 60.0
			comparison operator: < =
		}
	}
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
type model TypeModel {
        range Between_0_100 {
                primitive type: DoubleType
                lower limit {
                        double value 0.0
                }
                upper limit {
                        double value 100.0
                }
        }
}

unit model UnitModel {
        dimensionless {
                CpuUnit: PERCENTAGE
        }
        dimensionless {
                RamUnit: PERCENTAGE
        }
        time interval unit {
                Seconds : SECONDS
        }
        time interval unit {
                Minutes : MINUTES
        }
        time interval unit {
                Hours : HOURS
        }
}

location model OmistackLocation {
        country  DE {
                name:  Germany
        }
}
organisation model OmistackOrganisation {
        provider  omistack {
                www:  "www.uni-ulm.de"
                postal address: "Ulm University, Institute of Information Resource Management, Albert-Einstein-Allee 43, D-89081 Ulm"
                email:  ""
                public
                IaaS provider model:  BundleCamelModel.OmistackProvider
        }
        data centre  OmistackDataCentre {
                code name:  gwdg location:  OmistackLocation.DE
        }
        security level:  LOW
}
provider model  OmistackProvider {
        constraints  {
                implies  M1_SMALL_Constraint_Mapping {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMMemory from value:
                                        "m1.small" : 0 to value:  int value  2048
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMCores from value:
                                        "m1.small" : 0 to value:  int value  1
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMStorage from value:
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                                        "m1.small" : 0 to value:  int value  20
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMTypeCloudProviderId
                                        from value:  "m1.small" : 0 to value:  "RegionOne/2" : 0
                                }
                        }
                }
                implies  M1_LARGE_Constraint_Mapping {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMMemory from value:
                                        "m1.large" : 1 to value:  int value  8192
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMCores from value:
                                        "m1.large" : 1 to value:  int value  4
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMStorage from value:
                                        "m1.large" : 0 to value:  int value  80
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMTypeCloudProviderId
                                        from value:  "m1.large" : 1 to value:  "RegionOne/4" : 1
                                }
                        }
                }
                implies
                RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97_Constraint_Mapping_OS {
                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97
                        to:  BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints  {
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97.VMImageId
                                        to:  BundleCamelModel.OmistackProvider.Omistack.VM.VMOS from value:
                                        string value  "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" to value:
                                        "Ubuntu Server 14.04.2 AMD64 LTS" : 0
                                }
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97.VMImageId
                                        to:     BundleCamelModel.OmistackProvider.Omistack.VM.OSVendorType    from
    value:
                                        string value  "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" to value:
                                        NIX : 1
                                }
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97.VMImageId
                                        to:  BundleCamelModel.OmistackProvider.Omistack.VM.OSArchitecture
                                        from value:  string value
                                        "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" to value:  AMD64 : 0
                                }
                        }
                }
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                implies
                RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97_Constraint_Mapping_LOCATION {
                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97
                        to:  BundleCamelModel.OmistackProvider.Omistack.Location
                        attribute constraints  {
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97.VMImageId
                                                to:  BundleCamelModel.OmistackProvider.Omistack.Location.LocationId
                                                from value:  string value
                                                "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" to value:  RegionOne : 0
                                }
                        }
                }
                implies
                RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938_Constraint_Mapping_OS {
                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938
                        to:  BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints  {
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938.VMImageId
                                                to:  BundleCamelModel.OmistackProvider.Omistack.VM.VMOS from value:
                                                string value  "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" to value:
                                                "Windows2012R2_PW_No_Firewall_0.2" : 0
                                }
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938.VMImageId
                                                to:      BundleCamelModel.OmistackProvider.Omistack.VM.OSVendorType    from
value:
                                                string value  "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" to value:
                                                WINDOWS : 0
                                }
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938.VMImageId
                                                to:  BundleCamelModel.OmistackProvider.Omistack.VM.OSArchitecture
                                                from value:  string value
                                                "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" to value:  AMD64 : 0
                                }
                        }
                }
                implies
                RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938_Constraint_Mapping_LOCATION {
                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938
                        to:  BundleCamelModel.OmistackProvider.Omistack.Location
                        attribute constraints  {
                                attribute constraint  {
                                        from:

