Towards Business-to-IT Alignment in the Cloud

Kyriakos Kritikos\(^1\), Emanuele Laurenzi\(^{2,3,4}\), and Knut Hinkelmann\(^{2,3}\)

\(^1\) ICS-FORTH, Heraklion, Greece  
kritikos@ics.forth.gr
\(^2\) FHNW University of Applied Sciences and Arts Northwestern Switzerland  
\{emanuele.laurenzi, knut.hinkelmann\}@fhnw.ch
\(^3\) Department of Informatics, University of Pretoria, South Africa  
knut.hinkelmann@up.ac.za
\(^4\) IPM-FHSG University of Applied Sciences St. Gallen, Switzerland  
emanuele.laurenzi@fhsg.ch

Abstract. Cloud computing offers a great opportunity for business process (BP) flexibility, adaptability and reduced costs. This leads to realising the notion of business process as a service (BPaaS), i.e., BPs offered on-demand in the cloud. This paper introduces a novel architecture focusing on BPaaS design that includes the integration of existing state-of-the-art components as well as new ones which take the form of a business and a syntactic matchmaker. The end result is an environment enabling to transform domain-specific BPs into executable workflows which can then be made deployable in the cloud so as to become real BPaaSes.

Keywords: BPaaS, service, design, discovery, selection, alignment, mediation

1 Introduction

Due to intense market competition, organisations can survive only if they offer services that are either innovative or exhibit a better quality than their competitors. However, by owning a limited infrastructure and continuously requiring to improve the existing business processes (BPs) leads to reaching certain impassable limits. Moreover, the infrastructure maintenance, operation and management costs can be quite prohibiting, especially for small or medium enterprises.

Fortunately, nowadays, cloud computing can become the medium via which organisations can acquire cheap, commodity resources on-demand while also being able to achieve certain benefits, including: outsourcing infrastructure management with reduced cost, flexible resource management, and elasticity. Such benefits can certainly enable improving and more optimally controlling BP performance.

However, as cloud computing deals only with the infrastructure level, an organisation now faces the problem of how to align the business with the IT level. This is a widely known and hard to solve problem for which no complete solution exists. Moreover, many organisations do not have the expertise and know-how to use and combine the cloud services offered.
The above problems can be solved by combining BP management with cloud computing to realise the BP as a service (BPaaS) paradigm so as to enable migrating as well as more optimally managing BPs in the Cloud [6,31,32]. However, such a combination is by far not trivial as: (a) there are multiple levels involved from a BP down to IaaS services incorporating many entities that need to be properly managed; (b) there are multiple realisation opportunities for a BP in the Cloud constituting a quite large solution space; (c) it is difficult to align business with technical requirements and match both requirement kinds to respective cloud service capabilities; (d) the use of different services at different abstraction levels requires employing a cross-level BP monitoring and adaptation approach to handle the various issues that might be involved during BPaaS execution.

The first three challenges seem to concern most the BP design activity with the first one being also applicable to the rest of the BP lifecycle. Such challenges can then be transformed into the following more concrete challenges in the context of BPaaS design and thus of this work, seen also as the main research questions to be answered: (a) how to map a BP to a technical workflow with a suitable automation level; (b) how to align business terms and requirements with technical ones to drive the selection of the most suitable services to be then integrated into the workflow; (c) how to deal with the service incompatibility problem effectively to guarantee the correct execution of the designed workflow. Such a problem relates to checking the syntactic compatibility of messages exchanged between two or more selected workflow services.

To realise the vision of BPaaS, the CloudSocket project\textsuperscript{5} is delivering a platform that unifies together environments that support different BP lifecycle activities. This paper focuses on our contribution towards enabling the BPaaS Design Environment to deal successfully with the 3 aforementioned challenges. This translates to introducing an innovative architecture which includes suitable components that support: smart and semantic service discovery at both the business and technical levels, the optimal cross-level service selection, the mapping between business and technical requirements and the capability to mediate between the execution of two or more services to achieve message-level compatibility. In result, the developed environment enables the BPaaS provider to transform the initial business functional and non-functional requirements that match the necessities of potential BPaaS customers into an executable workflow that can then be made deployable into the cloud by exploiting other CloudSocket environments.

The BPaaS Design Environment has been built via exploiting state-of-the-art components as well as two innovative ones that constitute another contribution of this paper. The first component, the business matchmaker, enables to find services that support the user functional and non-functional requirements at the business level via following a novel questionnaire-based approach. Such services are then filtered and selected by employing state-of-the-art technical matchmaker and service selection components. The service selection component relies

\textsuperscript{5} www.cloudsocket.eu
on the second novel component, the syntactic matchmaking one, able to infer the message-based compatibility between two or more selected services and produce a respective mapping specification. Such a specification can then be exploited by a service mediation service to support the compatible message transformation between services and thus guarantee the smooth operation of the BPaaS workflow in which this mediation service is integrated.

This paper is structured as follows. Section 2 shortly analyses existing research results, some of which are exploited in the production of the BPaaS Design Environment. Section 3 analyses the main architecture of this environment by also explaining the main functionality and role of each component involved. Sections 4 and 5 detail the two main novel components involved in the architecture, the business and syntactic matchmakers. Section 6 introduces a use case to demonstrate the main benefits of the proposed environment as well as to validate it. Finally, the last section concludes the paper and draws directions for further research.

