BPaaS in Multi-Cloud Environments - The CloudSocket Approach

Frank Griesinger, Daniel Seybold, Stefan Wesner, Jörg Domaschka, Robert Woitsch, Kyriakos Kritikos, Knut Hinkelmann, Emanuele Laurenzi, Joaquin Iranzo, Román Sosa González, and Constantin Valeriu Tuguran

This is the reprint of an article published in the SCITEPRESS Digital Library (European Project Space: Developments, Implementations and Impacts in a Changing World, April 21-29, 2017, in Porto, Portugal Vol. 1 - 978-989-758-311-7). The original version is available online at https://doi.org/10.5220/0007901700500074
Abstract. Cloud computing proved to offer flexible IT solutions. Although large enterprises may benefit from this technology by educating their IT departments, SMEs face the risk to dramatically falling behind in cloud usage and hence lose the ability to efficiently adapt their IT to their business needs. This chapter presents the vision and the outcome of the H2020 project CloudSocket. The foundation lays the idea of Business Processes as a Service, where concept models and semantics are applied to align business processes with Multi-Cloud deployed workflows. The proposed CloudSocket platform consists of four architectural building blocks: (i) design, (ii) allocation, (iii) execution, and (iv) evaluation. These are organized as environments that cope with specific tasks and research questions. An overview of each environment is given along with main prototypes that were developed to push the state-of-the-art in the respective field. We show the success of the achievements in current research endeavours and how we will pursue the open questions.

1 Introduction

Service provisioning in a digital world is both an opportunity and a challenge for the public and the private sector to innovate service delivery models, embracing creative disruption from technology, and adopting an attitude of experimentation and entrepreneurship. In particular, we see that service offerings become digital relying on technologies such as big data, cloud computing, mobile technologies, robotics, IoT, social networks. Business processes are seen as an adequate instrument to manage the digitalization of business. Outsourcing parts or the whole business process is a common approach in order to reduce costs and increase flexibility [1]. In this context, cloud computing is a promising solution by providing compute-oriented service models (IaaS,
PaaS, SaaS) to more fine grained and multi-dimensional service models [3]. Hence, cloud computing has become a large driver of business models and digitalisation. While adopters of cloud computing originally were IT-oriented companies, SaaS also allows non-technical companies to benefit from cloud computing. Yet, SaaS offerings are usually inflexible and their vendors expect them to be one-size-fits-all solutions for their customers often ignoring that they need to integrate into the users existing business processes. In order to work around this inflexibility, new and innovative cloud service models are needed that support non-IT customers with a business-first view and due to that allow support existing business processes. Furthermore, offered solutions should be able to grow (and shrink) with the size of a company. In this respect the Business Process as a Service (BPaaS) [4, 35] service model presents a promising approach for moving business processes into the cloud by enriching them with technical descriptions in order to become cloud-enabled and executable business processes. Hereby, the main challenge is to align the business processes with the cloud-based IT, and to decide how to digitalise parts or the whole business processes in order to transform the business process to cloud-enabled IT-supported workflows. The H2020 EU project CloudSocket1 [5] investigates the challenge of BPaaS on several layers to finally bridge the semantic distance from domain experts to operating cloud-enabled business processes in a multi-cloud environment.

The chapter is structured as follows. Section 2 elaborates the vision of CloudSocket and the approach we followed to achieve this. Sections 3-6 present the different building blocks of CloudSocket and show case two main research efforts, for the Design, Allocation, Execution and Evaluation Environment, respectively. Section 7 concludes the chapter with a summary.

2 The CloudSocket Approach

CloudSocket aims at providing a platform for the holistic BPaaS life-cycle, supporting the digitalisation of the business processes, their transformation into executable workflows in the cloud and the actual operation of these workflows by exploiting cloud services. In order to provide a common context, this section outlines first the CloudSocket terminology and second the CloudSocket approach to provide a holistic BPaaS platform.

2.1 Business Processes in the Cloud

Business Processes in the Cloud are defined in literature as the outsourcing of business processes into the cloud2 [1], by typically interpreting “executable business processes” as a synonym for workflows [6],[7],[8],[9]. Workflows are commonly understood as orchestrations of IT services in the cloud. CloudSocket understands business processes in its traditional interpretation as domain specific business processes that are not necessarily digitalized, but can through business and IT alignment—partly or as a whole

---

1 https://site.cloudsocket.eu/
become digitalized. In addition, the BPaaS approach also considers then the next step of digitalising business by enabling the transformation of the business process and its operation in the cloud. The management of business processes in the cloud is sometimes understood as the management of operative workflow instances in the cloud, whereas the different cloud services that are used by the workflow are pre-defined services and also managed by the whole environment. CloudSocket sees the management of business processes as a holistic approach by considering the design, allocation, execution and operation as well as the evaluation as a continuous management life cycle. Managing business processes in the cloud requires the support all of the aforementioned phases together with additional aspects such as the usage of arbitrary software services, monitoring of the services, support for different cloud service models or the cloud orchestration with its flexibility and deployment. Although deployment in the cloud follows the approach “Everything as a Service” and thus intends openness on the type of offering (e.g. storage, processing power, applications, services, workflows) the intention is to use already digitalized elements and deal with the complex and heterogeneous horizontal integration. CloudSocket understands also the vertical integration using the business and IT alignment, and introduces deployment relevant aspects (e.g. country specific restriction, business rules and strategies, preferable payment model) into the business and IT alignment. Hence, business rules on a domain level like “It is mandatory to be compliant with European law” is translated into “processing and storing of personal data are not allowed on cloud offerings that have an IP address outside of Europe, and where the registered service provider and operator (and its owner) of that cloud offering has its legal address outside of Europe”. Hence, the idea of BPaaS is not focusing on another IT service composition in the cloud, but is the support of domains specific service design and its alignment to IT services. A detailed taxonomy of all these concepts can be found in the CloudSocket wiki3.

2.2 Enabling BPaaS in CloudSocket

CloudSocket enables the holistic BPaaS life-cycle by (i) abstracting parameters from pure technical distinctions up to business and domain specific characteristics, in order to describe and distinguish cloud offerings on domain specific business processes, (ii) allocating these services and resources in order to move the business processes to the (mult-) cloud domain, (iii) reusing, exposing and publishing the defined processes through semantic enriched repositories such as a marketplace (iv) automatic deployments, executions and adaptations at the cloud domain for different levels approaches as IaaS, PaaS and SaaS, (v) evaluate the results in order to improve the original processes creating business advantages. CloudSocket approaches the transformation of a domain specific business process into a cloud-enabled business process (CeBP) by applying five levels of transformations, which are depicted in figure 1.

