Microservices-based Approach for a Collaborative Business Process Management Cloud Platform

Diego Cocconi
Universidad Tecnológica Nacional (UTN) Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco (2400, Córdoba), Argentina
dcocconi@sanfrancisco.utn.edu.ar

Pablo Villarreal
Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI) / Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Universidad Tecnológica Nacional (UTN) Facultad Regional Santa Fe
Lavaisse 610, Santa Fe (3000, Santa Fe Province), Argentina
pvillarr@frsf.utn.edu.ar

***Abstract*—Nowadays, as a result of the adoption of new Internet technologies like cloud computing and containers, new software architectural styles like microservices, and emerging business models, organizations are able to establish collaborative networks for executing Collaborative Business Processes (CBPs) in a flexible way. Current approaches of Process-Aware Information Systems (PAISs) for implementing and executing CBPs have shortcomings, not only related to the services offered, but also about issues typical of the technological solution chosen, such as portability, elasticity, and privacy even when they are cloud-based. Portability refers to the dependency that is created due to the heterogeneity of the services offered by different cloud providers (generating a problem known as "vendor lock-in"), elasticity defines the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, and privacy refers to sensitive information about one person or a group that is expected to be hidden from others (e.g., identity, address, health, and hobbies). Then, the purpose of this work is to define an adequate approach for a cloud architecture of a CBP management platform facing these issues. To do so, starting from a definition of a cloud platform architecture that solves (almost all) shortcomings for CBP services offered, an enhancement making use of the microservices paradigm is proposed, overcoming the cloud difficulties previously identified.**

Keywords—business process, inter-organizational collaboration, process-aware information system, cloud computing, microservices, containers

I. Introduction

Nowadays, as a result of the adoption of new Internet technologies like cloud computing and emerging business models, organizations are able to establish *collaborative networks* for executing *Collaborative Business Processes* (*CBPs*) in a flexible way. A collaborative network [1] consists of autonomous, geographically distributed, and heterogeneous organizations that collaborate to achieve common goals [2]. Collaborative networks contribute significantly to enhance performance of Small and Medium Enterprises (SMEs) [3].

In a collaborative network the integration and collaboration among the organizations are established and carried out through CBPs [4]. A CBP (also called *process choreography* [5-6]) specifies the global view of interactions between roles that organizations perform to achieve a common business goal, and it serves as a contractual basis for the inter-organizational collaboration involved [7]. The implementation of collaborative networks (and inter-organizational collaborations) requires organizations can carry out the stages of the *Business Process Management* (*BPM*) lifecycle [5] to the agreed CBPs. These stages of the CBP management require to deal with: (1) organization autonomy; (2) decentralized execution; (3) global view of message exchange; (4) peer-to-peer interactions; and (5) adequate representation of communications [4].

Current solutions for CPBs based on Internet technologies like Web services require each organization to develop, implement, and maintain *Process-Aware Information Systems* (*PAISs*). A PAIS is necessary for each organization to execute its *Integration Business Processes* (*IBPs*) [7] whether using its own resources and infrastructure (hardware, software, network, etc.) or appealing to any *Business Process as a Service* (*BPaaS*) [25] cloud solution. An IBP (also known as *private process* [5] or *orchestration process* [6]) defines the public and private activities that an organization must perform to fulfill the message exchange agreed in a CBP. These solutions increase complexity and costs to the organizations, because they have to configure and implement a PAIS locally (or subscribe to a cloud service), and have to integrate all of them because they will need to interact with each other. Even though big companies can deploy this kind of solutions, the above aspects have a more negative impact on SMEs, governments of small cities and communities, and healthcare public or private institutions [8]. So it is important to use technologies that can make it feasible for organizations like these the application of CBP management in collaborative networks, enabling the access to easily integrable PAISs.

