Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Restriction-based Fragmentation of Business Processes over the Cloud

Slim Kallel[1] | Zakaria Maamar[2] | Mohamed Sellami[3] | Noura Faci[4] | Ahmed Ben Arab[1] | Walid Gaaloul[3] | Thar Baker[5]

[1]ReDCAD, University of Sfax, Sfax, Tunisia
[2]College of Technological Innovation, Zayed University, Dubai, UAE
[3]SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, Evry, France
[4]LIRIS, Claude Bernard Lyon 1 University, Lyon, France
[5]Department of Computer Science, Liverpool John Moores University, Liverpool, UK

Correspondence
*Slim Kallel, ReDCAD, ENIS, University of Sfax. Email: slim.kallel@fsegs.usf.tn

Present Address
Route de la Soukra, B.P.W, Sfax 3038, Tunisia.

Summary

Despite the elasticity and pay-per-use benefits of cloud computing (*aka* fifth utility computing), organizations adopting clouds could be locked-into single cloud providers, which is not always a "pleasant" experience when these providers stop operations. This is a serious concern for those organizations that who would like to deploy (core) business processes on the cloud along with tapping into these 2 benefits. To address the lock-into concern, this paper proposes an approach for decomposing business processes into fragments that would run over multiple clouds and hence, multiple providers. To develop fragments, the approach considers both restrictions over owners of business processes and potential competition among cloud providers. On the one hand, restrictions apply to each task in a business process and are specialized into budget to allocate, deadline to meet, and exclusivity to request. On the other hand, competition leads cloud providers to offer flexible pricing policies that would cater to the needs and requirements of each process owner. A policy handles certain clouds' properties referred to as limitedness, non-renewability, and non-shareability that impact the availability of cloud resources and hence, the whole fragmentation. For instance, a non- shareable resource could delay other processes, should the current process do not release this resource on time. During fragmentation interactions between owners of processes and providers of clouds happen according to 2 strategies referred to as global and partial. The former collects offers about cloud resources from all providers, while the latter collects such details from particular providers. To evaluate these strategies' pros and cons, a system implementing them as well as demonstrating the technical feasibility of the fragmentation approach using credit-application case study, is also presented in the paper. The system extends BPMN2-modeler Eclipse plugin and supports interactions of processes' owners with clouds' providers that result to identifying the necessary fragments with focus on cost optimization.

KEYWORDS:
Business process, Cloud, Fragmentation, Pricing, and Restriction.

1 | INTRODUCTION

Blending Business Processes (BP) with cloud computing is attracting the attention of the ICT community who sees a lot of benefits in adopting Everything-as-a-Service (*aaS) operation model when they need to secure resources (e.g., infrastructure, platform, and software) for their BP applications like payroll, supply chain, and customer relationship [1,2,3]). *Elasticity* and *pay-per-use* are some cloud benefits that permit, respectively, to scale (up/down) resources according to applications' changing demands and to bill applications' owners, only, when there is an effective use of resources.

In conjunction with this blend, the ICT community is also examining ways of fragmenting (i.e., decomposing) the execution of BPs over multiple, independent clouds. In addition to securing resources from many clouds, fragmentation relieves BP owners from being locked-into one particular cloud provider. Indeed, this concern has been undermining cloud computing expansion for some time[1]. Multiple and various clouds could be used, which permits to respond to the particular needs and requirements of each BP fragment [5]. For instance, a sensitive BP fragment prioritizes security over availability, while a critical BP fragment (in the context of evacuation services for emergency management [6]) prioritizes reliability over performance. In this paper, we examine how *restrictions* over owners of BPs impact the fragmentation of BPs. By restrictions, we mean budget that an owner wishes to allocate, deadline that an owner wishes to meet, and exclusivity (i.e., isolation) that an owner wishes to have. Along with these restrictions, we define *properties* of cloud resources that would impact the offers of cloud providers to the demands of BP owners. The properties that we propose include limitedness (*versus* unlimitedness), non-renewability (*versus* renewability), and non-shareability (*versus* shareability). To the best of our knowledge, this is the first time that cloud resources are associated with such properties that directly impact their availabilities. In the literature, abundance is cloud resource's main benefit, but this should not always be the case. According to Dimick, all resources are expected to decline [7].

Commonly referred to as *know-how*, *"... A business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations."* [8]. Simply put, a BP is a set of interconnected tasks (or activities) with respect to a process model that defines What to do,Why, When, Where, and by Whom (5Ws). At run-time, tasks are assigned to persons and/or machines so they execute them using (cloud) resources. From a cloud perspective, different criteria can impact the development of fragments of tasks in conjunction with cloud resources' availabilities. For instance, one criterion promotes loose coupling among fragments for the sake of adaptability while other criteria (*i*) reduce data transfer among fragments for the sake of efficiency, (*ii*) protect data fragments for the sake of security, and (*iii*) guarantee resource availability to fragments for the sake of performance. While all these criteria are strictly concerned with the **computation** (performance) of BPs, **managerial** concerns of BP owners are overlooked. **Such concerns include budget to allocate, deadline to meet, and exclusivity to afford.** In this paper, we address managerial concerns.

Our contribution is manifold: (*i*) definition of restrictions over BP owners in terms of budget, deadline, and exclusivity; (*ii*) definition of properties of cloud resources in terms of limitedness, non-renewability, and non-shareability; (*iii*) design of a BP fragmentation approach that considers both these restrictions and these properties when assigning cloud resources to BP tasks; (*iv*) development of a flexible pricing policy for cloud resources' offers; and last but not least, (*v*) demonstration of the BP fragmentation approach through a case study and system.

The remainder of this paper is organized as follows. Section 2 is an overview of BP fragmentation, in general, and BP execution in the context of cloud, in particular. A credit application case study is also included in this section. Section 3 discusses our BP fragmentation approach in terms of defining cloud resources' properties and presenting the steps for completing this fragmentation. Section 4 implements and validates the fragmentation approach using the case study. Finally, Section 5 concludes the paper and identifies some future work.

2 | RELATED WORK AND CASE STUDY

This section discusses first, BP fragmentation and then, BP execution over the cloud. The section concludes with a case study illustrating some managerial (and not computational) concerns of BPs' owners that are overlooked in existing BP fragmentation approaches.

2.1 | Business-process fragmentation

Fragments represent reusable process knowledge that can be used to construct more complex processes [9,10]. Mancioppi et al. [11] present a comprehensive discussion about the importance of BP fragmentation. They identify 3 motives: enable a distributed execution of process models, abstract process models, and enable the reuse of certain parts of process models. Mancioppi et al.'s classification of fragmentation techniques revolves

[1]*"According to a Logicworks survey by Wakefield Research, 78% of IT decision makers believe that concerns about vendor lock-in prevent their organisation from maximising the benefits of cloud resources. This means that the majority of IT leaders consciously choose not to fully invest in cloud because they value long-term vendor flexibility over long-term cloud success"* [4].

around 8 questions: what input is submitted to the fragmentation, why is the process model fragmented, when is the fragmentation performed in the process model's lifecycle, who does perform the fragmentation, what output results from the fragmentation, which fragmentation criteria are considered, where is the fragmentation performed in the process model, and finally, how is the fragmentation performed? In addition to Mancioppi et al.'s 3 motives, Mezni and Kbekbi mention fragmentation benefits in the context of business processes that are based on service composition[10]. These benefits are minimizing composition time and improving reliability of composition.

Maalouf et al.[12] look into fragmentation for reuse purposes during BP modeling. The authors developed a tool upon some semantic Web technologies like Resource Description Framework (RDF) and Simple Protocol and RDF Query Language (SPARQL). The first technology enriches BP elements with specific details like structure, context, and historical usage, prior to representing and storing these details in an RDF database. The second technology queries the database of existing processes based on users' requirements. Maalouf et al.'s tool relies on user-defined weights for similarity assessment so, that, the most relevant fragments are extracted on-the-fly during the development of new BP models. Relations between fragments could be of types successor/follower or parent/child. Hou et al.[13] explore BP fragmentation for distribution purposes with emphasis on Internet-of-Things (IoT)-aware BPs. These BPs have to cope with the high volume and velocity of data that IoT generates and hence, need to be transmitted and processed timely. The authors propose a location-based fragmentation algorithm to partition a BP before applying Kuhn-Munkres algorithm so, that, an optimal deployment of BP fragments is achieved. Fragment collaboration takes place through a topic-based publish/subscribe infrastructure, which allows to reduce network traffic and save process execution-time.

Anis-Zemni et al.[14] propose an approach that preserves/promotes privacy/reuse during BP fragmentation. They note that designing new and/or updating existing processes is still time consuming, costly, and prone to errors. This gets even worse when inter-organizational processes are the subject of design. Reusing existing fragments is an option as long as constraints like data privacy are satisfied. Anis-Zemni et al.'s approach consists of 3 stages: represent a BP as a concept graph using formal concept analysis augmented with privacy-preserving constraints; split the BP into low-grained task clusters that preserve privacy; and group the resulted clusters into coarse-grained and reusable fragments in a way that sensitive association disclosure is avoided. Clusters encompass tasks that are semantically close in term of data handled by these tasks. The authors define cluster grouping rules like "if two clusters share the same task, then they will be grouped together".

Hens et al.[15] discuss the use, advantages, and disadvantages of an event architecture that would support a distributed execution of BPs. The authors first, show how a BP flow can be automatically transformed into (logically-) different BP model fragments. Fragments are complemented with event rules, stating their respective starting logic and hereby, creating autonomous, self-serving BP model components. The fragments are deployed on dedicated BP engines and also encapsulated into publish/subscribe wrappers in order to communicate with the event architecture. Publish/subscribe to event messages enables fragment collaboration and thus, to reconstruct the global BP execution.