2 Background

2.1 Business-to-IT Alignment

The Business-to-IT alignment issue in the Cloud typically refers to the gap between business requirements and technical solutions [12]. Cloud offerings are described technically making it hard for business people, usually understanding a higher-level business language, to properly assess the best fitting cloud solution [33]. Achieving to identify suitable cloud solutions in this context requires specifying requirements for and capabilities of a service in both a business and IT language. To ensure understanding and transparency of knowledge, it is a common practice to represent it in models [11] [29]. Models abstract away from complex realities and become fit to their purpose, i.e., to precisely model the respective intended domain. In [13] we already adopted a model-driven approach where an extension of BPMN 2.0 allows modeling both business process requirements in a business language and workflows/cloud services in a technical language. The approach includes the translation of the business language into the technical one enabling the comparison between process requirements and workflow/cloud service specifications. In result, the matching workflows/cloud services are identified. The translation and the comparison are performed by semantically enriching models with ontologies. Hence, models become also machine-interpretable. In modeling research, this practice is well-known as Semantic Lifting. [2] defines it as "the process of associating content items with suitable semantic objects as metadata to turn unstructured content items into semantic knowledge resources". Semantic Lifting shifts the purpose of modelling beyond transparency and communication [14]. The interpretable knowledge base (i.e., ontology) allows for automation on models [13]. For example, in [8] it is described an ontology-based early warning system for automating the assessment of supply chain risks, while in [7] ontologies are combined with a case-based reasoning approach to automate the support on workplace learning. Closer to our current problem,
Faria [9] introduced an ontology called AgreementMarkerLight (AML) for the automatic identification of correspondences between process model activities. Similar approaches are described in [1]. However, they all concern mainly with one dimension only, i.e. the process matchmaking. This approach is not sufficient in the context of the process to workflow (or workflow fragments) matching as a business process is usually far less detailed than a workflow. Hence, only one matchmaking dimension would result in a quite low accuracy for the matching. In this context, our approach promises to deliver better matching accuracy than existing solutions. Firstly, a domain-specific conceptualization is performed on two dimensions, where one targets business process users, while the other targets the IT service experts. This allows designing domain-specific models that capture the appropriate domain knowledge on each of the two dimensions, business and IT. Benefits of domain-specific conceptualizations are well established within the modeling community [25] [17] [10]. Secondly, this work builds upon the findings in [13] but adopts a different perspective of Business-to-IT alignment in the Cloud. Namely, there is a shift from the language translation to the mapping of values between requirements and specifications on both the business and IT dimensions, separately. Hence, the Business-IT alignment paradigm is applied sequentially by further refining the results from the business layer in the IT layer. For this, three matchmaking components are proposed: (a) the business and (b) technical matchmakers enhanced with formal semantics for the machine-interpretation plus (c) the syntactic matchmaking component. The combination of these three components, from both business and technical terms, allows identifying the most suitable cloud services that will eventually form a workflow.

2.2 Technical Service Matchmaking

Technical service matchmaking involves functional and QoS matching. Functional matchmaking has been traditionally focused on I/O-based matching [18, 26] while QoS matching takes the view of QoS as conformance [20] and employs different kinds of techniques (semantic reasoning, constraint solving & mixed) [21] to infer if the solution space of the service is included in the solution space of the request. While most of the proposed work focus on one aspect individually, some approaches have attempted to consider both aspects simultaneously [3, 16]. However, they usually consider only the sequential pattern to combine the matching in both aspects and do not employ semantic techniques, thus not exhibiting the right performance and leading to results of medium or low accuracy.

As such, our previous work [23] has explored different ways the 2 matchmaking types can be jointly performed: (a) sequential combination (with two different variations); (b) parallel combination; (c) subsumes combination (construction of an hierarchy of service advertisements that are connected with the subsumes relation to exploit the fact that if a parent is matched by a request, then also its descendants will be matched). After an experimental evaluation
of these combinations, it was shown that the parallel combination of aspect-specific matching approaches leads to the best possible results with respect to performance, as matchmaking accuracy is exactly the same in both approaches.

Our approach exploits two aspect-specific matchmakers, a functional and a non-functional. The functional matchmaker relies on the combination of I/O-based and IR-based matching and has been developed in the Alive project [4]. This matchmaker exploits a smart graph-based structure able to dynamically tolerate changes in domain ontologies (i.e., the ontologies via service I/O is annotated) while supplies almost constant-in-time query operations over the graph (e.g., to discover all ancestors in the ontology for a specific service input / output parameter). I/O based matching is performed in a two-step manner. First, the output of the service and its request are considered for the matching which results in different kinds of match categories being produced that rely on the relations between the semantic concepts of the the service and the request output parameters. Then, input-based matching is performed on the fly to produce the final matchmaking results which can then be ranked according to different criteria.

The unary matchmaker [21] follows a hybrid approach in QoS service matchmaking. First, it considers ontological-based specifications of services to align them based on their QoS terms (their QoS metrics in particular). Then, it performs the respective filtering in a step-wise manner by considering each QoS term individually each time. As unary constraints are assumed to be involved in the service offer and demand, the matchmaker employs smart structures to support parameter-based filtering which results in ultra fast matching time. In fact, by being compared with other QoS matchmakers, this matchmaker was shown to be the fastest in both service matchmaking and registration and the most scalable without affecting in any case the accuracy of the produced matchmaking results.

2.3 Service Selection

Various service selection algorithms have been already proposed. However, they usually consider only one abstraction level by also neglecting semantics, thus producing results of imperfect accuracy. The accuracy is further reduced by considering that some algorithms employ smart but non-optimal solving techniques, like Genetic Algorithms or heuristics to accelerate the service selection time.