A first step to provide an adequate approach which allows organizations to generate, deploy, and enact PAISs on-demand, accordingly to the CBPs that the organizations agree to carry out, consists on exploiting the benefits of cloud computing technologies. As it was proposed in previous work [26], implementing a cloud solution based on an *Infrastructure as a Service* (*IaaS*) model (via a cloud provider like *Amazon*[1], or by means of a private implementation using a proper platform such as *OpenStack*[2] or *Apache CloudStack*[3]) could result in more agile collaborations, allowing to set up a collaboration at any moment and enacting more fluidly the involved processes, enabling to offer a *Platform as a Service*

[1] https://aws.amazon.com
[2] https://www.openstack.org/
[3] http://cloudstack.apache.org/

(*PaaS*) and/or a *Software as a Service* (*SaaS*) model to the organizations . This solution solves the shortcomings of: high costs and complexity of IT infrastructure required to implement PAISs, and rigidity of platforms for PAISs that do not enable organizations to generate, deploy, and enact PAISs on-demand, accordingly to the CBPs agreed in collaborative networks. However, there are still additional challenges: (1) there is an important integration work to be done between the collaborative system and third-party or legacy systems that perform the activities to be carried out in the IBPs of each organization; (2) fulfillment of the requirements of CBPs management is partially accomplished, e.g. global view of message exchange requires an advanced global monitoring service, which is difficult to achieve, even with an architecture in the cloud.

Besides, cloud solutions for CBPs have to deal with typical issues such as *portability*, *elasticity*, and *privacy*. Portability must be considered for conceiving a platform independent of the cloud provider; this should be contemplated for representing the different components in terms of an agnostic way. Elasticity is important for improving performance when several IBPs and PAISs are enacted, so the use of an appropriate elasticity controller, the definition of the correct metrics, and the selection of which components should be elastic is necessary. This leads to an appropriate degree of flexibility required to provide an appropriate QoS (Quality of Service) levels to the users [27]. Privacy issues are related to sensitive information shared in collaborations, or internal information managed in IBPs which handle sensitive data. However, cloud-based platforms for implementing and executing CBPs, such as the described in previous work [26] or others (Section 5), that are based on the deployment of the components in an IaaS provider (based on Virtual Machines —VMs) have more difficulties to achieve these requirements.

Microservices and containers allow going a step further. In one hand, micro-services can improve integration between the cloud platform and external systems (third-party or legacy systems) defining a microservice working as an interface between a process activity of an IBP and the third-party or legacy system that executes it; this allows that core microservices of the platform do work without taking care of the connection with the outside world. Only proper and well-designed interfaces and lightweight protocols that connect core microservices and microservices for the external systems are required. As microservices architectures are naturally distributed, they must deal with issues typical of this kind of systems. A series of architectural patterns allow to face these difficulties [19].

On the other hand, containers (and their orchestration systems, like *Kubernetes*[4] for *Docker*[5] containers) allow to deal with the problem of portability in cloud computing environments without the use of any standard, because containers are portable *per se*. Elasticity management becomes easier because at container level, the orchestration system implements an scaling mechanism, which is independent of the application; if microservices make use of containers for their implementation, they can be benefited by this approach, allowing scalable microservices. With each activity of the IBPs implemented as microservices, all business processes become scalable at activity granularity by nature. Identity services transversal to all the remaining services —such as the ones that offers OpenStack for all its cloud IaaS services, named *keystone*— can be applied at microservice level, ensuring the privacy required by each organization.

This work presents a new cloud-based platform for managing CBPs and proposes a microservice-based architecture for this platform to overcome the difficulties commented previously. Starting from a definition of the platform architecture proposed in previous work [26], an enhancement is provided by redefining the components in terms of microservices and adding new essential components to fulfill the requirements of the CBPs management. A concrete deployment of this architecture can be implemented chosen the proper technologies and cloud infrastructures already available for managing microservices architectures.

This work is organized as follows. Section 2 presents a background related to microservices, containers, and CBPs. Section 3 summarizes the new architecture of the CBPs management platform —which evolves from the one presented in [26]—, now based on microservices, and the main functionalities with a possible implementation. In Section 4 a case study which could be handled by the platform is described, together with the interaction between its main components. Section 5 gives a synopsis of related work about business processes in the cloud. Finally, Section 6 presents conclusions and future works.