Fdhila et al.[16] present a comprehensive set of change patterns that would support change propagation in a fragmented BP model. For instance, "delete" pattern removes an existing fragment from a BP model. This becomes challenging if the fragment refers to interactions between BP stakeholders that are different from those specified at design time. Another pattern, "replace", modifies the structure and elements of a given fragment in a BP model. This pattern is particularly useful when an entire or part of a BP redesign becomes necessary. For all change patterns, the authors provide change propagation algorithms that preserve consistency and compatibility of the BP choreography.

Xue et al.[17] use graph techniques to partition a Web service-based process, whether this process is structured or unstructured. To achieve this partitioning, the graph has to be directed helping to group nodes in the structure graph that underpins the process's business model. A set of transformation rules are developed for grouping vertices in a graph taking into account data and control dependencies between activities in a process. Although Xue et al. do not clearly state the objective of process partitioning, i.e., what is the optimization function?, the results of their experiments show that a partition-based decentralized service composition can have lower average response time and high throughput compared to a centralized service composition.

Last but not least, Pourmasoumi et al.[18] use event logs, and not process models like the rest do, to identify[2] morphological fragments that would meet requirements, composability and flexibility. The advantages of logs over models are that they reduce the detection error rate (number of misidentified fragments) and they speed the identification process. Pourmasoumi et al. build upon Bunge's ontology, [19], to define the characteristics of a process in terms of things, properties, state space, and lawful event space. These characteristics help identify the fragments along with the similarities that could exist between them.

The aforementioned paragraphs discuss BP fragmentation from multiple perspectives for instance, why to fragment (rationale)[12], how to fragment (techniques)[20,14,16,13,17], how to identify fragments[18], what to gain from fragmentation (benefits), and what could happen during fragmentation[15]. However, little interest is given to the concerns of BP owners who would like to achieve fragmentation (*i*) within a certain budget and/or a certain time frame based on what cloud providers would offer and (*ii*) in isolation from the competition so, that, the safety of their operations is ensured when their BPs are deployed on different clouds.

[2]We see differences between identifying fragments based on past executions of a process and fragmenting a process.

2.2 | Cloud resource allocation to business processes

Resource allocation has been recognized as an important topic for BP deployment over the cloud. This allocation is dependent on cloud resources' types such as computation, storage, and communication and should meet requirements of transparency, uniformity, and scalability[21].

Graiet et al.[22] examine the correctness and consistency of allocating cloud resources to BPs. Correctness is about correspondences between specified requirements and users' real needs while consistency is about checking that correspondences are free of contradictions. The authors model resource allocation behavior in BP management using Event B formalism, satisfy all process design and execution requirements, and verify cloud resources' constraints and properties (i.e., elasticity and shareability).

Hachicha et al.[23] provide a semantic framework for cloud resource management in BPs. The framework addresses certain concerns of cloud users such as do-not-share a computation resource with other tenants (similarly to our exclusivity restriction) and do-use a particular elasticity policy. The framework is built upon properties of cloud resources (i.e., limited, non-limited, shareable, non-shareable, elastic, and non-elastic) and social relations between BPs (e.g., cooperation and partnership). The framework also provides a CloudPro ontology to formally describe cloud resources along with their social relations, and a resolution of conflicts over resources to ensure their correct allocation to BPs.

Ahmed-Nacer et al.[5] discuss the secure deployment of BPs over multiple clouds using fake fragments. The use of multiple clouds addresses the lock-into concern (as stated in Section 1) and prevents cloud providers from understanding and/or capturing a company's main asset that is their know how. Using fake fragments at specific locations complicates a BP's process model, which permits to hide the direct interaction between clouds that execute sensitive fragments. This way of doing should delay the discovery of the process model by malicious cloud providers. Ahmed-Nacer et al.'s approach features the following steps: retain sensitive data and logic at premises, split BP logic into several BP fragment logics, add non-functional logic, obfuscate data, separate logic and data, and finally, split cases between clouds. Addressing security concerns in the cloud could benefit from trust as well. Liu et al. examine the impact of objective and subjective trust on scheduling multimedia services over multiple clouds[24]. To this end, they propose a scheduling algorithm that uses, in addition to trust, cost and deadline when looking for reasonable cloud resources.

Last but not least, Schulte et al.[25] examine how to allocate cloud resources to elastic processes based on BP owners' requests. The authors define both resource demands of tasks and a cost model for resources. They also propose a cost-efficient BP scheduling mechanism to achieve Service Level Objectives (SLOs; e.g., availability and response time) specified in some contract. This mechanism consists of leasing and releasing cloud-based computational resources (e.g., virtual machines) over time in order to optimize cost. This requires predicting the cost and performing a cost/performance analysis.

The aforementioned approaches propose optimization solutions to cloud resource allocation given certain requirements and needs of BPs' owners. However, these approaches seem "neglecting" priorities and/or concerns of cloud providers (a major stakeholder in any resource-allocation exercise) who could for instance, adjust the availability and affordability of their cloud resources according to the demands (e.g., during off-peak hours) of resources that they receive. We handle availability and affordability through clouds' properties and flexible pricing policies, respectively.

2.3 | Case study

To illustrate the challenges that organizations could face when adopting cloud, we consider a small credit company that acts on behalf of a group of commercial banks when applicants of credits originate from persons and not corporates. These commercial banks would like to focus on some core businesses, so they outsource non-core businesses to the credit company. To remain efficient, the credit company needs to cut operational costs by deploying some BPs like credit application[3] on the cloud. The BPMN[27] representation[4] of the credit application's process model is given in Fig. 1 where different tasks (e.g., check-for-completeness and request credit card) and gateways (e.g., application completeness and credit amount) are illustrated. When the company's clerk receives an online credit application, she verifies its completeness in compliance with existing local legislations and commercial banks' regulations. If incomplete, the clerk asks the applicant for more information. Otherwise, the clerk performs additional checks depending on the amount of credit (e.g., up to \$500). Then, the clerk sends the senior manager the application for final verification. In the case of acceptance, the applicant is notified, a credit-card production request is issued, and the process ends. Otherwise, the applicant is notified of the rejection and the process ends, too. Not represented in Fig. 1, the credit company regularly reports its activities like number of approvals and insolvent customers to the commercial banks so, that, decisions like revising credit limits are made.

In Fig. 1 we illustrate some $\mathcal{R}$estrictions ($\mathcal{R}$) related to budget ($\mathcal{R}_1$), deadline ($\mathcal{R}_2$), and exclusivity ($\mathcal{R}_3$). To identify the best cloud offers that would satisfy these restrictions, the company's owner contacts different cloud providers who, depending on their ongoing and future commitments towards existing customers, provide different prices. These prices take into account whether a process's owner would renew cloud resources in the future, would mind sharing cloud resources with other companies, would be willing to spend more on cloud resources, would change halfway

[3]Credit-application BP is a well-established workflow in the literature[26].

[4]Although business-process modeling paradigms do not fall into the scope of this paper, readers are referred to[28] for a good discussion about such paradigms.

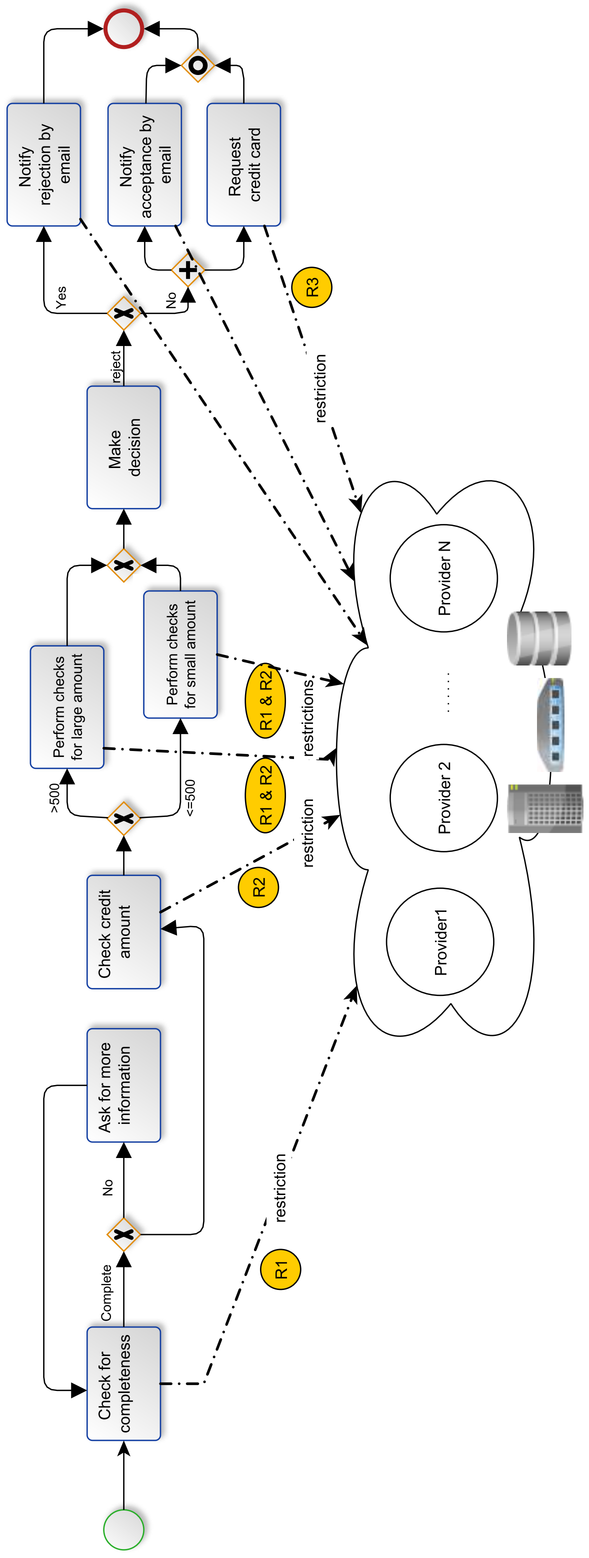


FIGURE 1 BPMN representation of credit application business-process

through his demands of cloud resources, would accept paying penalties because of these changes, etc. To the best of our knowledge existing fragmentation approaches do not address these questions that we deem critical. For instance, an owner is not willing to pay penalties, so a cloud provider should offer prices for both cases. In this paper, we discuss how our approach caters to the needs of both BP owners and cloud providers by allowing them to express restrictions over tasks and to make offers based on these restrictions, respectively.