By considering now that the service selection for a BPaaS includes different abstraction levels along with design choices (e.g., either to select an external SaaS or deploy an internal software component in an IaaS to realise the functionality of a BPaaS workflow task), we have developed a constraint-satisfaction-based algorithm [22] which resolves these two issues while also catering for other important features including: (a) the consideration of multiple optimisation objectives by employing the Analytic Hierarchy Process (AHP) [27] and Simple Additive Weighting (SAW) [15] techniques; (b) the addressing of non-linear constraints; (c) the capability to bridge the gap between the two levels (SaaS and IaaS) via the insertion of functions that derive the QoS at the SaaS level based on the respective capabilities selected at the IaaS level; (d) the ability to deal with
overconstrained user requirements by employing smart utility functions that allow the slight violation of the user requirements so as to produce at least one solution; (e) the capability to consider all possible execution paths in the BPaaS workflow and not just one (e.g., the critical one); (f) the capability to consider ranges of possible values for the service offerings (suitable when not a single value can be guaranteed for a certain QoS parameter); (g) the capability to consider dependencies between QoS parameters at the same level which will enable a more accurate evaluation of the respective solutions; (h) the capability [19] to accelerate the solving time by fixing parts of the problem to particular partial solutions by relying on the BPaaS execution history.

State-of-the-Art Advancement. As can be seen from the above analysis, the proposed BPaaS Design Environment incorporates not only existing state-of-the-art components in order to support some critical tasks, like holistic technical service matchmaking, but also some new innovative ones. The business matchmaker follows a dynamic questionnaire-based approach which enables business users to answer a minimum set of questions before the respective mapping of an initially designed BP to a set of services, able to realise its functionality, can be produced. Such an approach is more natural and user-intuitive as it employs questions mapped to a natural language with terms drawn from the business domain. Moreover, it supports the production of a minimal set of services to be further filtered and selected based on technical requirements such that the solution space is significantly reduced and thus service discovery time is more optimised.

The syntactic matchmaker enables the production of a really executable workflow via the suitable integration of services at the technical level based on their message compatibility. Innovation comes via the production of mapping specifications which are then exploited by introducing mediation tasks in the generated workflow which map to the usage of the facilities of a mediation service. In this respect, not only a service-based workflow is generated but it is guaranteed to work correctly at the message level via the incorporation of a mediation level within the control flow between the selected services. This constitutes a genuine feature of the proposed framework that is not exhibited in other approaches which rather consider that the message compatibility should be introduced manually by the modeller him/herself. The coverage of the cloud level is another new feature of the proposed BP design framework which is not currently exhibited by the state-of-the-art. Last but not least, our framework is able to address all layers involved in the BPaaS system along with their dependencies thus able to produce a more complete and optimal BPaaS design product.

3 Architecture

The creation of the BPaaS Design Environment was underpinned by the design science research (DSR) methodology of Vaishnavi and Kuechler [30]. First
existing approaches and recent literature were screened, which address the Business/IT alignment in the Cloud. Then, the EU research project CloudSocket created the settings to contribute to the awareness of the problem, that is, application scenarios were created in workshops, where both industrial and scientific experts were involved. The results and insights were useful to suggest the first draft of the BPaaS Design Environment. After the continuous development of several versions, the most refined draft was implemented in a web-based solution. Finally, as section 6 describes, the validation took place with respect to the most agreed application scenario among the members of the CloudSocket consortium.

The BPaaS Design Environment follows a model-driven and semantics-aware approach to support the business-to-IT alignment in the cloud. Such an approach comprises 3 main transformation steps: (a) BP-to-business-services; (b) business-services-to-technical-services; (c) BP & technical-services-to-executable-workflow. As such, the approach guarantees the technical feasibility of the solution produced by employing a two-step service matchmaking process at both the business and technical level as well as a service selection approach that is syntactic-compatibility-aware at the technical level.

To achieve its main goal, the environment exhibits an architecture, depicted in Figure 1, comprising 8 main components that are now analysed in detail. Some of the components correspond directly to some of the aforementioned steps while others play a supporting or an orchestration role. The architecture also spans the well-known 3 levels of user interface (UI), business logic and persistence.

**BPaaS Designer (BD).** It represents the main point of interaction with the user in the context of BPaaS design. It enables specifying both domain-specific BPs and respective executable workflows. It also guides users in providing suitable input to support the alignment and transformation of BPs into workflows.

**Orchestrator (Orch).** It is responsible for orchestrating its underlying components to handle the requests issued by the BPaaS Designer.
**Business Matchmaker (BM).** It is responsible for matchmaking the cloud services registered in the Knowledge Base based on business requirements derived from a questionnaire-based approach which is explained in the next section.

**Technical Matchmaker (TM).** A technical, functional and non-functional service matchmaker. It exploits state-of-the-art aspect-specific matchmakers in a parallelised fashion according to the approach in [23].

**Service Selector (SS).** It [22] is responsible for producing a concrete optimal solution for the service-based workflow at hand by considering the users technical non-functional requirements while also attempting to maximise the message compatibility between services. The latter is derived through using the next component.

**Syntactic Matchmaker (SM).** It is called dynamically by the SS while solving the service selection problem for the BPaaS workflow to find the compatibility between the next and all previously selected services in each execution path where such a service participates. Such a compatibility maps to the message compatibility [24] between the output parameters of the already selected services and the input parameters of the next service. When an incompatible solution is constructed, SS can backtrack and check another solution. To smartly deal with cases where the same call is issued, e.g., due to deep backtracking, SM stores the call results so as to immediately answer it. The mapping of the output parameters to the input ones of the next service is also recorded to enable updating the BPaaS workflow via a mediation service, as performed by the next component.