II. Background

Cloud computing is a new paradigm for creating distributed systems based on the Internet [9]. From a business point of view, cloud computing could be seen as a model for delivering on-demand services, allowing organizations pay for resources or applications only when they use them ("*pay-per-use*") instead of facing the considerable costs of procurement and maintenance of a hardware and software infrastructure [10].

There are three *service models* for cloud computing: SaaS, PaaS, and IaaS, mentioned before [10]. In the SaaS model, applications are offered as services, accessed through the Internet on-demand by users. This model is mature today and there are several cloud applications available [8]. The PaaS model offers development services for building cloud applications. IaaS model implies the provisioning of hardware resources via *virtualization* [10]. Cloud computing also has different *deployment models*, such as *private cloud*, *community cloud*, *public cloud*, and *hybrid cloud*. In a private cloud, infrastructure is operated solely by one organization and managed by the organization or a third-party. In a community cloud several organizations jointly construct and share the same cloud infrastructure, which could be hosted by a third-party or by one of the organizations. In a public cloud, service provider has the full ownership of the cloud architecture with its own policy, value, profit, costing, and charging model. Finally, the hybrid cloud is a combination of private, community, or public clouds [11]. Additionally, the term *cloud federation* comprises services from different providers aggregated in a single pool, supporting three basic interoperability features: resource migration, resource redundancy and combination of complementary resources [12].

[4] https://kubernetes.io/

[5] https://www.docker.com/

Cloud computing serves as a basis for other models requiring a more complex orchestration, like containers and microservices frameworks, providing APIs for provisioning data computing, storage, and delivery capabilities [13].

A. Microservices and Containers

The microservices software architecture paradigm is gaining popularity as an approach towards a flexible execution and an independent deployment of service-oriented software systems [14]. This architectural style is characterized by a set of small independent services, running in their own processes and interacting through lightweight mechanisms, that together conform a single application [15].

Microservices architecture is often compared with the *Service Oriented architecture* (*SOA*). SOA is often associated with Web services protocols, tools, and formats such as *SOAP*, *WSDL* (*Web Services Description Language*), and the *WS-* family* of standards [16], whilst microservices require the use of a simple and lightweight communication protocol which enhances decoupling property between services, like *REST* over *HTTP*, which can rely on multiple formats (*XML*, *JSON*) [17].

The downside of implementing an application as a group of autonomous components like microservices is that they form a distributed system, which by their nature, are very hard to deal with [19]. To have a fully functional microservices architecture and to take advantage of all its benefits, the following components have to be utilized to handle the main problems [18-20]: (1) a *service discovery* component, because there exist several services and many of them might have a lot of instances, so it is necessary to keeping track of the deployed services; (2) a *configuration server* that decouples source code from its configuration, enabling the change the configuration of the application without redeploying the code; (3) a *load balancer* component, which in order to make an application scalable, should be able to distribute the load on an individual service among its many instances; (4) a *circuit breaker* component for providing fault tolerance against chain failures product of the dependency between services that are working together; (5) an *edge server*, implementation of the *API Gateway* pattern, for exposing external APIs to the public as an interface; (6) a *centralized logging service*, capable of detecting new microservice instances and collecting log events from them, and also interpreting and storing log events in a structured and searchable way; and (7) a *monitoring service* able to analyze hardware resource consumption per microservice, helping to discover the root cause of the problem when response times and/or the usage of hardware resources become unacceptably high, providing an early warning system of something going wrong.