3 | RESTRICTION-BASED FRAGMENTATION OF BUSINESS PROCESSES

This section presents our approach for BP fragmentation. We first, define properties of cloud resources and then, specify restrictions over owners of BPs. Finally, we detail the fragmentation approach's 5 steps that are specification, announcement, pricing, allocation, and formation.

3.1 | Definition of properties of cloud resources

Cloud resources targeted in this work are at the infrastructure level (*aka* IaaS) offering the necessary execution support to BPs in terms of computation (e.g., CPU) to allow task processing, persistence storage (e.g., database) to allow certain tasks to retrieve data they need and/or to store data they produce, and network (e.g., router) to allow data communication between tasks (and later between fragments of tasks).

To define properties of cloud resources, we build upon our previous work on social coordination of BPs [29] as follows:

- limited (l) property: when the use of a cloud resource is restricted by a time frame and/or a threshold. The threshold is put for instance, on the processing power of a computational resource, the capacity of a storage resource, and/or the bandwidth of a network resource. It is worth noting that...
  - Beyond a certain time frame, the cloud resource ceases to exist (e.g., withdrawn) and/or becomes unavailable for a task (however, other tasks can continue using the same cloud resource).
  - Requesting to increase the threshold[5] results into a negative response by the resource to a task (however, other tasks can continue using the same cloud resource).
- Renewable (r) property: contrarily to a limited cloud resource, it is possible to renew the use of a cloud resource for another time frame for the same task that was initially assigned to this cloud resource.
- Expandable (e) property (*aka elasticity* in the cloud terminology): contrarily to a limited cloud resource, it is possible to expand the use of a cloud resource by increasing the threshold for the same task that was initially assigned to this cloud resource.
- Renewable & Expandable (re) property: contrarily to a limited cloud resource, it is possible both to renew the use of a cloud resource for another time frame and to increase the threshold (if possible) for the same task that was initially assigned to this cloud resource.
- Non-shareable (ns) property: when the concurrent use of a cloud resource by many tasks needs to be scheduled/coordinated.

We use the 5 cloud properties (l, r, e, re, and ns) to develop the use cycle ($\mathcal{UC}$) of a cloud resource (res) (Fig. 2). Unless stated a cloud resource is by default unlimited (ul) and/or shareable (s). Below are 3 examples of use cycles; readers are referred to [29] for more details on a (not necessarily cloud) resource's use cycle per property.

1. Unlimited cloud resource's use cycle is as follows: $\mathcal{UC}(\text{res}_{\text{ul}})$: made available $\xrightarrow{\text{waiting to be assigned}}$ not used $\xrightarrow{\text{use approval}}$ used $\xrightarrow{\text{no}-\text{longer useful}}$ withdrawn.
2. Limited cloud resource's use cycle is as follows: $\mathcal{UC}(\text{res}_{\text{l}})$: made available $\xrightarrow{\text{waiting to be assigned}}$ not used $\xrightarrow{\text{use approval}}$ used $\xrightarrow{\text{use update}}$ done $\xrightarrow{\text{use completion}}$ withdrawn. The transition from done to withdrawn followed by end-of-state shields a cloud resource from any new or additional tentative of use by tasks after completing a use cycle.
3. Non-shareable cloud resource's use cycle is as follows: $\mathcal{UC}(\text{res}_{\text{ns}})$: made available $\xrightarrow{\text{waiting to be assigned}}$ not used $\xrightarrow{\text{lock}}$ locked $\xrightarrow{\text{use approval}}$ used $\xrightarrow{\text{release}}$ unlocked $\xrightarrow{\text{use update}}$ done $\xrightarrow{\text{use completion}}$ withdrawn. The transition from not used to locked is satisfied when the performance of a task requests exclusive access to the cloud resource.

[5] Reasons for requesting threshold increase could be motivated by the unexpected large-volume of data to process/store/communicate.

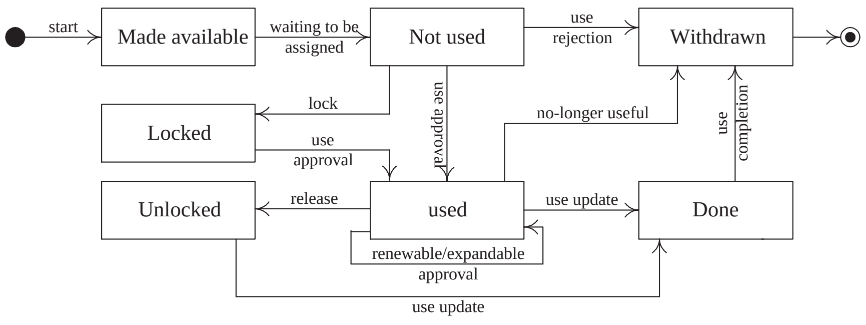


FIGURE 2 Cloud resource's use cycle (adapted from[29])

3.2 | Definition of restrictions over business processes' owners

In Section 1, we listed some computational concerns that BP owners could struggle with when developing task fragments to execute over the cloud. These concerns include adaptability (e.g., loose coupling *versus* strong coupling), efficiency, security, and performance. In this work, we particularly consider BP owners' managerial concerns and treat them as restrictions over tasks:

- Budget allows an owner to limit expenses on executing tasks over the cloud.
- Deadline allows an owner to meet dates when executing tasks over the cloud.
- Exclusivity allows an owner to express her willingness to share the cloud with other BP owners (e.g., cut costs).

Examples of the above restrictions could be 20$ as a budget for a compute resource and 2h as a deadline for completing all task execution.

3.3 | Steps of the fragmentation approach

Our fragmentation approach consists of 5 steps: specification, announcement, pricing, allocation, and formation.

3.3.1 | Specification

The specification step consists of defining the process model of a BP in terms of tasks and task dependencies (e.g., Fig. 1 in BPMN), needs of tasks in terms of cloud resources, and finally, restrictions on tasks although certain (could be all) tasks could be free of all (some of) restrictions. In Section 4, we illustrate how our system supports the specification step through a set of proper GUIs. In conjunction with the BP specification, the BP owner is made aware of the following points:

- Point_1: needs of cloud resources are defined in terms of computation (e.g., CPU speed), storage (e.g., database capacity), and network (e.g., bandwidth capacity).
- Point_2: restrictions on tasks are of types budget, deadline, and/or exclusivity. For the first two, we set a *tolerance ratio* that would allow a BP owner to process more offers from cloud providers that would, otherwise, be automatically discarded. For instance, a $10 budget along with a 20% tolerance ratio would lead to a revised budget of $12. This ratio is useful when dealing with critical tasks that must be executed. For the sake of simplicity, we assume that budget and deadline restrictions are always satisfied thanks to the tolerance ratio. It is worth noting that since exclusivity has a binary value that is either yes or no, having a ratio that would define an acceptable range for achieving this exclusivity, is not doable.
- Point_3: correctness criteria for successful completion of tasks are defined in terms of transactional properties[30] (Fig. 3):
  - For critical tasks, a BP owner assigns *retriable* as a transactional property to these tasks. Indeed, a task is *retriable* if it is sure to successfully complete after several finite activations[6]. This means the necessity of securing cloud resources[7] in case many activations (i.e., re-execution) of the task are deemed necessary until successful execution.

[6] A cap could be set to avoid infinite task activation to achieve successful execution.
[7] Ideally, cloud resources that are at least renewable should be considered so, that, the same cloud resources continue to be used.

- For "reversible" tasks, a BP owner assigns *compensatable* as a transactional property to these tasks so, that, their execution effects can be canceled[8]. This also means the necessity of securing appropriate cloud resources[9] in case undoing the task successfully takes place.
- Finally, *pivot* as a 3rd transactional property ensures that once a task successfully completes, its execution effects remain unchanged forever and cannot be undone. In the case of failure, the task cannot be retried[10].