Level 1: Domain Specific Business Processes describe the activities of a worker, which are, in the way they are presented, not executable by a workflow engine.

---

3 https://site.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/CloudSocket+Technology+and+Terminology+Wiki
**Level II**: Abstract Workflows provide a human-interpretable intermediate level between Business Processes and Executable Workflows. They describe the intention of the business designer on such level that the development of workflows as well as the allocation to concrete cloud offerings can be performed. As a consequence, they cannot be executed, but they are offered as potential exploitable business processes.

**Level III**: Executable Workflows represent workflows that orchestrate the interaction between human tasks and software services. 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.

**Level IV**: Cloud-deployable Workflow Bundles are packaged for cloud deployment consisting of all relevant configurations, so that it can be deployed in the cloud automatically 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 Service Level Agreements (SLAs) involved, the deployment management strategies and the actual multi cloud deployment.

**Level V**: Deployed Workflow in Production reflects a bundle that has already been purchased and deployed in the cloud. The corresponding workflow is ready to be instantiated and executed by the user by starting the workflow. 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.
2.3 The CloudSocket Architecture

In order to align the aforementioned five layers, each layer has to be described in an appropriate form. Hence, there are two challenges to be met: (1) identifying appropriate representation formats for each individual layer, (2) defining appropriate interconnecting mechanisms to link the different layers. In this respect, the CloudSocket architecture introduces the vision of BPaaS Environments that together compose the CloudSocket platform. We embed BPaaS into the Business Process Management System methodology (BPMS)[10] paradigm and each of the BPaaS Environments corresponds to a particular phase of BPMS, including the support for business and IT alignment. This mapping from BPMS phases to BPaaS Environments identifies the list of functional capabilities, which consists of loosely coupled, and hence exchangeable and partly optional environments.

Figure 2 introduces the four major building blocks in the CloudSocket architecture as well as their relationships and data exchanges. Each of the four building blocks supports one phase of the BPMS paradigm when applied for business process management in the cloud. The BPaaS offerings are provided to the customer via the Marketplace, whereas the BPaaS Execution Environment enables their deployment and operation in the cloud. The conceptual challenge of bridging domain specific business processes to executable workflows that are in production in the cloud and are constantly improved via evaluation information, is performed by the other BPaaS Environments.

![Fig. 2. Architecture of CloudSocket.](image-url)
The BPaaS Environments that support the BPaaS lifecycle are: (i) the BPaaS Design Environment enabling to model business processes, business requirements (KPIs), business decisions (DMN), workflows and semantic annotations, (ii) the BPaaS Allocation Environment able to link deployable workflows to concrete services and prepare the BPaaS offering, the so called BPaaS bundle, (iii) the BPaaS Execution Environment enabling to deploy, execute, monitor and adapt the BPaaS workflow, (iv) the BPaaS Evaluation Environment which lifts monitoring information and logs to key performance indicators at the business level as well as provides additional types of BPaaS analysis, and (v) the BPaaS Marketplace being the main entry point for BPaaS customers that can browse, search, select, and purchase BPaaS bundles as well as receive billing reports for the BPaaS bundles that they have bought. More details can be found in [11].

3 The Design Environment

The BPaaS Design Environment supports the design of all parts of the BPaaS. Such a design includes the domain-specific business layer and the cloud relevant technical layer, as well as the interaction between them. The business process layer includes business process models, organisation models and document models. The IT layer consists of IT services and their orchestrations (i.e., workflow models), which reveal technical aspects of business processes and therefore are meant to be designed and understood by technical people. In order to support the alignment of business processes with IT services and workflows, we added the possibility to assign functional and non-functional requirements to parts of a business process. In contrast to business processes, workflows do not have requirements. Instead there is a reference of IT service/workflow models to specification of functional and nonfunctional properties. Elements of the graphical models can be semantically annotated and mapped to concepts of the BPaaS Ontology, i.e. semantic lifting [12]. This allows for a smart alignment of business and IT by matching domain specific business process aspects to executable IT services/workflows in a semi-automatic or automatic manner. Within this context, the BPaaS Design Environment comprise two research features the Semantic Lifting and the Context-Adaptive Questionnaire. The former addressed the need to support consistency between human and machine interpretation of BPaaS models for the Business-to-IT alignment. The second feature addressed the need to perform a cloud service/workflow discovery in an intuitive and smart manner.

3.1 Feature A1: Semantic Lifting

As described in [13], the Semantic Lifting feature includes a domain-specific modeling language (DSML) as an extension of the OMG standard BPMN 2.0 [14]. Domain-specific conceptualization enhances understanding of the targeted stakeholders [15] [16]. In our context, stakeholders are modelers such as a cloud brokers with business background. Hence, the extended modeling language enables modelers to design business processes as well as business process requirements through language constructs typically understood by business people. In [13] we introduced an application example
of the extended language on a "Social Media Campaign" business process. Requirements can be specified on a general level referring from the whole process, or more in details specifying requirements for a group of or single activities. Functional and non-functional Requirements are represented in the form of a notebook. The former can be specified in two ways: (a) by assigning hierarchies from the APQC Process Classification Framework [17], and (2) by assigning an action and an object from a predefined taxonomy. This corresponds to the convention of BPMN to name activities by a verb and a noun [18]. The verb corresponds to the action and the noun to the object. Non-functional requirements are derived from the Cloud Service Level Agreement Standardization Guidelines [19] and consolidated in workshops performed iteratively within the CloudSocket consortia. These requirements are grouped into five categories: Performance, Data Security, Support Service, Payment, Target Market. In each group there are a number of attributes that reflect business language, e.g. the performance category includes the media type (i.e. document, video, imagine and audio) and number of process execution per a time frame, which are required by the user in the business process. All requirements can be found in [13]. Similarly, the extended modeling language supports modeling of cloud services specifications as well as their orchestration, i.e. workflow. Each lane in the BPMN 2.0 was extended to target modelers with IT background that can specify technical aspects in the cloud domain. Categories are the same as for the process requirements whereas attributes in each category change. For example, the performance category has Capacity, which comprises the available data storage, simultaneous connections and service users. Specification of cloud services and workflows can be found in [13]. Both requirements and specifications appear in the form of attributes in two notebooks, respectively. Attributes can be annotated with classes and values from an enterprise ontology. The act of annotating modeling elements with an ontology is called semantic lifting.