Microservices can reside on VMs to enhance the scaling of their instances properly, but the small size of the services does not require a huge machine size to rely on. Besides, cloning, transferring, and enacting large images across the network will result in a complex activity to be performed, even in the cloud. The overhead is caused by the *hypervisor*, which is a computer software layer that needs to manage all the VMs and how they access the resources. These drawbacks raised the need for technologies such as Linux *containers* [17]. Containers provide a mean to virtualize an operating system so that multiple workloads can run on a single operating system instance, differently from VMs, that virtualize the hardware to run multiple operating system instances on it [14]. So, they provide a separate space for the processes without the need of a hypervisor to control the machines. On top of that, Docker platform contributed on building lightweight containers and handling them as well [17]. A Docker container wraps a piece of software in a complete filesystem that contains everything needed to run the software: code, runtime, system tools, system libraries and anything that can be installed on a server, guaranteeing that the software will always run the same, regardless of its hosting environment, providing code portability [14, 17].

Container orchestration technologies (for example, Kubernetes or Apache Mesos[6]) automate container allocation and management tasks, essentially abstracting away the underlying physical or virtual infrastructure [16]. A scalability feature of Kubernetes supporting several parameters allows an automation process that can be implemented according to the number of concurrent users accessing it. It is expected that the exploitation of scalability will improve performance and server response time to users without reducing server utility capabilities [21]. With microservice architectures relying on containers, those microservices can be dynamically replicated to cloud infrastructures that are under heavy load. It is not necessary to scale the complete system, as it would be required with a monolithic system [20], or to scale in terms of entire VMs.

B. Collaborative Business Processes (CBPs)

As it was commented before, to carry out collaborations, organizations integrate their business processes, agree on common business goals, coordinate their actions, and exchange information by defining and executing Collaborative Business Processes (CBPs) [22]. The implementation of collaborations implies that organizations can run through the stages of the *Business Process Management* (*BPM*) lifecycle [5] for the CBPs involved.

During the *analysis* and *design* stages of the BPM lifecycle, organizations must define not only CBPs but also their Internal Business Processes (IBPs), which model the private behavior of organizations and support the interactions and roles they perform in the collaboration. To deal with organizational autonomy issues, CBPs are defined as abstract processes. This means they are not directly executable by a centralized Process-Aware Information System (PAIS). Instead, they are enacted by means of the IBPs of each organization in a decentralized way, executed by their respective PAISs [7]. There are languages that emerged to support the modelling of CBPs, such as BPMN [6] and UP-ColBPIP [4, 23]. These languages enable the definition of technology-agnostic models for CBPs. The purpose is to describe CBPs at a high level of abstraction from a conceptual point of view, providing concepts closely related to the business and organizational domains, which are suitable for both business and technological people.

Implementation stage of the BPM lifecycle consists in the development, configuration, and deployment of PAISs required for each organization to execute their IBPs and to interoperate for managing CBPs. Inter-organizational collaborations rely on the capability of organizations to implement PAISs to enable the execution of their own IBPs

[6] http://mesos.apache.org/

and interact with the other PAISs to achieve the message exchange agreed in CBPs [7]. Finally, *execution* stage consists in the "enactment" of the CBPs by means of the real enactment of IBP instances by the PAISs of each organization, to execute the private activities organizations need to carry out, as well as the public activities related to the message exchange between them.

Thus, collaborative networks can be defined from two different perspectives: (1) considering a global picture of the control flow of interactions, and (2) considering a local picture of the public and/or private activities of each organization together with the interaction points among them. The correspondence between these perspectives and the types of processes involved are shown in Figure 1.

The *global perspective* describes the global and public behavior of a collaboration and the responsibilities of the participants [4]. This behavior is represented in CBPs as a single control flow that defines the order in which interactions (exchange of messages) between participants take place, with interactions defined as if they were seen by an observer who can only see the public and externally visible actions of each participant. This perspective can be defined and described by using the technology-independent languages mentioned before: BPMN (*choreography diagrams*) and UP-ColBPIP (*interaction protocols*).