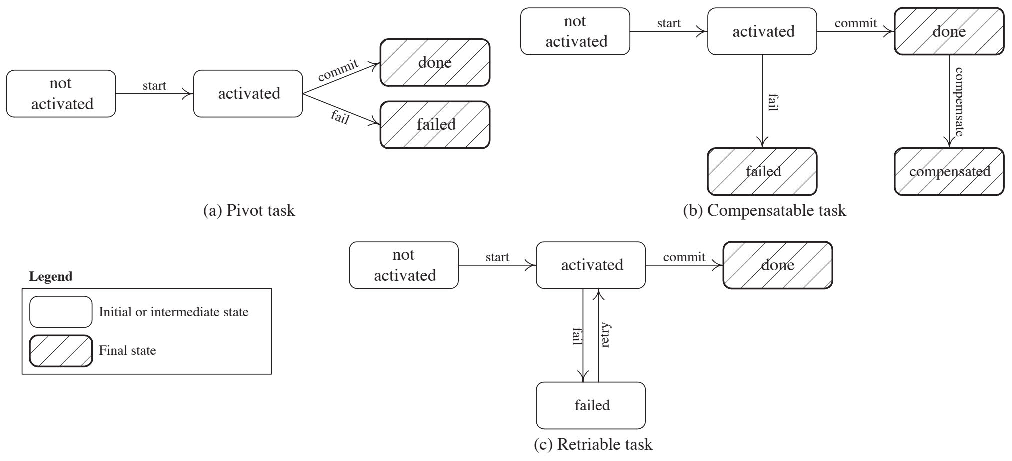


FIGURE 3 Task's lifecycle per transactional property

3.3.2 | Announcement

The announcement step consists of supporting the BP owner contact providers of cloud resources using one of the following strategies:

- Global strategy: demands of cloud resources for all tasks are concurrently broadcasted to all providers. Those providers that are capable of satisfying these demands, whether for a task, a group of tasks, or all tasks, respond to the BP owner.
- Partial strategy: demands of cloud resources for selected tasks (e.g., using finish-to-start dependency as a selection criterion) are gradually broadcasted to a restrictive group of providers. This restrictive group is now involved in the execution of the tasks that precede the selected tasks. Allocating the same clouds to successor tasks promotes cost saving since this avoids additional communication costs (when transferring data) between separate clouds. However, this could result into overlooking other clouds that could have better resource offers.

3.3.3 | Pricing

The pricing step consists of assisting providers define prices for their cloud resources taking into account these resources' types (i.e., computation, storage, and network) and properties (i.e., limited/unlimited, renewable, expandable, renewable&expandable, and shareable/non-shareable). We suggest 3 pricing categories that are commonly used in the airline industry for economy tickets (e.g., emirates.com) (Table 1): saver, flex, and flex+. Each category promotes a different refund, change, and loyalty policy.

- Saver is the lowest price due to no refund, no change, and no credits.
- Flex+ is the highest price due to refund and change at no-cost and maximum credits.
- Flex is between saver and flex+ prices due to refund and change with a fee and minimum credits.

[8](Z2S: please use blue color) The cancelation targets the outcomes of executing a task (kind of compensation) and not the task itself. Canceling a task could lead to the failure of the whole process, should this task be defined as critical.

[9]Ideally, cloud resources that are at least expandable should be considered so, that, the same cloud resources continue to be used.

[10]Any cloud resource that meets the BP owner's demands is fine.

TABLE 1 Price breakdown for a cloud resource

Policy	Prices		
	saver: lowest	flex: in-between	flex+: highest
Refund	No refund	With a fee	No fee
Change	No change	With a fee	No fee
Loyalty	No credits	Minimum credits	Maximum credits

In Table 1, refund and change strategies target BP owners who would like to cancel/postpone the execution of (some/all) tasks, to select other cloud providers after agreeing with certain providers, and/or to revise their demands of cloud resources. For the loyalty policy that is left as future work, a BP owner earns credits like free execution time, free storage capacity, and/or free communication bandwidth that can be redeemed instead of paying for cloud resources. Levels of credits vary per type of price with flex+ having the highest level. It is worth noting that some prices (e.g., saver) for certain cloud resources might not be available or sold-out at the discretion of these resources' providers.

Cloud providers submit 3 prices (i.e., saver, flex, and flex+) for their cloud resources. Each resource's price is characterized by 2 properties $p_{(i,j)}$, where subscript i refers to either l (limited), ul (unlimited), r (renewable), e (expandable), or re (renewable&expandable) and subscript j refers to either s (shareable) or ns (non-shareable). For example, saver.$p^{res_c}_{l\&s}$ corresponds to the saver price of a computational resource (res_c) that exposes limited and shareable properties (i.e., res_c can be used for a limited time period and/or with a limited capacity and will be shared other processes' tasks). Another example is flex.$p^{res_s}_{r\&e\&ns}$ that corresponds to the flex price of a storage resource (res_s) that exposes renewable, expandable, and non-shareable properties.

Along with the aforementioned price breakdown, two assumptions are made:

- A resource offer of a cloud provider always matches a BP owner's resource demand whether requested CPU power, requested database capacity, or requested bandwidth capacity (i.e., offered resources are equal or more than requested resources). This means that combining cloud resources to answer a provider's demand is out-of-scope this work but is worth investigating in the future.
- Prices of resources do not include data transfer-fee between clouds and from/to the Internet. This fee is transparently added to the total price of using cloud resources and dependent on the final number of clouds that would support the execution of BP fragments (Section 3.3.5).

3.3.4 | Allocation

The allocation step consists of assisting BPs' owners identify the appropriate resources for their tasks based on (*i*) the properties of resources (i.e., l/ul, r, e, r&e, and s/ns), (*ii*) the restrictions on tasks (i.e., budget, deadline, and exclusivity), and (*iii*) the transactional properties of tasks (i.e., pivot, retriable, and compensatable). The allocation step relies on Algorithms 1 and 2 for selecting eligible resources (S^t_{eres}) and allocating these resources to tasks, respectively.

Algorithm 1 processes the responses that resources' providers submit to the BP owner with respect to one of the announcement strategies discussed in Section 3.3.2. These responses form an initial set of resources (S^t_{res})[11] that is assigned, at initialization time, to the set of eligible resources (line 7). The algorithm proceeds as follows whose complexity is $O(n)$ (where n is the number of tasks in a BP) and hence, is linear:

1. If a task has an exclusivity restriction (line 8), then all shareable resources will be dropped from the set of eligible resources (line 9).
2. If a task has a budget restriction (line 11), then all resources that are over the budget (including the tolerance ratio) will be dropped from the set of eligible resources (line 12).
3. If a task has a deadline restriction (line 14), then all resources that do not meet the deadline (including the tolerance ratio) will be dropped from the set of eligible resources (line 15).

Algorithm 2 uses Algorithm 1's outcome, i.e., set of eligible resources (S^t_{eres}), to now support the BP owner select an optimal price offering according to the transactional properties of tasks. As stated earlier, we use optimal because the final decision of selecting a particular resource depends on the BP owner's goals whether lowering cost, guaranteing execution despite cost, or anything else. Algorithm 2 consists of 2 parts.

- Part 1 (lines 6-13) allows selecting a resource's price based on the transactional property of a task so, that, this task is executed at run-time (line 14).

[11] If the global announcement strategy is adopted, then the set of resources S^t_{res} remains the same throughout the whole exercise of assigning tasks to cloud resources. Otherwise, i.e., partial announcement strategy, the set of resources S^t_{res} varies from one round of execution to another since collecting providers' responses is sequentially done.

Algorithm 1 Selecting eligible resources

```
 1: Input
 2: Process p, a set of resources $S^{t_i}_{res}$ for each task formed upon providers' responses
 3: Output
 4: Final set of eligible resources $S^{t_i}_{eres}$ for each task
 5: for all tasks ($t_{1..n}$) in p do
 6:     //initialization
 7:     $S^{t_i}_{eres} = S^{t_i}_{res}$
 8:     if $t_i$ has exclusivity restriction then
 9:         $S^{t_i}_{eres} = S^{t_i}_{eres} - \{$shareable resources$\}$
10:     end if
11:     if $t_i$ has budget restriction then
12:         $S^{t_i}_{eres} = S^{t_i}_{eres} - \{$resources that are over the budget$\}$
13:     end if
14:     if $t_i$ has deadline restriction then
15:         $S^{t_i}_{eres} = S^{t_i}_{eres} - \{$resources that do not meet the deadline$\}$
16:     end if
17: end for
```

Algorithm 2 Allocating eligible resources to tasks

```
 1: Input
 2: Process p, set of eligible resources $S^{t_i}_{eres}$ along with their price offerings as per Table 1
 3: Output
 4: Final Set S($t_i$, $res^{t_i}_{op}$, $p^{res}_{j,k}$)
 5: for all tasks ($t_{1..n}$) in p do
 6:     if $t_i$ is pivot then //"?" means any type of cloud resource property (i.e., l/ul, r, e, r&e, and s/ns).
 7:         $res^{t_i}_{op}$ = select resource from $S^{t_i}_{eres}$ with optimal price among saver.$p^{res?}_{?,?}$/flex.$p^{res?}_{?,?}$/flex+.$p^{res?}_{?,?}$
 8:     else if $t_i$ is retriable then
 9:         $res^{t_i}_{op}$ = select resource from $S^{t_i}_{eres}$ with optimal price among flex.$p^{res?}_{r,?}$/flex+.$p^{res?}_{re,?}$
10:     else
11:         //$t_i$ is compensatable
12:         $res^{t_i}_{op}$ = select resource from $S^{t_i}_{eres}$ with optimal price among flex.$p^{res?}_{e,?}$/flex+.$p^{res?}_{re,?}$
13:     end if
14:     Execute $t_i$
15:     while $t_i$ is retriable and its execution fails do
16:         if $res^{t_i}_{op}$ can be renewed to re-execute $t_i$ then
17:             $res^{t_i}_{op}$ subject to meeting any restriction on $t_i$
18:         else
19:             $res^{t_i}_{op}$ = select resource from $S^{t_i}_{eres}$ with optimal price among flex.$p^{res?}_{r,?}$/flex+.$p^{res?}_{re,?}$
20:         end if
21:         Execute $t_i$
22:     end while
23:     if $t_i$ is compensatable and its effects need to be canceled then
24:         if $res^{t_i}_{op}$ can continue to be used in order to cancel $t_i$ then
25:         else
26:             if $res^{t_i}_{op}$ can be expanded to cancel $t_i$ then
27:                 expand $res^{t_i}_{op}$ subject to meeting any restriction on $t_i$
28:                 Cancel $t_i$
29:             else
30:                 // the required resource is used only to cancel $t_i$
31:                 $res^{t_i}_{op}$ = select resource with optimal price among saver.$p^{res?}_{?,?}$
32:                 Cancel $t_i$
33:             end if
34:         end if
35:     end if
36: end for
```

1. If a task is pivot (Fig. 3-a), then the optimal price among all eligible resources' saver, flex, and flex+ offerings is selected (line 7). Since a pivot task's execution could either succeed or fail, the BP owner is less concerned with this execution's outcome.
2. Otherwise, if a task is retriable (Fig. 3-c), then the optimal price among all either renewable or renewable&expandable eligible resources' flex and flex+ offerings is selected (line 9). Since a retriable task's execution should succeed, the BP owner should consider the resources that could be renewed "hoping" that the same resources would continue to be used if there is a need of re-executing the task. At this point of time, the number of times that a task is going to be re-executed, is unknown, so selecting either flex or flex+ prices would minimize the additional cost of changing the request of using another resource.