**Workflow Updater (WU).** It is responsible for updating the BPaaS workflow by performing the following actions for each workflow’s execution path: (a) we replay the solution construction in each path to obtain the respective mapping of the current service in the path from the SM; (b) we introduce a mediation service within the workflow, immediately before the current service, which takes as input the current output parameter set and the mapping specification and produces as output the input parameters of the current service.

**Knowledge Base (KB).** It includes all necessary and sufficient information to support all reasoning tasks executed in the system, including the business and technical service matching and selection for the BPaaS workflow at hand.

### 4 Business Matchmaking

The **Business Matchmaker** allows specifying requirements in a more user-centric approach than the one in [13]. It relies on a context-adaptive questionnaire that guides the user via a set of questions that reflect BP functional and non-functional requirements. Follow-up questions are displayed based on the result of the context-adaptive algorithm that considers: (a) the user preferences in terms of categories (e.g., *Performance* rather than *Data Security*)); (b) the information value (also known as entropy) of semantic attributes reflecting cloud service specifications at the business level, e.g. how distinguishing an attribute such as *monthly downtime* is for the service filtering. Namely, the higher the entropy value of an attribute, the higher its service distinguishability degree, and thus
the higher the assigned priority of the related question. This approach leads to the least possible number of questions being answered thus reducing the overall business service matchmaking time. The idea is that the questionnaire can be applied on the whole BP first. If no service can be found, we can then move down to groups of activities, until the level of main single activities.

4.1 The Context-Adaptive Questionnaire

The Context-Adaptive Questionnaire relies on the BPaaS ontology in [11]. Questions focus first on functional requirements and then on non-functional ones. The questionnaire enables to specify functional requirements in two ways:

- by asking the user to insert an action and an object from a predefined taxonomy in the BPaaS ontology. This corresponds to the convention of BPMN to name activities by a verb (i.e., action) and a noun (object) [28] whose combination provides the “what-is-about” knowledge.

- by asking the user to insert the most suitable category from the APQC Process Classification Framework.

Next, the user can choose one of the 5 non-functional (NF) categories: Data Security, Payment, Performance, Service support, and Target Market.

The NF categories were derived from the Cloud Service Agreement Standardisation Guidelines [5], an outcome of the European 2020 initiative "Digital Agenda for Europe" published to standardize and streamline the terminologies and understanding of cloud services. The NF categories were subsequently discussed and validated within the CloudSocket consortium. In result, a respective set of questions and sub-questions were derived out of them. For instance, the Performance category contains the following questions:

- What is your preferred monthly downtime in minutes?
  Possible answer: 30 minutes
- What would you like to upload?
  Possible answers: Audio MP3, Video MP4, PDF and/or Microsoft Office documents
- Should the process be executed on a daily, weekly, monthly or yearly basis?
  Possible answer: On a weekly basis
  • Sub-Question: How many times should the process be executed?
    Possible answer: at least 10 times
- What is your favorite response time level?
  Possible answer: High, Medium or Low
- How many simultaneous users should the cloud service support?
  Possible answer: at most 10

For each question, we have distinguished among four types of answers as: (1) single-answer selection; (2) multi-answer selection; (3) search-insert; (4) value-insert. Value- and search-insert require input from the user. While the former enables the user to insert values (e.g., the aforementioned downtime in minutes),
Fig. 2: The object selection for the functional requirements posing

the latter provides the possibility to crawl predefined values from the ontology and select the suitable one. For instance, answers related to the first 3 functional requirement questions (i.e., Action, Object and APQC category) are of search-insert type. Namely, users can insert keywords for the BP they are looking for, and the ontology returns the concepts matching these keywords. Fig. 2 shows the implementation result of this functionality.

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

Specifications from the KB as follows:

- A cloud service with the execution constraint of 20 times per day.

Requirements from the questionnaire as follows:

- Should the process be executed on a daily, weekly, monthly or yearly basis?  
  Answer: At least on a weekly basis.
  - How many times should the process be executed?
    Answer: At least 10 times

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

4.2 Question Prioritisation Algorithm

The NFR questions follow a question prioritisation algorithm. This enables to identify the matching cloud services by asking as few questions as possible. An-
answers to the questions, along with previous ones, are used to display the follow-up question. The algorithm considers the following:

- Grouping among non-functional attributes. For instance, assuming that a cloud service has the availability and response time attributes of the Performance category. If the user selects to answer one of the two, the follow-up question will be on the remaining attribute in the same category.
- Entropy expressing the variation degree in the values of each non-functional attribute (e.g., availability). Entropy of an attribute is “0” when every cloud service stored in the KB contains the same attribute value, while “1” in the opposite case. For example, if all cloud services in the KB have their data location in Switzerland, the entropy of this attribute is 0. As such, the question related to the preferred data location will not be asked as it will not filter out any services from the matching set.

The entropy formula is expressed as follows:

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

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: $p_{ij} = \frac{[\text{CS}]_{\text{csval}_{ik} = \text{val}_{ij}}}{[\text{CS}]}$ 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.

The prioritisation algorithm’s signature and main logic is as follows.

**Input.**

- Already stated variables: attr, CS, val, csval.
- The set of non-functional categories $C = \{\text{Data Security, Payment, Performance, Service support, Target Market}\}$.
- Set of tuples $<\text{attr}_i, Q_l>$ where $Q$ is the set of questions and $Q_l$ is a certain question where $1 \leq l \leq |Q|$. So, each tuple maps 1 attribute to 1 question.

**Output.** The filtered set of cloud services $\text{CS}$ that match with the content of the questionnaire, i.e., questions and answers.