        BundleCamelModel.OmistackProvider.Omistack.VM.VMImage.RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938.VMImageId
                                                to:  BundleCamelModel.OmistackProvider.Omistack.Location.LocationId
                                                from value:  string value
                                                "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" to value:  RegionOne : 0
                                }
                        }
                }
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
implies M1_TINY_Constraint_Mapping {
    from: BundleCamelModel.OmistackProvider.Omistack.VM to:
    BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints {
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMMemory from value:
            "m1.tiny" : 2 to value: int value 512
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMCores from value:
"m1.tiny"
            : 2 to value: int value 1
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMStorage from value:
            "m1.tiny" : 2 to value: int value 1
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMTypeCloudProviderId
            from value: "m1.tiny" : 2 to value: "RegionOne/1" : 2
        }
    }
}
implies M1_MEDIUM_Constraint_Mapping {
    from: BundleCamelModel.OmistackProvider.Omistack.VM to:
    BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints {
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMMemory from value:
            "m1.medium" : 3 to value: int value 4096
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMCores from value:
            "m1.medium" : 3 to value: int value 2
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMStorage from value:
            "m1.medium" : 3 to value: int value 40
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMTypeCloudProviderId
            from value: "m1.medium" : 3 to value: "RegionOne/3" : 3
        }
    }
}
implies M1_XLARGE_Constraint_Mapping {
    from: BundleCamelModel.OmistackProvider.Omistack.VM to:
    BundleCamelModel.OmistackProvider.Omistack.VM attribute constraints {
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMMemory from value:
            "m1.xlarge" : 4 to value: int value 16384
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMCores from value:
            "m1.xlarge" : 4 to value: int value 8
        }
        attribute constraint {
            from: BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
            BundleCamelModel.OmistackProvider.Omistack.VM.VMStorage from value:
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                                        "m1.xlarge" : 4 to value:  int value  160
                                }
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.VM.VMTypeCloudProviderId
                                        from value:  "m1.xlarge" : 4 to value:  "RegionOne/5" : 4
                                }
                        }
                }
                implies  M1_SMALL_Constraint_Mapping_LOCATION {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.Location attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.Location.LocationId from value:
                                        "m1.small" : 0 to value:  RegionOne : 0
                                }
                        }
                }
                implies  M1_LARGE_Constraint_Mapping_LOCATION {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.Location attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.Location.LocationId from value:
                                        "m1.large" : 1 to value:  RegionOne : 0
                                }
                        }
                }
                implies  M1_TINY_Constraint_Mapping_LOCATION {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.Location attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.Location.LocationId from value:
                                        "m1.tiny" : 2 to value:  RegionOne : 0
                                }
                        }
                }
                implies  M1_MEDIUM_Constraint_Mapping_LOCATION {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.Location attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.Location.LocationId from value:
                                        "m1.medium" : 3 to value:  RegionOne : 0
                                }
                        }
                }
                implies  M1_XLARGE_Constraint_Mapping_LOCATION {
                        from:  BundleCamelModel.OmistackProvider.Omistack.VM to:
                        BundleCamelModel.OmistackProvider.Omistack.Location attribute constraints  {
                                attribute constraint  {
                                        from:  BundleCamelModel.OmistackProvider.Omistack.VM.VMType to:
                                        BundleCamelModel.OmistackProvider.Omistack.Location.LocationId from value:
                                        "m1.xlarge" : 4 to value:  RegionOne : 0
                                }
                        }
                }
        }
        root  feature  Omistack {
                attributes  {
                        attribute  DeploymentModel {
                                value:  string value  Private value type:
                                BundleCamelModel.OmistackType.StringValueType
                        }
                        attribute  ServiceModel {
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                value: string value ^IaaS value type:
                BundleCamelModel.OmistackType.StringValueType
            }
            attribute Availability {
                unit type: PERCENTAGE value: string value "95" value type:
                BundleCamelModel.OmistackType.StringValueType
            }
            attribute Driver {
                value: string value "openstack-nova" value type:
                BundleCamelModel.OmistackType.StringValueType
            }
            attribute EndPoint {
                value: string value
                "http://omistack-beta.e-technik.uni-ulm.de:5000/v2.0" value type:
                BundleCamelModel.OmistackType.StringValueType
            }
            attribute Name {
                value: string value omistack value type:
                BundleCamelModel.OmistackType.StringValueType
            }
        }
        sub-features {
            feature VM {
                attributes {
                    attribute VMType {
                        value type: BundleCamelModel.OmistackType.VMTypeEnumeration
                    }
                    attribute VMOS {
                        value type: BundleCamelModel.OmistackType.VMOsEnum
                    }
                    attribute VMMemory {
                        unit type: MEGABYTES value type:
BundleCamelModel.OmistackType.MemoryList
                    }
                    attribute VMCores {
                        value type: BundleCamelModel.OmistackType.CoresList
                    }
                    attribute CostPerHour {
                        value type: BundleCamelModel.OmistackType.CostRange
                    }
                    attribute VMStorage {
                        unit type: GIGABYTES value type:
                        BundleCamelModel.OmistackType.StorageList
                    }
                    attribute OSVendorType {
                        value type: BundleCamelModel.OmistackType.OSVendorType
                    }
                    attribute OSArchitecture {
                        value type: BundleCamelModel.OmistackType.OSArchitectureType
                    }
                    attribute VMTypeCloudProviderId {
                        value type:
BundleCamelModel.OmistackType.VMTypeCloudProviderIdEnum
                    }
                }
                sub-features {
                    exclusive VMImage {
                        feature cardinality {
                            cardinality: 1 .. 1 value: 1
                        }
                        variants {
                            feature
RegionOne_11a845d0_7ed3_48c8_a36a_9a76a2fe4938 {
                                attributes {
                                    attribute VMImageId {
                                        value: string value
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
        "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" value type:

        BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                                    attribute  DefaultLoginName {
                                                                                            value:    string  value
Administrator value type:

        BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                                    attribute  DefaultLoginPassword {
                                                                                            value:    string  value
Admin1 value type:

        BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                            }
                                                                            feature cardinality {
                                                                                    cardinality:  0 .. 1
                                                                            }
                                                                    }
                                                                    feature
RegionOne_9c154d9a_fab9_4507_a3d7_21b72d31de97 {
                                                                            attributes  {
                                                                                    attribute  VMImageId {
                                                                                            value:  string value

        "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" value type:

        BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                                    attribute  DefaultLoginName {
                                                                                            value:    string  value
ubuntu value type:

        BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                                    attribute  DefaultLoginPassword {
                                                                                            value                type:
BundleCamelModel.OmistackType.StringValueType
                                                                                    }
                                                                            }
                                                                            feature cardinality {
                                                                                    cardinality:  0 .. 1
                                                                            }
                                                                    }
                                                            }
                                                    }
                                            }
                                            feature cardinality {
                                                    cardinality:  1 .. 8
                                            }
                                    }
                                    feature  Location {
                                            attributes  {
                                                    attribute  LocationId {
                                                            value type:  BundleCamelModel.OmistackType.LocationIdType
                                                    }
                                                    attribute  longitude {
                                                            value:  float value  9.96291
                                                    }
                                                    attribute  latitude {
                                                            value:  float value  48.42599
                                                    }
                                            }
                                            sub-features  {
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                                                feature  EU {
                                                        sub-features  {
                                                                feature  Germany {
                                                                        sub-features  {
                                                                                feature  Ulm {
                                                                                        feature cardinality {