3. Otherwise (the task is compensatable (Fig. 3-b)), the optimal price among all either expandable or renewable&expandable eligible resources' flex and flex+ offerings is selected (line 12). Since a compensatable task's execution could be successfully canceled, the BP owner should consider the resources that could be extended "hoping" that the same resources continue to be used in case of canceling this execution's effects. Selecting either flex or flex+ prices would minimize the additional cost of changing the request of using another resource.

- Part 2 (lines 15-35) depends on the outcome of executing a task as per line 14. 2 cases could arise:

  1. If a retriable task's execution fails (line 15), then the BP owner would ideally like to continue using the same resource that was initially assigned to this task provided that this resource's renewal meets all the restrictions on this task (lines 16-17). Otherwise, the selection of another resource among all flex and flex+ offerings is initiated (line 19). Upon a resource's successful selection, the task is re-executed (line 21). It is worth noting that re-executing a task could fail a certain number of times before success. This justifies the selection among flex and flex+ offerings.
  2. If a compensatable task's execution effects need to be canceled (line 23), then the BP owner would ideally like to continue using the same resource that was initially assigned to this task either without expanding this resource (lines 24-25) or with requesting to expand this resource (lines 27-29; the expansion should meet all restrictions on the task). Otherwise, the selection of another resource among all saver offerings is initiated (line 31). Upon resource successful selection, the task is canceled (line 32). As the cancelation must succeed, this justifies the use of saver offerings.

The complexity of Algorithm 2 also is $O(n)$ (where n is the number of tasks in a BP), although the number of re-executions could be excessive due to the retriable requirement (a maximum number of retrials would be set).

3.3.5 | Formation

The formation step consists of developing task fragments that correspond to the clouds offering the necessary resources to all these tasks. This step uses the outputs of Algorithm 2 that are BP tasks and their cloud resources permitting to optimize a certain function for instance, total price of consuming cloud resources in our work. Another example of optimization could be total execution time of process and usage of cloud resources [31]. A fragment is about grouping all tasks that use cloud resources from the same cloud provider. Formally, a Fragment (F) is a couple of a set of tasks t_i and the cloud provider pro_j offering the necessary resources res_k to these tasks in an optimal way: $F =< \{(t_i, res_k)\}, pro_j >$ where $\forall k, res_k \in \{pro_j\}$.

4 | IMPLEMENTATION OF THE FRAGMENTATION APPROACH

This section describes our system for BP fragmentation over the clouds in terms of how we developed the system in compliance with the approach's steps and how we tested the system using the credit application BP (Section 2.3). The system consists of (*i*) a Java Eclipse plugin that extends BPMN2-modeler Eclipse plugin[12] in support of the approach's specification step and (*ii*) a Web application in support of the remaining steps. Separate Web interfaces are made available to BP designers and cloud providers, respectively.

4.1 | System's step-by-step use

Using the Java Eclipse plugin (Fig. 4), the designer proceeds, as per the specification step, with modeling a BP in terms of tasks and dependencies between tasks (e.g., pre-requisite and co-requisite), restrictions on tasks (i.e., budget, deadline, and exclusivity), and transactional properties of tasks (i.e., pivot, retriable, and compensatable). This modeling is illustrated with Fig. 4 showing the extended tool palette (right side) that adds 2 categories of elements to the BPMN2 modeler: cloud resources and cloud tasks. The former extends TextAnnotation BPMN2 meta-class in order to specify a cloud resource (computation, storage, or communication) and its properties (e.g., limited and renewable). The latter extends regular tasks by defining their required cloud resources, transactional properties, and restrictions. A demo of the Java Eclipse plugin is available at www.redcad.org/projects/BPFragmentation/demo.mp4.

For the remaining steps of the fragmentation approach (i.e., announcement, pricing, allocation, and formation), BP designers and cloud providers use the Web application (www.redcad.org/projects/BPFragmentation/app)[13] to specify their needs and offers, respectively. First, the designer imports the specified BP, saved as an XML document by the plugin, into the Web application. Then, the designer selects the announcement strategy

[12] www.eclipse.org/bpmn2-modeler.
[13] A user manual is available at www.redcad.org/projects/BPFragmentation/guidePBfragmentation.pdf.

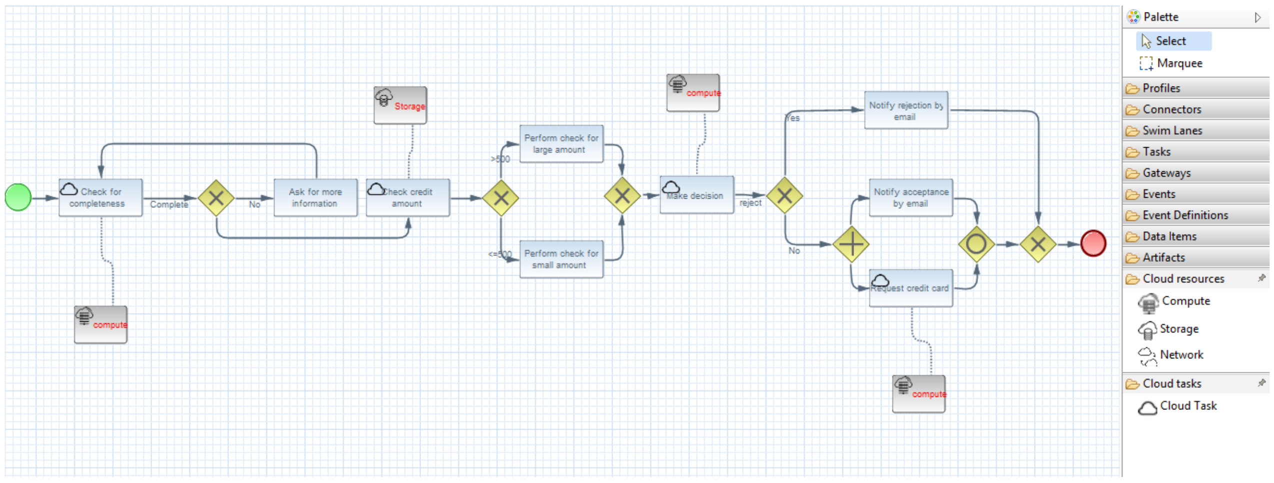


FIGURE 4 Illustration of the use of Java Eclipse plugin

(i.e., global and/or partial) for contacting potential cloud providers. Using the same Web application, the cloud providers specify their resources' prices and properties. We recall that prices vary from one designer's request to another and are offered at the discretion of each resource provider. For instance, some prices like flex are not offered to certain BP designers. Technically speaking, all information related to BP tasks and cloud resources are saved in the same XML document to ease the communication between all stakeholders. Once all the necessary inputs are gathered in the XML document, the BP designer launches Algorithm 1 (bottom of Fig. 5) and then Algorithm 2 so, that, eligible then optimal cloud resources are identified, which leads to forming the BP fragments.

4.2 | Experiments

We use the credit application BP of Section 2.3 to experiment the system for BP fragmentation over the clouds. We report on these experiments according to the fragmentation approach's steps.