**Business Logic.**

1. IF the number of categories left is positive ($|C| > 0$), select a category $c_n$, ELSE exit.
2. IF $c_n$ has a positive number of semantic attributes left, i.e., $|\text{attr}_i, s.t \text{attr}_i.cat = c_n| > 0$, THEN calculate the entropy of all the selected category’s attributes, ELSE remove the current category $c_n$ from $C$ and go to (1).
3. Select attribute $\text{attr}_i$ with highest entropy.
4. Display question $Q_l$ that is mapped with the $\text{attr}_i$. 

5. Get user answer mapping to a value val$_{ij}$ of attribute attr$_{i}$
6. Filter services in CS which do not satisfy the condition: csval$_{ik}$ = val$_{ij}$.
7. Remove the semantic attribute attr$_{i}$ from the category $c_n$ and go to (2).
8. Exit.

5 Syntactic Matchmaking

Business and technical matchmakers cannot guarantee the message compatibility between selected services in a BPaaS workflow. Such a compatibility is thus an obligatory, hard constraint in service selection for producing optimal, message-compatible solutions that can be safely executed. So, a component is needed able to derive such compatibility and supply it as a function to the Service Selector.

The main idea is that this component should first find which output messages of previously selected services match to which input messages of the currently selected service (based on the Service Selector’s solution generation process) for each execution path in the BPaaS workflow. Then, it should check for each message-to-message match if the first message conveys less information than that required by the second message. If this checking succeeds, there is no compatibility between the execution path’s considered services. When all message pair matches are compatible, the considered services are message-compatible.

Message Matching. The first message compatibility step can rely on existing semantic service annotations to easily and rapidly discover matching message pairs, as the messages involved in these pairs should correspond to semantically compatible concepts. However, even in the presence of such knowledge, message matching is by far trivial and follows a two-step process: (a) semantic message matching; (b) syntactic message matching. This process is exemplified via the example of a certain service pair which involves service $S_1$ with 2 output parameters mapped to ontology concepts $A$ & $B$ and service $S_2$ with 2 input parameters mapped to ontology concepts $C$ & $D$.

At the semantic level, we follow a bipartite matching approach which checks whether every parameter of the current service has a respective mapping to one parameter of the previously selected services (or the initial user input) in a certain execution path and attempts to discover a solution with the lowest overall distance$^6$. As such, we first define a local matching degree between two parameters to be the distance between the parameters’ annotation concepts in the ontology subsumption hierarchy, provided that the second parameter’s concept subsumes the first parameter’s one. If the latter does not hold, the distance is infinite. This guarantees that no information loss occurs as in the opposite case, the more concrete concept in the $S_2$ input will require the specification of additional pieces of information than those exhibited in the concept in the $S_1$ output. A mapping solution’s overall distance is then the sum of the distances.

$^6$ here distance is opposite to matching as the more distant two parameters are, the less is their matching degree
Towards Business-to-IT Alignment in the Cloud

of the matches found. As such, the matching problem can be defined as follows:

$$\min \left\{ \frac{1}{|J|} \cdot \left( \sum_{i \in I} \left( \frac{\text{dist}(M_i, N_j)}{\text{max}_P \text{Size}} \cdot x_{ij} \right) \right) + \sum_{j \in J} \left( 1 - \sum_{i \in I} x_{ij} \right) \right\}$$

where $I$ and $J$ are the sets of input and output parameters, respectively, $x_{ij}$ is a decision variable whether the output parameter $i$ matches the input one $j$, $\text{dist}(M_i, N_j)$ is the distance between annotation concepts $M_i$ and $N_j$ of the two parameters pair while $\text{max}_P \text{Size}$ represents the maximum subsumption path length in the respective domain ontology used.

Suppose that the following relations hold in the running example: A subsumes B, C & B, C subsume D. In this respect, the best possible matching is \{A $\rightarrow$ C, B $\rightarrow$ D\} with overall distance of 2. The other matching solution \{A $\rightarrow$ D, B $\rightarrow$ C\} is not selected as the local distance between B & C is infinite so the overall distance is also infinite.

The algorithm then proceeds at the syntactic level by considering only those message pairs with a finite local degree of match. For each message pair filtered, we note the information items for the output parameter and the information items of the input parameter and then we see whether the former include the latter. As the matching of information items to the matching ontology concept has been already identified, we perform this checking by replacing the information items with the attributes of the ontology concept. Even if the concepts matched are not identical, as they are related with a subsumption relation, they will certainly have common attributes. So, the problem then is mapped to checking whether the concept attributes of the output parameter form a superset of the attributes of the input parameter.

Message types might also convey information not included in an ontology requiring to perform a different matching kind for them. The logic of this matching is similar to that for the semantic level. In particular, bipartite matching is performed with a sole difference in the way the distance is calculated at the local level. At that level, we need to consider both how similar the field names are and how close are their types. Name similarity can rely on well-known string distance measures (e.g., Levenshtein) while type similarity can rely on the approach in [24] mapping to the compatibility level between types. The overall distance at the local level would equal the weighted sum of the two different distances.

When there is no match for an input parameter part, this means that the compared services are semantically incompatible. Otherwise, if all input parameter parts are matched, then the compared messages are semantically compatible.