**Smart Business and IT-Cloud Alignment.** The developed BPaaS Ontology primarily enables the smart Business and IT-Cloud alignment. The idea is that business-related attributes are mapped to IT-related attributes such that the most suitable cloud services or workflows can be automatically retrieved for the given business process requirements. For example, let's assume that a business user, while specifying the process requirements sets the number of monthly process execution to "50" and chooses "video" as a media type to upload in the process. From the ontology it is known the minimum amount of mega byte for a video (e.g. 5MB), which is considered as a technical aspect. It is now possible to calculate the minimum amount of storage capacity (technical aspect) that would be required to satisfy the requirements, i.e. 50*5 = 250MB. Hence, only cloud services or workflows that have a minimum storage capacity of 250MB should be retrieved. The calculation of the required storage capacity is performed through a semantic rule. Next, a semantic query is used to compare business process requirements with cloud service/workflow descriptions.

### 3.2 Feature A2: The Context-Adaptive Questionnaire for Service Selection

The Context-Adaptive Questionnaire aims at finding the matching cloud service(s) with the least possible number of questions. It allows specifying requirements using a domain-specific business language in a user-centric manner. The questionnaire presents a set
of questions that focus first on business process functional requirements and then on non-functional requirements. Questions relating to non-functional requirements are displayed according to a prioritisation algorithm. As it is described in [20], the algorithm considers the following: 1. the user preferences in terms of categories (i.e., Performance, Data Security, Support Service, Payment, Target Market); 2. the entropy of semantic attributes that relate to cloud service specifications, e.g., the entropy for the monthly downtime of a cloud service. The questionnaire can be applied on the whole business process first. If no service can be found, we can then move down to groups of activities, until the level of single activities.

Entropy Calculation: The entropy value determines how relevant an attribute is for the service filtering. Namely, the lower the entropy value of an attribute, the lower its relevance degree, and thus the lower the assigned priority of the related question. For example, if all cloud services in the repository have the same percentage of monthly availability, the entropy value for the attribute availability will be the lowest (i.e., ”0”). As such, in case the user chooses the Performance category, the question related to the preferred availability will get the lowest priority as it won’t filter out any services from the matching set. Hence, the question related to the preferred availability will be asked at the end. The information value of each attribute is calculated with

$$Entropy (\text{attr}_i) = - \sum_{j=1}^{J} (p_{ij} \cdot \log_2 (p_{ij}))$$

where $J$ is the total number of attribute values and $p_{ij}$ is the probability that a certain attribute value $\text{val}_{ij}$ of attribute $\text{attr}_i$ appears in a specific cloud service. By considering that this probability is independent and uniform across all attribute values, then $p_{ij}$ can be expressed as: $p_{ij} = \frac{|\text{CS}_{i_{csvalik}}|}{|\text{CS}|}$, where the nominator denotes the number of cloud services that exhibit the respective attribute value ($\text{csval}_{ik}$ denotes the value of $\text{attr}_i$ for cloud service $k$) and the denominator the number of all cloud services.

Ontology-based Metamodeling Approach. The Context-Adaptive Questionnaire adopts the ontology-based metamodeling approach introduced in [21]. That is, the web-based questionnaire interface is a graphical representation of concepts that are in the metamodel, i.e. the Questionnaire ontology. The latter contains questions and answers expressed in a non-technical language and makes use of the above mentioned BPaaS ontology. The questionnaire allows specifying functional requirements in the same way as presented for the Semantic Lifting feature, i.e. action, object and APQC category. At the starting view of the questionnaire the users can insert keywords for the object he/she is looking for, and the ontology returns the concepts matching these keywords in a list. Next, the user selects the appropriate object (e.g. invoice) and will continue by receiving the follow-up question about action and APQC category. Then, if the user has not identified the cloud service yet, he/she can further specify the non-functional requirements by choosing one of the categories from the questionnaire (e.g. performance). This enables the prioritisation algorithm to kick in over the attributes of the chosen category.

While answering the not matching cloud services are filtered out, and the cloud service list gets updated. Each time a question is answered, one or more semantic rules
are applied to convert implicit knowledge reflecting the business requirements into an explicit one. Next, in the same way as in the Semantic Lifting feature, facts from the specified requirements together with the derived facts are compared with the cloud services from the repository through semantic query. In result, the cloud services that match with the requirements are displayed. Cloud providers can insert cloud services and related specifications in the ontology repository in an easy manner through a web-based form. This solution also follows an ontology-based metamodeling approach. Besides the smart Business-IT alignment, an ontology-based metamodeling approach supports the continuous evolution of requirements. In case new concepts (e.g. service, service specification category, question or answer) have to be added or existing ones should be changed or deleted, it can be quickly done in the ontology. Changes will then be propagated to the web-based interfaces (i.e. questionnaire and cloud service form) automatically, without programming interventions.

3.3 Summary

In this section we described the two research features that are comprised in the BPaaS Design Environment. The Semantic Lifting feature enables specifying process requirements in business language and cloud service/workflows descriptions in IT language. The specification is supported by means of the BPaaS Ontology, which offers a domain-specific standard vocabulary. The BPaaS ontology also enables the smart Business-IT alignment through semantic rules and queries. This is shown in the Context-Adaptive Questionnaire that shows how the specified business process requirements can be matched to the cloud service specifications. Additionally, cloud service specifications can be easily added though a web-based form. Both the questionnaire and the cloud service form can be quickly adapted to new requirements as they adopt the ontology-based metamodeling approach.

4 The 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 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 is not only a fully executable workflow model but also a deployment plan.

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.

4.1 Feature B1: CAMEL Language

CAMEL\(^4\) [22] is a multi-domain-specific-language (DSL) covering domains/aspects such as deployment, requirement, provider, and scalability. Its development has relied on existing DSLs, like CloudML [23], plus new ones (e.g., Scalability Rule Language - SRL [24]). All these DSLs have been integrated into a coherent whole by moving them into the same modelling space and consolidating them (e.g., removing redundant concepts, renaming other concepts, etc.).