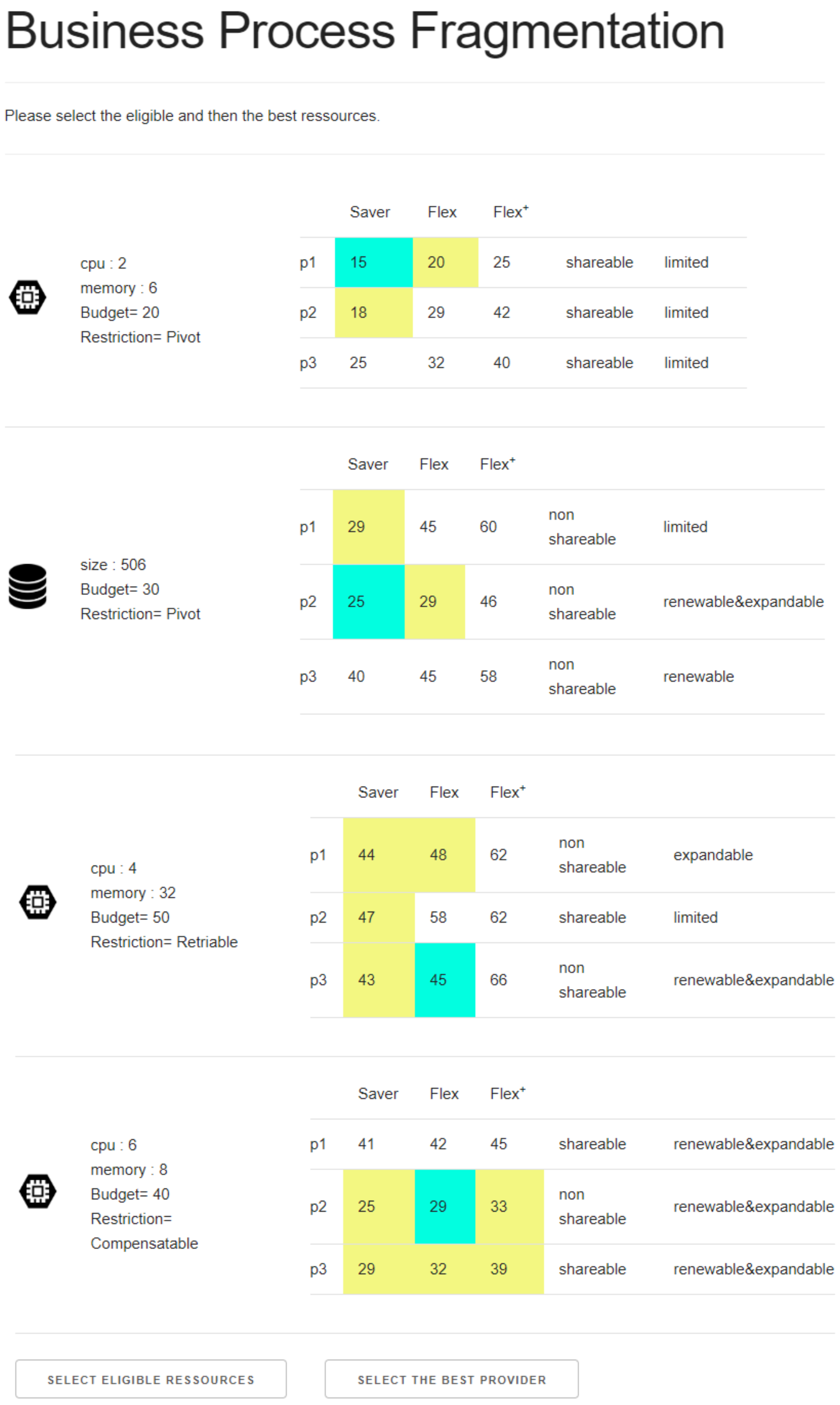
1. Specification step is about the actions that a BP designer executes as per Section 4.1. The results of this execution are summarized in Table 2. A 10% tolerance ratio was set randomly during the experiments (we recall that different ratios mean different eligible resources). For the sake of illustration, only 4 tasks of the credit application BP needed cloud resources.

TABLE 2 Definition of credit application BP's tasks

Execution aspects to be aware of	Tasks			
	t_1: check for completeness	t_2: check credit amount	t_3: make decision	t_4: request credit card
Required resources	Computation	Storage	Computation	Computation
Resources' specifications	Memory:6 CPU:2	Capacity:50	Memory:32 CPU:4	Memory:64 CPU:8
Restrictions	Budget: 20 Deadline: no Exclusivity: yes	Budget: 30 Deadline: no Exclusivity: yes	Budget: 55 Deadline: no Exclusivity: yes	Budget: 60 Deadline: no Exclusivity: yes
Transactional properties	Pivot	Pivot	Retriable	Compensatable

2. Announcement step is about contacting cloud providers (3 during the experiments $pro_{\{1,2,3\}}$) using the global and partial strategies. This contact takes place thanks to the XML document (mentioned in Section 4.1) that all stakeholders have access to.

3. Pricing step allows setting the prices of cloud resources. On the one hand, Fig. 5 summarizes the offers of cloud providers that the BP designer receives with respect to the global announcement strategy. On the other hand, Table 3 (instead of 5 screenshots like the one shown in Fig. 5) summarizes the offers of cloud providers that the BP designer incrementally (i.e., not all at once) receives with respect to the partial announcement strategy.



Business Process Fragmentation

Please select the eligible and then the best ressources.

cpu : 2
memory : 6
Budget= 20
Restriction= Pivot

	Saver	Flex	Flex$^+$		
p1	15	20	25	shareable	limited
p2	18	29	42	shareable	limited
p3	25	32	40	shareable	limited

size : 506
Budget= 30
Restriction= Pivot

	Saver	Flex	Flex$^+$		
p1	29	45	60	non shareable	limited
p2	25	29	46	non shareable	renewable&expandable
p3	40	45	58	non shareable	renewable

cpu : 4
memory : 32
Budget= 50
Restriction= Retriable

	Saver	Flex	Flex$^+$		
p1	44	48	62	non shareable	expandable
p2	47	58	62	shareable	limited
p3	43	45	66	non shareable	renewable&expandable

cpu : 6
memory : 8
Budget= 40
Restriction=
Compensatable

	Saver	Flex	Flex$^+$		
p1	41	42	45	shareable	renewable&expandable
p2	25	29	33	non shareable	renewable&expandable
p3	29	32	39	shareable	renewable&expandable

FIGURE 5 Total availability of cloud providers' offers due to the global strategy

TABLE 3 Incremental availability of cloud providers' offers due to the partial strategy

		Prices			Properties	
Offers for	Providers	saver	flex	flex+	sh/n-sh	l/e/r/re
t_1	pro_1	15	20	25	sh	l
	pro_2	18	29	42	sh	e
	pro_3	25	32	40	sh	r&e
Contacting pro_1 for t_2 after its selection for t_1						
t_2	pro_1	29	45	60	n-sh	l
Contacting pro_1 again for t_3 after its selection for t_2						
t_3	pro_1	44	48	52	n-sh	e
Contacting $pro_{2,3}$ for t_3 after discarding p_1's offers						
t_3	pro_2	47	58	62	sh	l
	pro_3	43	45	66	n-sh	r&e
Contacting pro_3 for t_4 after its selection for t_3						
t_4	pro_3	29	32	39	sh	r&e

4. Allocation step is initiated after collecting either totally or incrementally providers' offers of cloud resources and happens thanks to Algorithms 1 and 2.

   Algorithm 1 identifies eligible resources. In the credit application BP, the unique constraint to satisfy is that the proposed price should be lower than the combined budget restriction and tolerance ratio. In Fig. 5, the eligible resources are highlighted in yellow.

   Algorithm 2 selects the optimal prices (highlighted in turquoise in Fig. 5) of the required cloud resources. In the case of compensatable or retriable tasks, the selected resources can change during the execution of these tasks like discussed in this algorithm's lines 15–34. To simulate the change of cloud resources, we put together an in-house process log that reports on the process execution. This log is shown in Listing 1 and targets the 4 tasks that need cloud resources.

Global announcement: according to Appendix 1-Listing 1, the chronology of identifying eligible and, then, optimal resources occurs as follows:

- t_1 is pivot and is successfully executed using a limited resource (lines 4–15 in Listing 1). The eligible resources belong to pro_1 and pro_2. Indeed, their prices are less than t_1's budget. Thus, the lowest price (in case of equal, random is used) of all pro_1's and pro_2's cloud resources will be selected by the BP engineer among saver, flex, and flex+. As per Fig. 5, the optimal price is saver and is highlighted in turquoise.
- t_2 is pivot and is successfully executed using a limited resource (lines 17–28). The eligible and unique optimal resource is offered by pro_2 (the price proposed by pro_2 is less that pro_1).
- t_3 is retriable. The optimal resource belongs to pro_3 along with a flex price and renewable&expandable property. Unfortunately, t_3 fails for the first time using pro_3's resource as per lines 30–41 in Listing 1. The good news is that this cloud resource is renewable and the renewal can be afforded thanks to flex price so t_3 is re-executed again as per line 17 in Algorithm 2 and lines 44–55 in Listing 1. For the second time, t_3 fails (lines 58–69). According to Algorithm 2's line 19, another eligible resource needs to be selected when the resource cannot be renewed again. The resource proposed by pro_1 with a flex price of \$48 is selected since it is the optimal resource.
- t_4 is successfully executed from the first time using a renewable&expandable resource (lines 71–82). The optimal resource for t_4 is proposed by pro_2 as shown in Fig. 5.

Partial announcement: according to Appendix 1-Listing 1, the chronology of identifying eligible and, then, optimal resources occurs as follows:

- t_1 is pivot (Table 3, t_1 rows). pro_1 is selected along with its saver, flex, and flex+ prices (line 7, Algorithm 2).
- Now, the cloud resource demand for t_2 is sent to pro_1, only. The designer receives the offer from pro_1 (Table 3, t_2 rows). Algorithms 1 and 2 are executed again leading to accepting pro_1's offer as an optimal resource.
- As a result of selecting pro_1 again, the cloud-resource demand for t_3 is also sent to pro_1 leading to its selection. Assuming that t_3 failed twice and that in the second time the designer needs to select another resource (pro_1 has been discarded), then, a cloud-resource demand is sent to both pro_2 and pro_3. pro_3 is selected along with its flex and flex+ prices after considering that t_3 is retriable.
- Now, the demand for t_4 is sent to pro_3, only. The designer receives the offer from pro_3 (Table 3, t_4 rows).

5. Formation step leads to identifying groups of tasks per cloud depending on the announcement strategies. We recall that the fragments are declared **final** once the execution of all tasks is **successfully complete**.