cardinality:  1 .. 1
                                                                                        }
                                                                                }
                                                                        }
                                                                        feature cardinality {
                                                                                cardinality:  1 .. 1
                                                                        }
                                                                }
                                                        }
                                                        feature cardinality {
                                                                cardinality:  1 .. 1
                                                        }
                                                }
                                        }
                                        feature cardinality {
                                                cardinality:  1 .. 1 value:  1
                                        }
                                }
                        }
                        feature cardinality {
                                cardinality:  1 .. 1
                        }
                }
}
type model  OmistackType {
        enumeration  VMTypeCloudProviderIdEnum {
                values  [ "RegionOne/2" : 0,
                "RegionOne/4" : 1,
                "RegionOne/1" : 2,
                "RegionOne/3" : 3,
                "RegionOne/5" : 4 ]
        }
        enumeration  VMOsEnum {
                values  [ "Ubuntu Server 14.04.2 AMD64 LTS" : 0,
                "Windows2012R2_PW_No_Firewall_0.2" : 1 ]
        }
        enumeration  VMImageIdEnum {
                values  [ "RegionOne/11a845d0-7ed3-48c8-a36a-9a76a2fe4938" : 0,
                "RegionOne/9c154d9a-fab9-4507-a3d7-21b72d31de97" : 1 ]
        }
        range  CostRange {
                primitive type:  DoubleType lower limit  {
                        double value  0.054 included
                }
                upper limit  {
                        double value  0.095 included
                }
        }
        string value type  StringValueType {
                primitive type:  StringType
        }
        list  StorageList {
                values  [ int value  20,
                int value  80,
                int value  1,
                int value  40,
                int value  160 ]
        }
        list  MemoryList {
                values  [ int value  2048,
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
                    int value  8192,
                    int value  512,
                    int value  4096,
                    int value  16384 ]
                }
                enumeration  OSVendorType {
                    values  [ WINDOWS : 0,
                    NIX : 1 ]
                }
                enumeration  OSArchitectureType {
                    values  [ AMD64 : 0 ]
                }
                enumeration  LocationIdType {
                    values  [ RegionOne : 0 ]
                }
                enumeration  VMTypeEnumeration {
                    values  [ "m1.small" : 0,
                    "m1.large" : 1,
                    "m1.tiny" : 2,
                    "m1.medium" : 3,
                    "m1.xlarge" : 4 ]
                }
                list  CoresList {
                    values  [ int value  1,
                    int value  2,
                    int value  4,
                    int value  8 ]
                }
        }
    }
```

12.2 WS-Agreement Sample

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<wsag:Template wsag:TemplateId="01fc0444-4132-4b48-9ee4-5b7579d11092" xmlns:cs="http://wsag.sla.cloudsocket.eu"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sla="http://sla.atos.eu">
  <wsag:Name>Template JIRA ticketing</wsag:Name>
  <wsag:Context>
    <wsag:AgreementResponder>BROKER-B</wsag:AgreementResponder>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <sla:Service>BPAAS-1</sla:Service>
    <cs:context>
      <cs:assessment>BROKER-B</cs:assessment>
      <cs:monitoring>BROKER-B</cs:monitoring>
    </cs:context>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:GuaranteeTerm wsag:Name="gt03">
        <wsag:ServiceLevelObjective>
          <wsag:KPITarget>
            <wsag:KPIName>gt03_kpi</wsag:KPIName>
            <wsag:CustomServiceLevel>
              <cs:slo>
                <cs:constraint>CPU_AVG_2MIN_Condition NOT EXISTS</cs:constraint>
                <cs:description>Ensures that AVG(CPU_Usage, 2min) &amp;lt; 50%</cs:description>
              </cs:slo>
            </wsag:CustomServiceLevel>
          </wsag:KPITarget>
        </wsag:ServiceLevelObjective>
        <wsag:BusinessValueList>
          <wsag:CustomBusinessValue count="1" duration="P0Y0M0DT0H0M0.000S">
            <sla:Penalty type="discount" expression="0.05" unit="%" validity="P1D"/>
            <sla:description>Violation of SLO incurs in a 0,05% monthly discount.</sla:description>
          </wsag:CustomBusinessValue>
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium

```
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:GuaranteeTerm wsag:Name="gt04">
        <wsag:ServiceLevelObjective>
          <wsag:KPITarget>
            <wsag:KPIName>gt04_kpi</wsag:KPIName>
            <wsag:CustomServiceLevel>
              <cs:slo>
                <cs:constraint>CPU_AVG_DAILY_Condition NOT EXISTS</cs:constraint>
                <cs:description>Ensures that AVG(CPU_Usage, 1day) &lt; 40%</cs:description>
              </cs:slo>
            </wsag:CustomServiceLevel>
          </wsag:KPITarget>
        </wsag:ServiceLevelObjective>
        <wsag:BusinessValueList>
          <wsag:CustomBusinessValue count="1" duration="P0Y0M0DT0H0M0.000S">
            <sla:Penalty type="discount" expression="5" unit="%" validity="P1M"/>
            <sla:description>Violation of SLO incurs in a 5% monthly discount. If twice in a day, that day is not charged.</sla:description>
          </wsag:CustomBusinessValue>
          <wsag:CustomBusinessValue count="2" duration="P1D">
            <sla:Penalty type="discount" expression="100" unit="%" validity="P1D"/>
            <sla:description>Violation of SLO incurs in a 5% monthly discount. If twice in a day, that day is not charged.</sla:description>
          </wsag:CustomBusinessValue>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
    </wsag:All>
  </wsag:Terms>
    </wsag:Template>
```

Copyright © 2016 BOC and other members of the CloudSocket Consortium