Table 1 depicts an overview of the domains covered by CAMEL and the roles involved in their modelling. As it can be seen, CAMEL mainly targets DevOps users. Such users need to express the deployment topology of their application plus technical requirements that span both non-functional and deployment characteristics. Such requirements can then be matched by respective provider models also expressible in this DSL so as to support the selection of the most optimal cloud offerings for BPaaS component deployment. Furthermore, the last column of Table 1 clearly shows the boundaries between design and runtime as well as between user-provider input and system-generated output. In this respect, the system can expand a CAMEL model via information covering the instance level. This is inline with the capability of CAMEL to support the type-instance pattern in modelling catering for the models@runtime approach.

CAMEL has been developed based on Eclipse EMF\(^5\). In particular, it has been specified as an Ecore model further constrained by inline OCL\(^6\) rules which attempt to impose the semantics of each domain and guarantee the cross-domain consistency of CAMEL models. Via the use of Eclipse EMF, the following Eclipse facilities are offered supporting: (a) the extension or adjustment of Ecore models; (b) the UI-based and programmatic editing and validation of models conforming to a meta-model specified in Ecore. For the UI-based model editing, a default tree-based editor is offered. However, PaaSage has additionally developed two other editors, namely the textual and the requirements editor. The textual editor has relied on Eclipse XText\(^7\) technology which enables specifying the concrete syntax of a DSL and the automatic generation of a textual editor conforming to this syntax. The requirements editor is a form-based web editor developed based on Eclipse RAP\(^8\).

Table 2 shows a short comparison of all CAMEL editors, where the semantics of the table columns is the following: (a) Model Validation: capability to syntactically

\(^4\) http://www.camel-dsl.org
\(^5\) https://eclipse.org/emf/
\(^6\) https://wiki.eclipse.org/OCL
\(^7\) https://eclipse.org/Xtext/
\(^8\) https://eclipse.org/rap/
Table 1. CAMEL Overview Table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
<th>Role</th>
<th>Design/ Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel</td>
<td>Top model, container of other models, application</td>
<td>user, system</td>
<td>both</td>
</tr>
<tr>
<td>Deployment</td>
<td>Application topology</td>
<td>devops, system</td>
<td>both</td>
</tr>
<tr>
<td>Requirement</td>
<td>Hw, security, location, OS, provider, QoS reqs</td>
<td>devops, business</td>
<td>design</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric, Sensors, Attributes, Conditions</td>
<td>devops, system</td>
<td>both</td>
</tr>
<tr>
<td>Scalability</td>
<td>Scalability rules, (composite) events, scaling actions</td>
<td>devops</td>
<td>design</td>
</tr>
<tr>
<td>Security</td>
<td>Security controls, attributes and metrics</td>
<td>devops</td>
<td>design</td>
</tr>
<tr>
<td>Location</td>
<td>Physical and cloud-specific locations</td>
<td>devops</td>
<td>design</td>
</tr>
<tr>
<td>Unit</td>
<td>Units of measurement</td>
<td>devops</td>
<td>design</td>
</tr>
<tr>
<td>Type</td>
<td>Value types and values</td>
<td>devops</td>
<td>design</td>
</tr>
<tr>
<td>Organisation</td>
<td>Organisations, users, roles, policies, cloud/platform credentials</td>
<td>admin</td>
<td>Pre-Design</td>
</tr>
<tr>
<td>Execution</td>
<td>Execution contexts, measurements, SLO assessments, adaptation history</td>
<td>system</td>
<td>runtime</td>
</tr>
<tr>
<td>Provider</td>
<td>Provider offerings</td>
<td>admin</td>
<td>both</td>
</tr>
</tbody>
</table>

and semantically validate a CAMEL model according to EMF and OCL semantics; (b) Aspects: the information aspects in CAMEL covered; (c) Repository Integration: capability to integrate with a model repository to support the storage of the CAMEL models edited; (d) Access Control: the capability to authenticate users and allow them to manipulate only information for which they have the rights; (e) Format: capability to support both the XMI and textual(-syntax) encoding of a CAMEL model; (f) Roles: the user roles (devops, business, admin) that can be involved in the editing of models.

Within CloudSocket, another form & web-based CAMEL editor called Bundle Designer—part of the BPaaS Allocation Environment—has been developed. It also supports the extensions performed in CAMEL in this project. Table 2 also covers the performance of this editor on the considered criteria. This editor has been developed with the rationale that the use of suitable UI metaphors does not impose the modeller to have deep or even any knowledge about CAMEL. It also includes its own repository for the management of CAMEL models.

Table 2. CAMEL Editors Comparison Table.

<table>
<thead>
<tr>
<th>Editor</th>
<th>Model Validation</th>
<th>Aspects</th>
<th>Repository Integration</th>
<th>Access Control</th>
<th>Format</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree-based Textual</td>
<td>✓</td>
<td>All</td>
<td>✓</td>
<td>-</td>
<td>XMI</td>
<td>any</td>
</tr>
<tr>
<td>Web-based</td>
<td>✓</td>
<td>All</td>
<td>✓</td>
<td>✓</td>
<td>Textual</td>
<td>devops</td>
</tr>
<tr>
<td>CloudSocket</td>
<td>✓</td>
<td>Requirements, Metric, Organisation</td>
<td>✓</td>
<td>✓</td>
<td>Both</td>
<td>business</td>
</tr>
</tbody>
</table>

PaaS/SaaS Extension. Within CloudSocket, CAMEL was extended [25] to cover the allocation dependencies across the whole BPaaS hierarchy as well as the description of PaaS and SaaS services. Through this extension, the CloudSocket platform can now
support the cross-level orchestration of cloud services during BPaaS execution. Concerning allocation, this involved covering the following: (a) the allocation of BPaaS software components to PaaS services; (b) the allocation of BPaaS workflow tasks to both external SaaS services and internal SaaS software components.

The major highlights of the CAMEL extension are the following: (i) Both external SaaS services and internal service components are covered which can be used to realise the functionality of BPaaS workflow tasks; (ii) The lifecycle of any kind of software component of a BPaaS can now be managed via either OS-specific scripts or a PaaS-based configuration; (iii) Similarly to a VM, a PaaS is explicitly modelled as a new external component kind. Such a component is associated to a set of PaaS requirements which enable the filtering of the possible PaaS service space. Such requirements could, for instance, indicate the kind of runtime framework required for the BPaaS.