   - 2 fragments are formed following the global announcement:
     $F_1(\text{global}) =< \{(t_1, res_1), (t_3, res_3)\}, pro_1 >$ and $F_2(\text{global}) =< \{(t_2, res_2), (t_4, res_4)\}, pro_2 >$.
   - 2 fragments are formed following the partial announcement:
     $F_1(\text{local}) =< \{(t_1, res_1), (t_2, res_1)\}, pro_1 >$ and $F_2(\text{local}) =< \{(t_3, res_3), (t_4, res_4)\}, pro_3 >$.

Independently of the number of fragments, the contents of fragments in terms of tasks very depending on the announcement strategy.

For the needs of benchmarking, we considered a second credit application BP instance whose execution is captured in a new log included in Appendix 2-Listing 2. Since the specification and announcement steps are similar to those discussed earlier, we discuss below the findings of the pricing, allocation, and fragmentation steps for this second BP instance. The offers of the providers with respect to the global announcement strategy are in Fig 5 and the offers of the providers for the partial announcement strategy have been revised as per Table 4.

TABLE 4 Incremental availability of cloud providers' offers due to the partial strategy (experiment #2)

		Prices			Properties	
Offers for	Providers	saver	flex	flex+	sh/n-sh	l/e/r/re
t_1	pro_1	15	20	25	sh	l
	pro_2	18	29	42	sh	e
	pro_3	25	32	40	sh	r&e
Contacting pro_1 for t_2 after selecting pro_1 for t_1						
t_2	pro_1	29	45	60	n-sh	l
Contacting pro_1 again for t_3 after selecting pro_1 for t_2						
t_3	pro_1	44	48	52	n-sh	e
Contacting pro_1 for t_4 after after selecting pro_3 for t_3						
t_4	pro_1	45	53	62	sh	r&e

In the allocation step, the execution of Algorithm 1 allows selecting the eligible resources, which are similar to the first experimentation, since the same cloud providers' offers are selected. However, the outcomes of executing Algorithm 2 depend on the execution process and announcement strategy.

Global announcement: according to Appendix 2-Listing 2, the chronology of identifying eligible and, then, optimal resources occurs as follows:

- t_1 and t_2 are both pivot and are successfully executed using limited resources (lines 4-15 and lines 17-28 in Listing 2). There are no change compared to the first experimentation. The offer of pro_1 is selected for t_1 and the offer of pro_2 is selected for t_2.
- t_3 is retriable. As shown in Listing 2 (lines 30-55), t3 fails only once using the resource of pro_3, since this resource is renewable&expandable and proposed with flex price.
- t_4 is compensatable. It is successfully executed from the first time using a renewable&expandable resource (lines 58–68). However, this task needs to be canceled as shown in lines 58–68 in Listing 2. The optimal resource for t_4 is proposed by pro_2 as shown in Fig. 5. We suppose that this resource can continue to be used to cancel t_4.

Partial announcement: according to Appendix 2-Listing 2, the chronology of identifying eligible and, then, optimal resources occurs as follows:

- Similar to the first experimentation, pro_1 is selected for t_1 and t_2. (Table 4, t_1 and t_2 rows). These two tasks are pivot and successfully executed from the first time.
- Since pro_1 is selected for t_2, the cloud-resource demand for t_3 is also sent to pro_1. Although t_3 failed once, pro_1 continues to be used because it is retriable&expandable.
- The demand for t_4 is, also, sent to pro_1. The offer of pro_1 is selected as shown in Table 4 for t_4 rows. The cancelation of this task is performed by the same resource (lines 57–74, Listing 2).

For the formation step, the following fragments are identified:

- 3 fragments are formed following the global announcement:
  $F_1(\text{global}) =< \{(t_1, \text{res}_1)\}, \text{pro}_1 >$, $F_2(\text{global}) =< \{(t_2, \text{res}_2), (t_4, \text{res}_4)\}, \text{pro}_2 >$, and $F_3(\text{global}) =< \{(t_3, \text{res}_3)\}, \text{pro}_3 >$. Compared to the first experiment, 2 fragments compared to 3 fragments are selected.
- 1 fragment is formed following the partial announcement:
  $F_1(\text{local}) =< \{(t_1, \text{res}_1), (t_2, \text{res}_2), (t_3, \text{res}_3), (t_4, \text{res}_4)\}, \text{pro}_1 >$. Compared to the first experiment, 2 fragments compared to 1 fragment are selected.

Table 5 summarizes the 2 experiments that were carried out to demonstrate the BP fragmentation approach with focus on benchmarking the announcement strategies' cost of resources and number of fragments. The 1st column presents the tasks forming the BP. The 2nd column presents a potential scenario of executing these tasks. In the first experiment, t_2 is executed with success at the first time, while t_3 fails twice before it is executed with success at the third time with a new resource. This illustrates the retriable property and resource no-availability after being consumed. In the 3rd and 4th columns (resp., global and partial announcement strategies), the cost of using resources is calculated taking into account the tasks' transactional properties and execution outcomes (whether success, failure, or failure then success).

In these 2 experiments, the cost of resources is lower for the global announcement compared to the local announcement. However, this cannot be generalized. The cost depends on many parameters such as announcement strategy, transactional properties of tasks, availability properties of resources, outcomes of executing tasks (success *versus* failure), etc.

TABLE 5 Summary of the experiments

Experiment #1		Announcement	
Task	Description	Global	Partial
t_1	success at the first time	15	15
t_2	success at the first time	25	29
t_3	failure twice then success with a new resource	2*45+48	2*48+45
t_4	success at the first time	29	32
	Cost of resources	207	217
	Number of fragments	2	2

Experiment #2		Announcement	
Task	Description	Global	Partial
t_1	success at the first time	15	15
t_2	success at the first time	25	29
t_3	failure once then success with the same resource	45	48
t_4	success at the first time then compensated	29	42
	Cost of resources	114	134
	Number of fragments	3	1

5 | CONCLUSION

We presented the design and development of an approach for decomposing business processes into fragments that would run over cloud resources of type infrastructure (IaaS). The approach addresses the concerns of both process owners and cloud providers. On the one hand, owners would like to execute processes' tasks according to strict budget and deadline, and sometimes in isolation from other competing processes. This execution is, also, constrained by some transactional properties of tasks referred to as pivot, retriable, and compensatable. For instance, a retriable task could require more cloud resources, should its execution need to be retried a couple of times. Should these resources come from the same providers? If not, how does this impact the execution of the whole process? These are some questions raised and addressed in this paper. On the other hand, providers would like to be competitive by offering flexible prices for their clouds, so they cater to the needs and requirements of each process owner. These prices are constrained by some cloud properties referred to as un/limitedness, not/renewability, and not/shareability. What happens if a cloud resource is limited and hence, is no longer available for a task whose execution needs to be retried a couple of times? Would this resource be enough to cover all the execution retrials? Again, these are some questions raised and addressed in this paper, too. Our fragmentation approach consists of 5 steps: specification of owners' needs of clouds, announcement of these needs to potential providers of clouds, definition of prices according to cloud properties, allocation of necessary clouds to tasks, and formation of fragments based on this allocation. Each step was detailed in terms of who does what and what outcomes to expect. The technical feasibility of this approach has been demonstrated through a credit-application case study and a system that extends BPMN2-modeler Eclipse plugin allowing business processes' owners and clouds' providers to specify their restrictions and properties, respectively. The system also supports owners reach out to providers using global and/or local announcement strategies. 2 sets of experiments have been carried out, as well, explaining how the fragmentation approach's 5 steps are performed.

In term of future work, we would like, first, to extend the fragmentation approach to other forms of cloud resources for instance, platform (PaaS) and software (SaaS). Are un/limitedness, not/renewable, and not/shareable properties still appropriate for these resources and what other types of restrictions, besides budget, deadline, and exclusivity, could be considered in the context of these resources? Second, we would like to look into the mapping BP fragment constructs onto languages, e.g., CAMEL [32] and TOSCA [33], that are dedicated for modeling cloud-based applications. Finally, we would like to examine the role of cloud resources in supporting IoT aware business processes. Could these processes be fragmented? If yes, how? And, how could cloud address the limited computational capabilities of (some) things?

References

1. Khajeh-Hosseini A, Greenwood D, Sommerville I. Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. In: Proceedings of the 2010 IEEE Third International Conference on Cloud Computing (CLOUD'2010). IEEE. ; 2010; Miami, FL, USA.

2. Motahari-Nezhad H, Stephenson B, Singha S. Outsourcing Business to Cloud Computing Services: Opportunities and Challenges. *IEEE IT Professional, Special Issue on Cloud Computing* 2009; 11(2).

3. Xu B, Xu L, Fei X, Jiang L, Cai H, Wang S. A Method of Demand-driven and Data-centric Web Service Configuration for Flexible Business Process Implementation. *Enterprise Information Systems* 2017; 11(7).

4. Logicworks . Why Vendor Lock-In Remains a Big Roadblock to Cloud Success. www.cloudcomputing-news.net/news/2016/sep/01/vendor-lock-in-is-big-roadblock-to-cloud-success-survey-finds; September 2016 (checked out in March 2019).

5. Ahmed-Nacer A, Goettelmann E, Youcef S, Tari A, Godart C. Obfuscating a Business Process by Splitting Its Logic with Fake Fragments for Securing a Multi-cloud Deployment. In: Proceedings of the IEEE World Congress on Services (SERVICES'2016). IEEE. ; 2016; San Francisco, CA, USA.

6. Dong M, Li H, Ota K, Yang LT, Zhu H. Multicloud-Based Evacuation Services for Emergency Management. *IEEE Cloud Computing* 2014; 1(4): 50–59.

7. Dimick D. As World's Population Booms, Will its Resources be Enough for Us?. https://tinyurl.com/y92vfb5q; October 2014. National Geographic.