The *local perspective* describes the behavior of a collaboration based on the activities of each participant. From this perspective, a collaboration is described as a composition of business processes that define the independent behavior of each participant and the interactions between them. Each participant has its own control flow and interaction points. Two types of business processes can be defined for each participant according to this perspective (Figure 1): (1) *interface processes* (public behavior) and (2) *integration processes* (public and private behavior). An interface process describes the public and externally visible actions of a participant in terms of the activities that support the receiving and sending of messages with each other [4-5, 24]. On the contrary, an integration process describes both the public and private behavior required to support the role a participant performs in the collaboration. The private behavior adds the internal activities and events required to support the internal business logic of a participant. BPMN can be used to represent both kind of processes of this perspective (through its public and private processes) in an independent way from the final implementation technology. WS-BPEL is an example of a specific-technology language that enables the definition of both kind of processes of this perspective but based on the Web services technology.

III. Microservices-based Platform for CBPs

In this Section the proposed platform for CBPs management is described. Then, first the offered services are summarized. Next, a conceptual architecture of the platform is presented, and all their components are described. Finally, implementation considerations are given.

A. Offered Services

The offered services to the final users (organizations) provided by the platform are based on those proposed in previous version [26], i.e. the support for the design, implementation, and execution stages of inter-organizational collaborations and CBPs lifecycles.

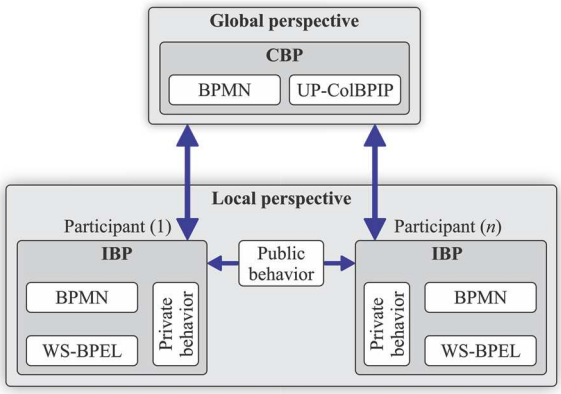

Fig. 1. Global and local perspectives of collaborative networks.

1) Services for the Design and Implementation of CBPs

Before starting to use the cloud provided services, organizations can register in the cloud platform and join collaborative networks. Then, they are ready to agree on new collaborations, which declares the CBPs to be executed. Once a CBP model is agreed by all organizations, the platform provides cloud services to generate the IBP models for each organization from the CBP model. Each IBP model contains the activities required to execute the collaboration from the point of view of the corresponding organization, but it is still an incomplete model that must be configured with execution details —private behavior, data, and resources necessary to carry out the activities, in order to achieve an executable implementation of the CBP. From the completed conceptual IBP models, there are services that allow the generation on-demand of the PAISs organizations need to implement and execute their IBPs. This implies the generation of an executable model of each IBP to be executed by the PAIS.

2) Services for the Execution and Monitoring of CBPs

Once all PAISs are generated and configured with their respective executable IBP models, organizations can make use of cloud services to support the execution of the CBPs through the distributed (autonomous) execution of the IBP instances supported by the respective instantiation of their assigned PAISs. With all PAISs and IBPs already instantiated and enacted, messaging and monitoring services are started. These services enable organizations to be aware of the global state of the collaboration.

B. Architecture of the Platform

The architecture describes the set of microservices that support all the cloud services explained in previous Subsection. A general view of the architecture is shown in Figure 2.

Due to organizations might not want to participate in collaborations by means of public cloud services that would expose or manage sensitive information, in the predecessor architecture —intended for deployment on an IaaS cloud service provider, it was proposed the use of private clouds for such organizations, interacting with a public cloud where the public cloud services reside. This enabled more autonomy to the organizations and attended the requirement that some of them would want to maintain and preserve the private information and activities of their IBP processes. The public cloud had components supporting all functionalities of the platform, including those that managed aspects related to the establishment of a collaboration as well as other functions that have to be necessarily global to all organizations; private

clouds, on the contrary, had only the components necessary to implement the processes of the owner organization, and a stub that provides the connection with the public cloud and its global services.