**Adaptation Extension.** Most cloud application provisioning platforms cover the reconfiguration of cloud applications only at the IaaS level and mainly in terms of horizontal scaling actions. However, for more advanced cloud applications like BPaaSes that require covering multiple abstraction levels across which there exist dependencies, such capabilities are quite restrictive. In fact, the isolated and individual handling of different levels can lead to a vicious cycle of adaptation where the adaptation at one level diminishes the results of another level and vice versa. As such, it is highly recommended to support the cross-level adaptation of a BPaaS by orchestrating the execution of level-specific adaptation capabilities.

To support cross-level adaptation, there is a need to specify adaptation rules that can be triggered when certain event patterns take place and which execute adaptation workflows incorporating level-specific adaptation actions. To this end, as CAMEL is able to support such kind of rules, especially with respect to their event part, it has been extended [25, 26] to support the specification of adaptation workflows. Such adaptation workflows are represented by one composite adaptation task which applies a well-known control-flow construct to a set of other adaptation tasks. The following constructs are currently covered: (a) sequence; (b) parallel; (c) conditional; (d) switch. At the level of simple adaptation tasks, different levels have been covered, including the infrastructure, service and workflow levels, with the capturing of well-known adaptation actions, like migration and service replacement.

**4.2 Feature B2: DMN-to-CAMEL-Mapper**

As the benefits of cloud computing come along with an additional technical depth, MDE tries to reduce this complexity by providing domain-specific languages for the cloud computing domain such as TOSCA [27] or CAMEL [22]. These DSLs ease the cloud adoption by enabling a cloud application deployment model on a higher level, which can be executed by cloud orchestration tools (COTs), such as Cloudiator [28]. Still, the specification of a deployment model requires a certain degree of technical knowledge and the deployment model is static in nature. Current modelling approaches do not reflect the dynamic in changing business requirements that impact an implemented deployment model at run-time. As shown in the first lane of Fig. 1, any requirement
change leads to the remodelling of the deployment model, which is error-prone and cost-intensive.

![Dynamic deployment modelling](image)

**Fig. 3.** Dynamic deployment modelling.

In order to enable (i) dynamic and (ii) reusability of cloud application modelling, CloudSocket introduces a simple decision layer on top of the DSL modelling, enabling the transformation of business requirements into a technical deployment model at both design and run-time as shown in the second lane of Fig. 3. In the following we present the technical realisation of this decision layer in CloudSocket, namely the DMN-to-CAMEL-Mapper, while a more detailed study of the dynamic and reusability challenges in cloud modelling can be found in [29]. The DMN-to-CAMEL-Mapper prototype is based on the Decision Model and Notation (DMN) [30] as decision layer and CAMEL [31] as the cloud modelling DSL. The DMN standard provides a human-readable common notation for modelling and automating decisions. We choose decision tables (DTs) to represent decisions as these are well known to business experts. An example of a DT is shown in Table 3.

### Table 3. Image and Region DT.

<table>
<thead>
<tr>
<th>Hit Policy</th>
<th>Privacy Level</th>
<th>Provider</th>
<th>VM Image</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>String</td>
<td>String</td>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>1</td>
<td>low</td>
<td>Provider X</td>
<td>Image X</td>
<td>US</td>
</tr>
<tr>
<td>2</td>
<td>low</td>
<td>Provider Y</td>
<td>Image Y</td>
<td>Europe</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A DT consists of three column types: (i) a hit policy, (ii) an input variable set, and (iii) an output variable set. The hit policy defines the selection over overlapping decisions with policies like Unique, i.e., only a single decision will be selected or Collect, i.e., all decisions can be selected. Each input variable can potentially map to a respective output variable of a sub-decision table. Hence, there is a possible cascade of decisions leading to hierarchical decision tables. Any DT is associated with a business knowledge model (BKM) defining the decision logic, i.e., the mapping between the input and output parameters. DMN is chosen as it is an impact gaining standard and it is already well adopted on the business level. We propose a realisation based on a hierarchical set of DTs to rearrange CAMEL model artefacts as described in [29]. Both languages, CAMEL and DMN, are integrated into the meta-modelling platform ADOxx⁹, providing a modelling tool for dynamically generating CAMEL models via DMN.

⁹ [https://www.adoxx.org/](https://www.adoxx.org/)
4.3 Summary

*CAMEL* is constantly evolving as it is exploited in various running European projects. Apart from the improvement of the currently covered aspects, new ones are planned to be captured, which include the modelling of big data applications and their datasets. Furthermore, some of the CAMEL editors will be updated in line to the respective updates made to this DSL.

*Smart Service Discovery and Selection Tool* within CloudSocket, a sophisticated tool for the smart discovery and selection of cloud services has been developed\(^{10}\). This tool encompasses both previous work as well as research work conducted within the project [32–34]. Its main highlights is that it supports both functional and non-functional semantic service discovery as well as the cross-level selection of services for BPaaS concretisation with additional capabilities to handle cross-level dependencies via the use of functions as well as to fix parts of the selection problem via the consideration of best BPaaS deployments derived from the BPaaS execution history. This tool is planned to be extended according to the following directions: (a) integration with a sophisticated UI like the one of the Allocation Environment; (b) coverage of the behaviour of a service apart from its IO for functional service matchmaking; (c) the coverage of the PaaS level for the cross-level service selection.

The DMN-to-CAMEL-Mapper is exploited to semi-automatically create the technical CAMEL model with the description of the required cloud services for the BPaaS Bundle execution [35]. This enables integration of business requirements into the model creation process. The decisions that have to be made for the BPaaS bundle deployment model comprise (i) the service components to use, (ii) the cloud resource configuration for the service components, and (iii) the adaptation specification of the service components at run-time.