8. Weske M. *Business Process Management - Concepts, Languages, Architectures,-driven*. 2nd edition, Springer . 2012.

9. Bucchiarone A, Marconi A, Pistore M, Raik H. A Context-Aware Framework for Dynamic Composition of Process Fragments in the Internet of Services. *Journal of Internet Services and Applications* 2017; 8(6).

10. Mezni H, Kbekbi M. Reusing Process Fragments for Fast Service Composition: A Clustering-based Approach. *Enterprise Information Systems* 2019; 13(1): 34–62.

11. Mancioppi M, Danylevych O, Karastoyanova D, Leymann F. Towards Classification Criteria for Process Fragmentation Techniques. In: Proceedings of the Business Process Management Workshops (BPM'2011). Springer. ; 2011; Ulm, Germany.

12. Maalouf E, Di Zuzio M, Sokhn M. Business Process Fragmentation for Enhancing Process Modeling. In: Proceedings of the 2014 Conference on Electronic Governance and Open Society: Challenges in Eurasia (EGOSE'2014). ACM. ; 2014; St. Petersburg, Russian Federation.

13. Hou S, Zhao S, Cheng B, Cheng Y, Chen J. Fragmentation and Optimal Deployment for IoT-aware Business Process. In: Proceedings of the 2016 IEEE International Conference on Services Computing (SCC'2016). IEEE. ; 2016; San Francisco, USA.

14. Anis Zemni M, Benbernou S, Soror S. Privacy-Preserving Business Process Fragmentation for Reusability. In: Proceedings of Atelier Protection de la Vie Privée/Géolocalisation et Vie Privée (APVP'2011). ; 2011; Soreze, France.

15. Hens P, Snoeck M, Poels G, De Backer M. Process Fragmentation, Distribution and Execution Using an Event-based Interaction Scheme. *Journal of Systems and Software* 2014; 89.

16. Fdhila W, Indiono C, Rinderle-Ma S, Reichert M. Dealing with Change in Process Choreographies: Design and Implementation of Propagation Algorithms. *Information Systems* 2015; 49.

17. Xue G, Liu J, Wu L, Yao S. A Graph-based Technique of Process Partitioning. *Journal of Web Engineering* 2018; 17(1&2).

18. Pourmasoumi A, Kahani M, Bagheri E. Mining Variable Fragments from Process Event Logs. *Information Systems Frontiers* 2017; 19(6).

19. Bunge M. *Treatise on Basic Philosophy, Ontology I: The Furniture of the World*. 3. Springer Netherlands . 1977.

20. Ahmed-Nacer A, Goettelmann E, Youcef S, Tari A, Godart C. Business Process Design by Reusing Business Process Fragments from the Cloud. In: Proceedings of the 8th IEEE International Conference on Service-Oriented Computing and Applications (SOCA'2015). IEEE. ; 2015; Rome, Italy.

21. Ahmad F, Sarkar A. QaaS (Quality as a Service) Model for Web Services Using Big Data Technologies. *Enterprise Information Systems* 2017; 11(9).

22. Graiet M, Mammar A, Boubaker S, Gaaloul W. Towards Correct Cloud Resoruce Allocation in Business Processes. *IEEE Transactions on Services Computing* Januray/February 2017; 10(1).

23. Hachicha E, Gaaloul W, Maamar Z. Social-Based Semantic Framework for Cloud Resource Management in Business Processes. In: Proceedings of the IEEE International Conference on Services Computing (SCC'2016). IEEE. ; 2016; San Francisco, CA, USA.

24. Yunchang Liu Y, Chunlin Li C, Youlong Luo Y, Yanling Shao Y, Jing Zhang J. Scheduling Multimedia Services in Cloud Computing Environment. *Enterprise Information Systems* 2018; 12(2).

25. Schulte S, Schuller D, Hoenisch P, Lampe U, Dustdar S. Cost-driven Optimization of Cloud Resource Allocation for Elastic Processes. *International Journal of Cloud Computing* 2013; 1(2).

26. Rozinat A, Wynn M, Aalst v. dW, Hofstede tA, Fidge C. Workflow Simulation for Operational Decision Support Using Design, Historic and State Information. In: Proceedings of the 6th International Conference on Business Process Management (BPM'2008). Springer. ; 2008; Milan, Italy.

27. Object Management Group (OMG) . Business Process Model and Notation. www.omg.org/spec/BPMN/2.0.2; .

28. Caron F, Vanthienen J. Exploring Business Process Modelling Paradigms and Design-Time to Run-Time Transitions. *Enterprise Information Systems* 2016; 10(7).

29. Maamar Z, Faci N, Sakr S, Boukhebouze M, Barnawi A. Network-based Social Coordination of Business Processes. *Inf. Sys.* 2016; 58.

30. Little M. Transactions and Web Services. *Communications of the ACM* 2003; 46(10).

31. Hedhli A, Mezni H. A DFA-based Approach for the Deployment of BPaaS Fragments in the Cloud. *Concurrency and Computation: Practice and Experience* 2018: 1–26.

32. Rossini A, Kritikos K, Nikolov N, et al. The Cloud Application Modelling and Execution Language (CAMEL). https://tinyurl.com/ycsqanae; 2017.

33. Palma D, Spatzier T. Topology and Orchestration Specification for Cloud Applications (TOSCA). https://tinyurl.com/yatr5lxv; 2013.

Appendix 1

Listing 1: Business process log for experiment #1

```
<trace>
  <!-- T1 --> <event>
      <string key="concept:name" value="T1" property="Pivot"/>
      <string key="lifecycle:status" value="activated"/>
      <resource key="R1" property="Limited"/>
      <date key="time:timestamp" value="2018-11-12T00:48:38.000+01:00"/>
  </event>
  <event>
      <string key="concept:name" value="T1" property="Pivot"/>
      <string key="lifecycle:transition" value="done"/>
      <resource key="R1" property="Limited"/>
      <date key="time:timestamp" value="2018-11-12T00:58:38.000+01:00"/>
  </event>
```

```
    <!−− T2 −−> <event>
        <string key="concept:name" value="T2" property="Pivot"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R2" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T01:48:38.000+01:00"/>
    </event>
    <event>
        <string key="concept:name" value="T2" property="Pivot"/>
        <string key="lifecycle:transition" value="done"/>
        <resource key="R2" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T03:40:00.000+01:00"/>
    </event>
    <!−− T3 −−> <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:28:38.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="failed"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:31:03.000+01:00"/>
    </event>
    <!−− Renew R3 since T2 is retriable, its execution fails, and R3 can be renewable (line 17, Algo2)−−>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:31:38.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="failed"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:45:38.000+01:00"/>
    </event>
    <!−− Select another eligible resource R3.1, since T3 is retriable, its execution fails, and R3 cannot be renewable(line19,Algo2) −−>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R3.1" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:48:33.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="done"/>
        <resource key="R3.1" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:58:38.000+01:00"/>
    </event>
    <!−− T4 −−> <event>
        <string key="activity:name" value="T4" property="compensatable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R4" property="RenewableANDExpandable"/>
        <date key="time:timestamp" value="2018−01−12T11:00:18.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T4" property="compensatable"/>
        <string key="lifecycle:status" value="done"/>
        <resource key="R4" property="RenewableANDExpandable"/>
        <date key="time:timestamp" value="2018−01−12T11:08:32.000+01:00"/>
    </event>
</trace>
```

Appendix 2

Listing 2: Business process log for experiment #2

```
  <trace>
    <!−− T1 −−> <event>
        <string key="concept:name" value="T1" property="Pivot"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R1" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T08:40:38.000+01:00"/>
    </event>
    <event>
        <string key="concept:name" value="T1" property="Pivot"/>
        <string key="lifecycle:transition" value="done"/>
        <resource key="R1" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T08:49:00.000+01:00"/>
    </event>
    <!−− T2 −−> <event>
        <string key="concept:name" value="T2" property="Pivot"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R2" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T08:55:40.000+01:00"/>
    </event>
    <event>
        <string key="concept:name" value="T2" property="Pivot"/>
        <string key="lifecycle:transition" value="done"/>
        <resource key="R2" property="Limited"/>
        <date key="time:timestamp" value="2018−11−12T09:50:36.000+01:00"/>
    </event>
    <!−− T3 −−> <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:53:02.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="failed"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T10:59:15.000+01:00"/>
    </event>
    <!−− Renew R3 since T2 is retriable, its execution fails, and R3 can be renewable (line 17, Algo2)−−>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T11:44:18.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T3" property="Retriable"/>
        <string key="lifecycle:status" value="done"/>
        <resource key="R3" property="Renewable"/>
        <date key="time:timestamp" value="2018−01−12T11:56:22.000+01:00"/>
    </event>
    <!−− T4 −−> <event>
        <string key="activity:name" value="T4" property="compensatable"/>
        <string key="lifecycle:status" value="activated"/>
        <resource key="R4" property="RenewableANDExpandable"/>
        <date key="time:timestamp" value="2018−01−12T11:59:08.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T4" property="compensatable"/>
        <string key="lifecycle:status" value="done"/>
        <resource key="R4" property="RenewableANDExpandable"/>
        <date key="time:timestamp" value="2018−01−12T12:08:04.000+01:00"/>
    </event>
    <event>
        <string key="activity:name" value="T4" property="compensatable"/>
        <string key="lifecycle:status" value="Compensated"/>
        <resource key="R4" property="RenewableANDExpandable"/>
        <date key="time:timestamp" value="2018−01−12T12:12:18.000+01:00"/>
    </event>
  </trace>
```

□