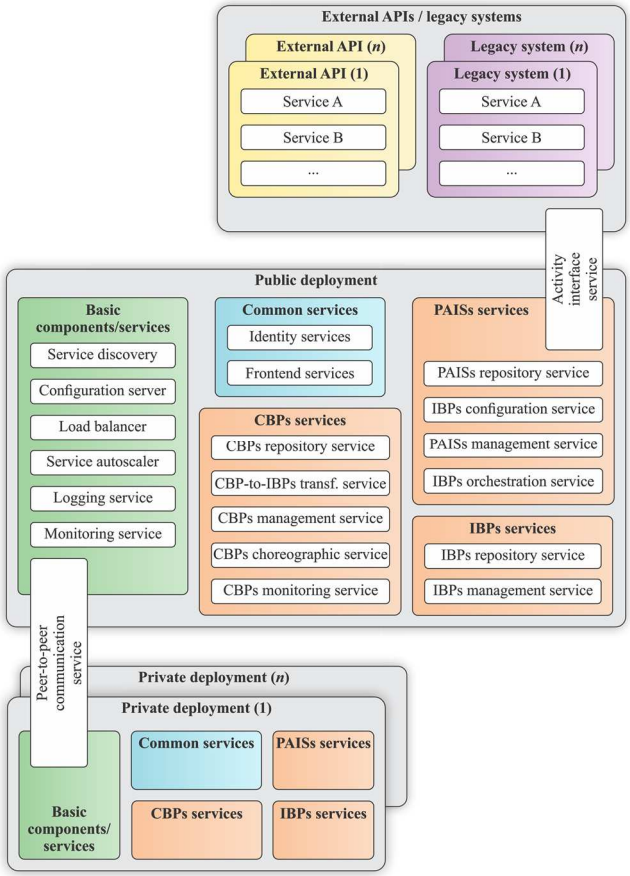

Fig. 2. Architecture of the cloud-based platform for CBP management showing the main groups of microservices.

Using microservices, the schema is similar: microservices can be deployed over different cloud providers —even in a private infrastructure. Anyway, services for accessing the collaborations and CBPs repository among others, which need to be global for all the organizations, requires to be deployed in a public cloud. All the services, both the global and those services (private services) required to be accessed in a private way for each organization can be deployed in a public cloud. But other configurations are allowed, in the sense that any organization can deployed their required private services in a private cloud (as it is shown in Figure 2).

Anyway, there must be a main public service as before, which will provide global services for accessing the collaborations and CBPs repository, among others, which need to be global for all the participants. In the private clouds, this kind of services are not necessary. The services deployed in the different clouds are communicated to each other via communication services. This means that the final implementation of the platform can be done upon a cloud federation or a hybrid cloud model.

The architecture of the platform is basically grouped in five categories of components/services: (1) *basic components/services*; (2) *common services*; (3) *CBPs services*; (4) *IBPs services*; and (5) *PAISs services*.

The group of *basic components/services* includes the typical components and/or services required to provide a fully functional microservices platform —i.e. well known components or services that allow solving the challenges that this kind of architectural paradigm implies; some of them are implementations of typical patterns of this paradigm [18-20].

- *Service discovery*: allows the registration of all microservices, as well as mapping between the required service to the endpoints of instances, via a URL or an IP address.
- *Configuration server*: provides configuration settings for each service independently of their implementation.
- *Load balancer*: distributes the load on an individual service among its many instances.
- *Service autoscaler*: automatically scales the number of service instances based on an observed metric, such us CPU utilization or another application-related metric.
- *Logging service*: collects log events from each individual microservice, and also interprets and stores those log events in a structured and searchable way.
- *Monitoring service*: collects data necessary to monitor failures of services when response times and/or the usage of hardware resources become unacceptably high.
- *Peer-to-peer communication service*: provides asynchronous and scalable communication between services of the platform deployed in different providers/private infrastructures.

Common services provide features widely used by the rest of the functionalities offered by the remaining microservices.

- *Identity services*: manage the identities of all the users and organizations that use the platform, ensuring that they can access all the public data (like information in CBPs repository) and preserve the data that is private of each organization, even if they make use of a private cloud.
- *Frontend services*: provide the interface to support user interactions and to delegate their requests to the corresponding microservices.