5 The Execution Environment

With the BPaaS Marketplace as direct interface to the customer, the BPaaS Execution Environment deploys and executes a BPaaS bundle after it was purchased. Thus, this environment actually takes care of: (a) deploying the BPaaS according to the deployment plan included in the bundle, (b) deploying the monitoring infrastructure to be used for monitoring the BPaaS and (c) importing the respective executable workflow model 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 KPIs and SLOs that have been defined for it. In case of a violation of an SLO, particular adaptation plans are executed, which are triggered via the adaptation rules that have been already defined in the BPaaS bundle. 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

---

\(^{10}\) [https://site.cloudsocket.eu/cloudsocket-innovation-shop/-/asset_publisher/l0RQCGr8dds4/content/2-4-smart-service-discovery-and-composition-tools](https://site.cloudsocket.eu/cloudsocket-innovation-shop/-/asset_publisher/l0RQCGr8dds4/content/2-4-smart-service-discovery-and-composition-tools)
workflow encapsulated by the BPaaS and producing as well as supporting the retrieval of BPaaS monitoring and assessment results for later evaluation purposes in the BPaaS Evaluation Environment.

5.1 Feature C1: BPaaS Orchestration

For the cloud orchestration, we not only investigate in the capabilities of provider-independence on the IaaS layer, but we integrate the same capability for the PaaS layer. And even more, we target an orchestration across those layers. The work done here is divided in two aspects: (i) specification, and (ii) execution of the BPaaS bundle. For the specification, we extended CAMEL to allow the description of PaaS-based application components with the following focus: (a) description of requirements on PaaS services, (b) description of PaaS types and instances mapping to certain PaaS capabilities, and (c) capabilities to configure the lifecycle of a component via a PaaS API. The respective model enhancements, apart from what is described in Section 4.1, can be found in [36] and [25].

![Fig. 4. High-level Cloudiator architecture.](image)

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

The PUL has been integrated into Cloudiator[37]. The Cloudiator framework serves as Cloud Orchestration Tool and empowers the Cloud Provider Engine. As seen in Figure 4, it consists of colosseum, which offers a RESTful interface to its model, which make us of registries, an application repository, and workers that handle the internal jobs, such as application deployment or resource provisioning. Further, Cloudiator provides abstraction layers for the Cloud providers on different levels: (i) Sword abstracts the IaaS Clouds, and (ii) Dagger abstracts the PaaS Clouds by utilizing PUL. Therefore Cloudiator is able to handle PaaS- and IaaS-based services simultaneously and interchangeably. The management of services was designed independently from their Cloud level. All features that map to the origin IaaS orchestration were also mapped to the PaaS layer. For example, the monitoring system[38] was in turn unified to work across Cloud providers and levels. Also the application description was extended during the project[39].
5.2 Feature C2: Cross-layer Adaptation Management

Identical to the aforementioned Cloud orchestration efforts, the work was split into (i) specification, and (ii) execution. The sub-model SRL (Scalability Rule Language)\[40\] of CAMEL was extended in its scope to cover additional cloud levels as well as adaptation action workflows. When defining behavioral rules in CAMEL, the user can now define adaptations, such as migration, service replacement, cloud burst, or task modifications, in addition to the well-known scaling actions. A more detailed description can be found in \[26\].

To execute these adaptations a first implementation of the CAMEL extensions were realized in the Cloudiator framework. The already existing integration abstractions regard cloud provider and layer allowed the seamless integrations of the adaptation actions. As the adaptations are described in workflows, we integrated a simplistic workflow processing component to realize the execution of the aforementioned adaptation workflows\[25\].

To realize a migration scenario, the user or an optimization tool would define an adaptation workflow with three steps: (i) instance creation, (ii) state copy, and (iii) instance deletion. In the first step a new instance of a component is created in the target cloud of the migration. The state copy can be user-defined scripts that copy data to a new location. The third step cleans up the old instance and free the resources. As Cloudiator already caters for the wiring of components the new instance is registered in the service chain as defined by the user.

5.3 Feature C3: The BPaaS Marketplace

For BPaaS customers, e.g. SMEs or other organizations, the CloudSocket Marketplace is the main entrypoint to the CloudSocket platform and the offerings of a certain CloudSocket Broker. CloudSocket Marketplace can be found online\[11\]. Anybody can access the URL and browse the business process offerings of CloudSocket Brokers. Figure 5 shows a screenshot of the marketplace. Here you can see a list of processes. For each process you can see the definition diagram, implementation workflow and Service Level Agreement.

There are two types of processes at the moment in the marketplace: (i) active processes and (ii) abstract processes. The active business process can be bought, installed and run. The buying process is as usual as for any household item: the user sees the process she likes, adds it to her shopping cart and when she is content with his selection, she makes the purchase. She can add as many processes as she wants. After that she can proceed to checkout and pay either by PayPal or bank order. For the user to be able to place the order she needs to be logged in. After purchasing the processes they will be provisioned automatically for her and she will receive by email the address where her processes are running and can be accessed. She can execute them as many times as she wants. The abstract processes cannot be bought and installed, but they allow the user to express her interest in it by clicking the I want it button and starting a discussion with the broker.

\[11\] http://csmarket.ymens.com
5.4 Summary

Throughout the project, we achieved to develop and enhance the cloud orchestration tool Cloudiator, with new components that allow us to close gap between commercial and research tool, to allow management and adaptation seamlessly not only between cloud layers, e.g. IaaS and PaaS, but also between different cloud providers. For more information on other research efforts in this respect during CloudSocket, see the respective deliverable[25]. Following to the prototypical implementations of the components, we aim for a stable implementation and the verification of their need to achieve highly adaptable BPaaS instantiation. This can be done by case studies that target the comparison of current commercial service deployment approaches, and our cross-layer adaptation management that is capable of switching through provider-independent service offers in a way that is timely reasonable and still cost-efficient, e.g. with a minimum overhead of replicated services during adaptation process. The PaaS Unified Layer is enlarging its scope of applicability by supporting Container as a Service (CaaS) providers and Function as a Service (FaaS) providers. Concretely, the deployment of Docker containers on OpenShift3 is being developed as part of the AGILE project12, while Amazon EC2 Container Service is the next provider on the roadmap. Regarding FaaS providers, expected platforms to be supported are Amazon Lambda and OpenWhisk, which are planned to be developed on future projects. Cloud service customers have to deal with the complexity of integration and maintenance of cloud services offered by multiple vendors and providers. This brings in the need for an intermediary who can simplify the process of Cloud Services [41] consumption. In addition it is difficult for the customer to identify all the services she requires. Cloud Services requirement

12 http://agile-iot.eu/
detailing, identification, discovery, integration and maintenance will always remain a major issue for customers due to non-standardized features, description and type of these services. All of these issues are taken care of by this intermediary entity called Cloud Services Brokerage (CSB) service. By definition, a cloud broker is a third-party individual or business that acts as an intermediary between the purchaser of a cloud computing service and the sellers of that service. The CSB we plan on developing is an enhanced type of electronic marketplace for Cloud Services, which aggregate multiple service providers in a single location for customers to evaluate and buy Cloud Services. It also simplifies various other processes of maintaining, administration, billing and payment between external vendors and customers. The Marketplace as a Service for Cloud Services (MaaS-CS) is a single-point, aggregated and customized solution and owns the responsibility of security, governance and quality of all the services provided. Figure 6 describes the proposed model for MaaS-CS.