Let us now continue the running example to explain syntactic matchmaking. Suppose that $A$ & $C$ have been found equivalent. $A$ maps to message type $MT_1$ containing 4 information pieces $MT_{11}$, $MT_{12}$, $MT_{13}$ and $MT_{14}$. $C$ maps to message type $MT_2$ which contains 3 information pieces $MT_{21}$, $MT_{22}$, and $MT_{23}$. Based on matching message types to ontology concepts, we have that
MT_{11} and MT_{21} map to A.A1 while MT_{12} and MT_{22} map to A.A2. As such, the information pieces are transformed into \{A.A1, A.A2, MT_{13}, MT_{14}\} for first message type and \{A.A1, A.A2, MT_{23}\} for the second. For those pieces that do not map to ontology attributes, we need to solve a bipartite matching problem again. Suppose that \text{dist}(MT_{13}, MT_{23}) = 0.8 and \text{dist}(MT_{14}, MT_{23}) = 0.2. Then, obviously the sole mapping to be selected will be \{MT_{13} \mapsto MT_{23}\}. If we replace MT_{23} with MT_{13}, we then need to check whether \{A.A1, A.A2, MT_{13}\} is subset of \{A.A1, A.A2, MT_{13}, MT_{14}\} which holds.

6 Validation

Our approach was validated based on a use case developed within the Cloud-Socket consortium by the industrial partners. We focused on one of the most common BPs among SMEs - the Send Invoice one. This BP is modelled in BPMN, see Fig. 3, via our integrated BPaaS Design environment. It starts with the “Manage Customer Relationship” activity; next an exclusive gateway splits the BP flow between either creating a new invoice or updating an existing one. Then, the invoice completeness is checked, and finally the invoice is sent. In the following, starting with this BPMN process, we acquaint the reader with a prerequisite as well as the main steps involved in our approach.

6.1 Prerequisite Step: Service Profile Registration

The following services were inserted in the KB as instances of CloudService class:

- YMENS, Zoho and Sugar CRM were inserted as CRM systems which were annotated with the action Manage, the object Customer and the APQC category 3.5.2.4 Manage Customer Relationship
- Mathema Document Generator, Open Source Billing, Simple Invoice and InvoiceNinja as invoicing systems annotated with the action Generate, the object Invoice and the APQC category 9.2.2.2 Generate Customer Billing Data
- Gmail, Ninja_email and Mailjet were inserted as e-mail systems which were annotated with the action Manage, the object Invoice and the APQC category 9.2.2.3 Transmitting Billing Data to Customers

Table 1 shows a part of the non-functional profiles of the considered services.

| Service          | Monthly Downtime | Response Time Level | File Type                      | No of Simul. Users | Execution Constraint |
|------------------|------------------|--------------------|-------------------------------|--------------------|----------------------|
| YMENS CRM        | 4 min            | High               | Office doc, PDF, audio, video | 500                | none                 |
| Zoho CRM         | 4 min            | High               | Office doc, PDF, audio, video | 500                | none                 |
| Sugar CRM        | 10 min           | Medium             | Office doc, PDF, audio, video | 200 (monthly basis)| none                 |
| InvoiceNinja     | 4 min            | High               | Office doc, PDF, audio, video | 600                | none                 |
| Ninja Email      | 4 min            | High               | Office doc, PDF, audio, video | 400                | none                 |
| Simple Invoices  | 10 min           | Medium             | Office doc, PDF, audio, video | 300                | none                 |
| Mailjet          | 4 min            | High               | Office doc, PDF, audio, video | 100 (monthly basis)| 1K                   |
| Open Source Billing | 4 min         | Medium             | Office doc, PDF, audio, video | 200                | none                 |
| Gmail            | 4 min            | High               | Office doc, PDF, audio, video | 100                | None                 |

Table 1: Functional requirements for each group and single activity

6.2 First Main Step: Business Matchmaking

The Business Matchmaker was used to identify the most suitable cloud services. In particular, as a first step, we applied the questionnaire on the whole BP. Fig. 4 shows the notebook from which the questionnaire was started.

We specified functional requirements in the first 3 questions - object Send, action Invoice and APQC category 9.2.2 Invoice Customer - and none of the cloud services matched; Fig. 5 shows the empty list at the right-hand side.

Next, the questionnaire was applied on two single activities (i.e., Manage Customer Relationship and Send Invoice) as well as on a group of activities (i.e., Create Invoice, Update Invoice and Check Invoice Completeness).

Table 2 shows the functional requirements for each activity/group. In the first case, after specifying action, object and APQC category, the questionnaire showed the three matching cloud services: YMENS, Zoho and SugarCRM. In the 4th question, we chose the Performance category, and the question prioritisation algorithm kicked in. The question regarding the number of simultaneous users
Fig. 4: The starting notebook for the whole process

Fig. 5: Functional requirements failure and NFR category selection

was asked (attribute with highest entropy) and a value of 500 was entered. This filtered out SugarCRM as it has the capability of max 200 simultaneous users.

| BPMN Activity                                      | Action     | Object              | APQC Category                                  |
|----------------------------------------------------|------------|---------------------|------------------------------------------------|
| Manage Customer Relationship (Single activity)     | Manage     | Customer Relationship| 3.5.2.4 Manage Customer Relationship            |
| Send Invoice (Single activity)                     | Send       | Invoice             | 9.2.2.3 Transmit Billing Data to Customers     |
| Create New Invoice, Update Invoice, Check Invoice Completeness (Group of activities) | Generate   | Invoice             | 9.2.2.2 Generate Customer Billing Data         |

Table 2: Functional requirements for each group and single activity

Similarly, we applied the questionnaire on the designated group of activities. The matching services were InvoiceNinja and Open Source Billing, see Fig. 6a.
Towards Business-to-IT Alignment in the Cloud 17

(a) The selected invoice management services

(b) The selected email services

Fig. 6: The selected services for last two activity groups

Finally, we applied the questionnaire on the last activity of the BP: Send Invoice. The matching cloud services were Ninja E-mail and Mailjet (see Fig. 6b).