![Proposed Reference Model for an MaaS-CS and CSB.](image)

**6 The Evaluation Environment**

The evaluation of a BP has the main goal to derive added-value knowledge which can be exploited in the optimisation of this BP. Such an evaluation usually includes two main functionalities: (a) the assessment of Key Performance Indicators (KPIs); (b) the drill-down of KPIs in order to find the root-causes of a certain high-level KPI violation. Such an evaluation is supported by a corresponding system which employs certain techniques in order to realise the analysis functionalities needed. By considering the current state-of-the-art BP systems evaluation systems in the literature, we can detect the following drawbacks [42]: (a) not all possible abstraction levels are covered and
their dependencies; (b) semantics are not captured leading to imprecise evaluation results; (c) the evaluation is performed via a static set of KPIs which is hard to extend and leads to huge re-engineering effort; (d) there is inflexibility with respect to the way a certain KPI metric can be evaluated as many factors need to be take into account in this evaluation; (e) the drill-down techniques involved mainly rely on machine learning which indicates that there is a sufficient monitoring & evaluation history available in order to reach the appropriate accuracy level.

Based on the above analysis and by also catering for the evaluation of BPaaS, i.e., BPs in the cloud, the CloudSocket project has developed a BPaaS Evaluation Environment [43] which resolves the above drawbacks. Above from relying on semantics to increase the accuracy level, this environment also enables the flexible evaluation of KPIs by allowing the user to smartly explore the possible KPI metric space. In addition, it offers two forms of KPI drill-down which rely on concrete dependency hierarchies between KPIs and their metrics, respectively, thus being able to be more precise as well as not require the existence of any relevant monitoring/evaluation history. In the following, there is an analysis of how information is collected and integrated to support the BPaaS Evaluation while the two forms of KPI analysis are detailed. Additional information about the Evaluation Environment and what is the extra evaluation functionality that it supports can be found in [43].

6.1 Feature D1: Information Harvesting

Any BP evaluation system cannot offer its main capabilities without possessing the right information that supports them. In the context of a BPaaS, such information needs to span: (a) dependency hierarchy information about the BPaaS; (b) monitoring information; (c) service description information; (d) organisational information. All these information kinds need to be appropriately connected and linked in order to be able to support the KPI analysis & drill-down. In addition, they need to be semantically lifted in order to raise the accuracy of the analysis/evaluation facts produced.

To this end, our BPaaS Evaluation Environment relied on two main ontologies to support the structuring and linking of the above information kinds. The evaluation ontology is able to cover BPaaS hierarchy, service description and organisational information. This is done at both the type and instance level in order to be able to continuously keep up with the current state of the BPaaS system. The capturing of such kinds of information also provides support to other BPaaS evaluation tasks, like process and organisation mining, due to the capturing of information usually pertaining to the process log which includes which actors were involved in the execution of a certain BPaaS workflow task.

On the other hand, the KPI ontology [42] covers the modelling of KPIs and respective monitoring information through the careful extension of a prominent non-functional service description language called OWL-Q [44]. This ontology is quite expressive as it is able to both specify KPI and as well as KPI metric dependencies. In addition, due to the use of OWL-Q as a basis, it covers well all required measurability aspects, such as the metric, attribute, value type and unit ones. Furthermore, the metric computation specification has been enhanced with the capability to exploit both internal
and external to the BPaaS management system information from any kind of information source. In addition, this ontology links KPIs to certain goals which can then enable goal-based analysis in order to be able to infer how well the operational, strategic and tactical goals of the system are satisfied according to the current performance level of the BPaaS. Finally, such an ontology does not require any technical knowledge about the underlying evaluation framework. On the contrary, as ontologies are conceptualisations of certain domains, it is more close to human perception.

By relying then on these two ontologies, an information harvesting system has been developed which is able to draw information from multiple components of the CloudSocket platform as well as semantically lift and structure this information according to the two aforementioned ontologies. The resulting information is then stored in a Semantic Knowledge Base (Semantic KB) so as to support the various analysis tasks offered by the BPaaS Evaluation Environment. As it will be shown in the next subsection, the population of the KB enables the issuing of semantic queries that can be used for the evaluation and drill-down of BPaaS KPIs.

6.2 Feature D2: KPI Evaluation & Drilldown

In order to resolve the main issues analysed above for BP performance evaluation, we have developed a KPI analysis module [42] in the BPaaS Evaluation Environment which supports both KPI evaluation & drill-down. Such a support comes in a very flexible way via the use of OWL-Q which enables the business expert to specify KPIs in a high-level such that there is no tight coupling with the semantic technologies exploited mostly concerning the underlying Semantic KB. Furthermore, the business expert can experiment with various elements that comprise a KPI to more smartly browse the possible KPI metric space thus leaving enough space for creativity. KPI analysis can also be performed either for the current situation or across the whole measurement history of a KPI which can be limited based on a user-provided time range. In addition, two forms of KPI drill-down are offered based on the respective relations captured for KPIs and KPI metrics.

KPI evaluation is flexibly performed by transforming the OWL-Q-based specification of a KPI into a SPARQL query which is then issued in the Semantic KB to produce the respective value for the KPI metric involved. This transformation is performed by considering that KPI metrics are composite and thus need to be computed from aggregation formulas which can involve other metrics. To this end, the following issues were appropriately handled: (a) the BPaaS dependency hierarchy should be accounted to select only the appropriate measurements for the aggregation; (b) the computation should go down to the level of metrics for which measurements are already stored in the CloudSocket platform. More details about the transformation algorithm can be found in [42].

All kinds of KPI drill-down supported rely on the KPI evaluation functionality. The first KPI drill-down form is based on the capturing of parent-child relations between KPIs. Such relations indicate a certain kind of dependency between KPIs to be exploited for root-cause analysis purposes. In particular, they can be used to indicate that a violation of a parent KPI can be due to the violation of one or more child KPIs. For
instance, consider the availability of a BPaaS. Such an availability depends on the availability of all BPaaS service components as well as that of the Workflow Engine executing this BPaaS. In this respect, by conditioning the availability of all these components (thus constructing respective KPIs) and knowing their influence of the high-level KPI we can then establish the right parent-child dependency relations. By relying on the computation independence of the KPIs involved, the drill-down algorithm of the first form attempts to evaluate the KPIs in the hierarchy in a top-down manner by using the KPI evaluation functionality.

The second KPI drill-down form relies on the parent-child relations between KPI metrics which are incarnated in the KPI metric computation formulas. In other words, the parent KPI metric is calculated based on its computation formula from its child KPI metrics, thus giving rise to an hierarchy of KPI metric dependencies. The added value of exploiting such an hierarchy is that there is no need to place conditions over all metrics in the hierarchy. Thus, a business expert might pose thresholds over the high-level metrics in this hierarchy but as we go down to a more technical level, the placement of conditions over more technical metrics is impossible as being out of the expertise scope of the expert. As such, this kind of KPI drill-down enables going down to the lowest possible level and really finding out problematic measurements for the lowest possible KPI metrics. In this respect, the first kind of KPI drill-down is more suitable for a high-level overview of the KPI dependencies but, when the need to have this overview more detailed, the second KPI drill-down form can be exploited. The main logic of the algorithm for the second form of KPI drill-down is that it expands first the whole metric computation tree to reach the level of leaf metrics (for which measurements already exist in the system) and then processes this hierarchy in a bottom-up manner in order to produce the measurements for the intermediate and top-level metrics.

6.3 Summary

The research & development work in the CloudSocket project has led to the delivery of an innovative environment which offers different kinds of analysis over a certain BPaaS. This environment offers a greater flexibility with respect to state-of-the-art frameworks in terms of KPI evaluation while it relies on a high-level language which is more suitable to the business expert. By combining KPI analysis with other forms of added-value BPaaS evaluation, which include: (a) the derivation of process-mining knowledge; (b) the detection of event patterns leading to the violation of KPIs; (c) the discovery of the best deployment(s) for a BPaaS, the broker is equipped with a powerful environment which can really produce business intelligence knowledge to be used for optimising the design and allocation of the offered BPaaSes. In this respect, the broker can really increase his/her gains, reduce costs as well as supply BPaaS products which might offer better service levels from the competition.

Apart from thoroughly evaluating the developed environment, it is also planned to extend it according to the following directions: (a) exploitation of state-of-the-art process mining techniques for the production of additional kinds of knowledge focusing on organisational and business rule aspects; (b) capability to dynamically generate and improve BPaaS adaptation rules based on the event pattern detection functionality; (c)
coverage of the PaaS level in best deployment discovery; (d) extension of SPARQL query evaluation to better handle some statistical operators.

7 Conclusion

During the life-time of the CloudSocket project, we pursued different research efforts to achieve the management of business process in a hybrid cross-cloud environment. We have proposed a definition for business process in the cloud, by characterizing the term Business Process as a Service and specifying its management lifecycle [45].

For the design phase of the BPaaS lifecycle, we developed a taxonomy for BPaaS and methods that enable the supportive finding of services based on business requirements as well as a semantic approach encompassing tool-based supportive methods to specify and find suitable business processes in the Cloud. These are solutions to implement semantic lifting for the Business-to-IT alignment. Concerning the allocation phase, we extended the existing Cloud DSL CAMEL to allow the definition of services across Cloud levels, and also bind them to tasks of a BPaaS workflow. Also, tools to enact this allocation and support business expert to dynamically define technical workflows were achieved. Furthermore, CAMEL was extended with the capability to specify adaptation rules that drive the adaptive behaviour of a BPaaS at runtime. Such an extension was accompanied with a corresponding cross-level BPaaS adaptation framework able to realise the triggering of the adaptation rules and the execution of the respective BPaaS adaptation workflows. In order to enable the needed flexibility during the execution phase, we enhanced the Cloud orchestration tool Cloudiator to realize cross-layered orchestration, and adaptation. The extensions (of both Cloudiator and CAMEL) are further elaborated and developed in other projects, such as Melodic\textsuperscript{13} or CloudPerfect\textsuperscript{14}. The specific PaaS abstraction layer is also used in the AGILE project\textsuperscript{15}. In addition, we created a user-friendly marketplace to allow the enactment of cloud services from a high-level business perspective.

The evaluation phase was supported via the introduction of a BPaaS Evaluation Environment comprising: (a) the Hybrid Business Dashboard which enables users to enact the evaluation functionality as well as depict through appropriate UI metaphors the results of this enactment; (b) a set of components agglomerated into the form of a Business Intelligence Analysis Tool which support the harvesting and semantic linking of BPaaS monitoring and allocation information as well as the derivation of business intelligence knowledge. Through this environment, the whole BPaaS management lifecycle loop is closed and the derived business intelligence knowledge can lead to improving the design, allocation & adaptive behaviour of a BPaaS.

Throughout many publications and demonstrations, it was possible to fertilize the research endeavors across all areas within the Cloud-based Business-IT alignment. With a novel approach on Continuous Integration in international projects[46], we successfully managed the integration of a multitude of research assets. Even more, our results are needed features in current, on-going projects, which proves our efforts as a success.

\begin{thebibliography}{9}
\bibitem{13} http://melodic.cloud/
\bibitem{14} https://cloudperfect.eu/
\bibitem{15} http://agile-iot.eu/
\end{thebibliography}
The interested reader may want to have a look at our public deliverables, which are published online. We published webinars and explanations to the main results of our research and development efforts in our YouTube channel.

Acknowledgements. The research leading to these results has received funding from the EC’s Framework Programme HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References


16 https://dev.cloudsocket.eu/web/guest/deliverables
17 https://www.youtube.com/channel/UC8qbCYS49FIG7S8j5AABpAA