6.3 Second Main Step: Technical Matchmaking & Selection

As the final result maps to two services per each activity (group), we now proceed with the technical matchmaking and selection. Suppose that the user provides the following global requirements for the whole process: \( \text{cost} < 100 \text{ euros per month} \), \( \text{cyletime} < 1 \text{ minute} \) and \( \text{VPM} < 16 \) (#vulnerabilities per month). Further, suppose that the user imposes that for the Manage Customer Relationship activity, the following constraints should hold: \( \text{resonsetime} < 30 \text{ seconds} \) and \( \text{VPM} < 10 \). Finally, Table 3 depicts the non-functional profiles of the remaining services.

| Service                    | Cost  | Response Time | VPM |
|----------------------------|-------|---------------|-----|
| ZOHO CRM                   | 30 euros | 35 seconds | 10  |
| YMENS CRM                  | 35 euros | 20 seconds | 05  |
| Mailjet                    | 25 euros | 10 seconds | 02  |
| Ninja_Email                | 10 euros | 15 seconds | 03  |
| Open Source Billing        | 35 euros | 25 seconds | 08  |
| Invoice Ninja              | 45 euros | 10 seconds | 05  |

Table 3: The technical non-functional offerings of the 6 services

Technical non-functional matchmaking would then filter Zoho CRM as it does not conform to the local constraints posed for the CRM-based activity. This leads to performing the selection over 4 solutions as we have one candidate for the first (group) of activities and 2 candidates for the rest two activity groups. However, while running service selection, it is detected that the Ninja_Email and Open Source Billing are incompatible, which leaves us with 3 solutions. Moreover, the solution mapping to selecting YMENS, Open Source Billing and Ninja_Email has VPM equal to 16 which violates the respective global constraint. So, in the end, we need to select only between two solutions.
The combinations to be checked for selecting the best one are depicted in Table 4. As the broker requires to optimise all non-functional terms (cost, cycle time and VPM), it gives equal preference over them. By also considering that the activities are sequentially executed in the BPaaS workflow, the final result would map to selecting the services YMENS, InvoiceNinja, and Ninja_Email. While there is perfect syntactic compatibility between InvoiceNinja and Ninja_Email as they are offered by the same organisation, in the case of YMENS CRM and InvoiceNinja the respective message types are compatible but still need to be aligned (e.g., attributes accountid and id number which map to the same attribute id of concept Client). As such, the MS service was included between these 2 services resulting in a workflow with 4 services sequentially executed (YMENS CRM → MS → InvoiceNinja → Ninja_Email).

| Solution                      | Cost | Cycle Time | VPM | Utility |
|-------------------------------|------|------------|-----|---------|
| YMENS + InvoiceNinja + Ninja_Email | 90 euros | 45 seconds | 13  | 0.144   |
| YMENS + Open Source Billing + Mailjet | 95 euros | 50 seconds | 15  | 0.099   |

Table 4: The final ordered solutions produced

Please note that while the produced workflow is automatically generated, there is still some manual work that needs to be done by the modeller which is more trivial than the selection of services from a huge solution space. Such work includes the insertion of user tasks to enable users to provider their input as well as of manual tasks in case some service input or output requires special, human-oriented processing. We plan to extend our work to incorporate such tasks automatically in the produced workflow in the near future.

7 Conclusions and Future Work

This paper has introduced a novel architecture for the design of business process as a service (BPaaS) products which is able to effectively deal with the business-to-IT alignment problem in order to map an initial domain-specific BP into an executable BPaaS workflow. Such an architecture has been carefully designed and implemented to include suitable components which focus on different parts of the business-to-IT alignment problem, including business and technical matchmakers, a service selection as well as an automatic workflow update component to enable the effective addressing of the message compatibility problem in service-based workflow execution.

Our future work will focus on more advanced research challenges which include: (a) the automatic production of a more complete and more close to production workflow via the incorporation of different kinds of non-service tasks (see previous section); (b) the automatic population of the KB; (c) the coverage of additional cases in business-to-technical-requirement alignment.
Acknowledgments  This research has received funding from the European Community’s Framework Programme for Research and Innovation HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

1. Antunes, G., Bakhshandeh, M., Borbinha, J., Cardoso, J., Dadashnia, S., Francescomarino, C.D., Dragoni, M., Fettke, P., Gal, A., Ghidini, C., Hake, P., Khiat, A., Klinkmüller, C., Kuss, E., Leopold, H., Loos, P., Meilicke, C., Niesen, T., Pesquita, C., Pénus, T., Schoknecht, A., Sheetrit, E., Sonntag, A., Stuckenschmidt, H., Thaler, T., Weber, I., Weidlich, M.: The Process Model Matching Contest 2015. Lecture Notes in Informatics (2015), https://www.matthiasweidlich.com/paper/process_model_matching_contest_2015.pdf
2. Azzini, A., Braghin, C., Damiani, E., Zavatarella, F.: Using Semantic Lifting for improving Process Mining: a Data Loss Prevention System case study (2013)
3. Benaboud, R., Maamri, R., Sahnoun, Z.: Agents and owl-s based semantic web service discovery with user preference support. International Journal of Web & Semantic Technology 4(2), 57–75 (2013)
4. Cliffe, O., Andreou, D.: Service Matchmaking Framework. Public Deliverable D5.2a, Alive EU Project Consortium (10 September 2009), available at: http://www.ist-alive.eu/index.php?option=com_docman&task=doc_download&gid=28&Itemid=49
5. Cloud Select Industry Group (C-SIG): Cloud Service Level Agreement Standardization Guidelines. Technical Report, EC (2014), http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?action=display&doc_id=6138
6. Duipmans, E.: Business Process Management in the cloud: Business Process as a Service (BPaaS), Tech. rep. (2012)
7. Emmenegger, S., Hinkelmann, K., Laurenzi, E., Martin, A., Thönsen, B., Witschel, H.F., Zhang, C.: An Ontology-Based and Case-Based Reasoning Supported Workplace Learning Approach. pp. 333–354. Springer, Cham (feb 2017), http://link.springer.com/10.1007/978-3-319-66302-9\_17
8. Emmenegger, S., Hinkelmann, K., Laurenzi, E., Thönsen, B.: Towards a Procedure for Assessing Supply Chain Risks Using Semantic Technologies. In: Fred, A., Dietz, J., Liu, K., Filipe, J. (eds.) Knowledge Discovery, Knowledge Engineering and Knowledge Management, Communications in Computer and Information Science, vol. 415, pp. 393–409. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-54105-6\_26
9. Faria, D., Pesquita, C., Santos, E., Cruz, I.F., Couto, F.M.: AgreementMakerLight Results for OAEI 2013 http://disi.unitn.it/~p2p/OM-2013/oaei13_paper1.pdf
10. Frank, U.: Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In: Domain Engineering, pp. 133–157. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://link.springer.com/10.1007/978-3-642-36654-3\_6
11. Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., van der Merwe, A., Woitsch, R.: A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. Computers in Industry 79, 77–86 (2016)
12. Hinkelmann, K., Kritikos, K., Kurjakovic, S., Lammel, B., Woitsch, R.: A Modelling Environment for Business Process as a Service. pp. 181–192. Springer, Cham (Jun 2016), http://link.springer.com/10.1007/978-3-319-39564-7_18
13. Hinkelmann, K., Laurenzi, E., Lammel, B., Kurjakovic, S., Woitsch, R.: A Semantically-Enhanced Modelling Environment for Business Process as a Service. In: 2016 4th International Conference on Enterprise Systems (ES), pp. 143–152. IEEE (Nov 2016), http://ieeexplore.ieee.org/document/7880484/
14. Hrgovcic, V., Karagiannis, D., Woitsch, R.: Conceptual Modeling of the Organisational Aspects for Distributed Applications: The Semantic Lifting Approach. In: Compsac Workshops. pp. 145–150. IEEE (Jul 2013)
15. Hwang, C., Yoon, K.: Multiple Criteria Decision Making. Lecture Notes in Economics and Mathematical Systems (1981)
16. Jiang, S., Aagesen, F.A.: An Approach to Integrated Semantic Service Discovery. In: First International IFIP TC6 Conference. pp. 159–171. Springer (2006)
17. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M.: Fundamental Conceptual Modeling Languages in OMiLAB. In: Domain-Specific Conceptual Modeling, pp. 3–30. Springer International Publishing, Cham (2016), http://link.springer.com/10.1007/978-3-319-39417-6_1
18. Klusch, M.: Semantic Web Service Coordination. In: CASCOM: Intelligent Service Coordination in the Semantic Web. pp. 59–104. Springer (2008)
19. Kritikos, K., Magoutis, K., Plexousakis, D.: Towards knowledge-based assisted iaas selection. In: CloudCom. pp. 431–439. IEEE Computer Society, Luxembourg (2016)
20. Kritikos, K., Plexousakis, D.: Requirements for QoS-Based Web Service Description and Discovery. IEEE Transactions on Services Computing 2(4), 320–337 (2009)
21. Kritikos, K., Plexousakis, D.: Novel Optimal and Scalable Nonfunctional Service Matchmaking Techniques. IEEE Transactions on Services Computing 7(4), 614–627 (2014)
22. Kritikos, K., Plexousakis, D.: Multi-cloud Application Design through Cloud Service Composition. In: CLOUD. pp. 686–693. IEEE, New, NY, USA (2015)
23. Kritikos, K., Plexousakis, D.: Towards Combined Functional and Non-functional Semantic Service Discovery. In: ESOCC. pp. 102–117. Springer, Vienna, Austria (2016)
24. Kritikos, K., Plexousakis, D., Paternò, F.: Task model-driven realization of interactive application functionality through services. TiiS 3(4), 25 (2014)
25. Laurenzi, E., Hinkelmann, K., Reimer, U., van der Merwe, A., Sibold, P., Endl, R.: DSM4PTM - A domain-specific modelling language for patient transferal management. In: ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 3, Porto, Portugal, April 26-29, 2017. pp. 520–531 (2017), https://doi.org/10.5220/0006388505200531
26. Plebani, P., Pernici, B.: URBE: Web Service Retrieval Based on Similarity Evaluation. IEEE Transactions on Knowledge and Data Engineering 21(11), 1629–1642 (2009)
27. Saati, T.: The Analytic Hierarchy Process. McGraw-Hill (1980)
28. Silver, B.: BPMN Method and Style, Second Edition. Cody-Cassidy Press, Aptos, second edn. (2011)
29. Uschold, M., King, M., Morale, S., Zorgios, Y.: The Enterprise Ontology. The Knowledge Engineering Review 13(01), 31–89 (1998)
30. Vaishnavi, V., Kuechler, B.: Design Science Research in Information Systems (2004), http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf
31. Watfa, M.K., Najjar, N.A.L., Cheikha, J., Buali, N.: A New Framework for Cloud Business Process Management, pp. 83–92. Springer International Publishing, Cham (2016), https://doi.org/10.1007/978-3-319-38904-2_9
32. Whibley, P.: BPM in the Cloud - Transforming the Business Case for Process Improvement. Tech. rep. (2012)
33. Woitsch, R., Utz, W.: Business Processes as a Service (BPaaS): A Model-Based Approach to align Business with Cloud offerings (jan 2015), https://zenodo.org/record/35583#.WbDscNhLfmE