The remaining three groups of services provide the business core services offered by the platform. The *CBPs services* group services for accessing to the CBPs repository, establishing a collaboration, and monitoring the CBPs execution.

- *CBPs repository service*: manages access to the CBPs model repository, which stores collaborative networks, inter-organizational collaborations, and their CBP models. These services are only available from the public cloud and can be accessed by all organizations.
- *CBP-to-IBPs transformation service*: provides the operations that implements the MDA-based method proposed in previous work [22] to generate an IBP model of an organization from a CBP model. This method provides a model-to-model transformation process that takes as inputs a CBP model and a target organization role, and automatically generates as

output an IBP template model that contains the public and private activities and control flow logic that an organization has to implement for performing the target role in the CBP.

- *CBPs management service*: manages aspects related to the establishment of the collaboration. They have to negotiate the selection of the CBP model agreed by all the organizations in the collaborative network that want to participate in the collaboration. These services are also only available from the public cloud and can be accessed by all organizations.
- *CBPs choreographic service*: allows the coordination between the different independent IBP instances that are enacted to perform the real execution of the CBP, coordinating the sending/reception of asynchronous messages among those IBPs to carry out the collaboration.
- *CBPs monitoring service*: responsible for monitoring the current CBPs execution states.

IBPs services basically group services for accessing to the IBPs repositories and preparing the proper IBP models of each organization to be enacted.

- *IBPs repository service*: manages access to the IBPs model repositories, which store IBP models owned by each organization, together with templates that organizations can define and will be used for the transformation from CBPs to IBPs takes place.
- *IBPs management service*: manages the IBP models of each organization, saving/retrieving these models from the IBP repository and allowing to add the private activities that organizations require to complete the templates generated from the CBPs they participate in (in an agnostic way, i.e. ignoring the specification details of the PAISs that will execute the models).

Finally, *PAISs services* comprise services for accessing to the PAISs repositories, configuring and deploying the executable IBP models of each organization, orchestrating its execution, and providing the interfaces to interact with external services and/or previous existent legacy systems.

- *PAISs repository service*: stores the executable IBP models of each organization already configured, linked to all the required external services and/or legacy systems required for the execution.
- *IBPs configuration service*: generates the specifications of the executable IBP models from the fully complete IBPs in terms of the PAISs that will enact them, with all the public and private behavior that are required to fulfill the activities that each organization needs for their execution, together with all those activities involved in the CBP orchestration.
- *PAISs management service*: manages the PAISs of each organization on-demand, creating them, loading and deploying the executable IBP models retrieved from the PAISs repository, and acting as a dispatcher to the *IBPs orchestration service*, which will be in charge of the enactment.
- *IBPs orchestration service*: once a PAIS is enacted, this service manages the execution of the IBP, performing the orchestration between the activities and the services that need to be called for executing the IBP.
- *Activity interface service*: acts like an edge server that interacts with external programs or legacy systems.

Then, the platform architecture is flexible enough considering the services it must provide. The main interactions between components/services to perform these offered services will be detailed using a case study in Section 4. But flexibility must be considered about cloud computing and microservices aspects too. Elasticity must be provided for all services, so it is necessary to count with the *service autoscaler* and *load balancer* components of the basic group. Portability (and vendor lock-in in consequence) is another important issue, considering that the platform is conceived to be independent of the cloud provider or private cloud platform; taking advantage of the fact that the microservices architecture can be easily deployed by means of a container infrastructure, which helps the process of setting up the platform, portability is achieved without the need of any standard. About privacy, two approaches are offered to deal with sensitive information: (1) a transversal *identity service* that only allows access to the services and data that correspond to a particular organization and (2) the possibility that the organization that would like can deploy the platform on a private environment, communicated with the public cloud, but sharing only the necessary information to interact in the CBPs, preserving its own sensitive data and processes. In general, components/services of the *basic group* allow to deal with all the difficulties that a platform based on microservices will face.

C. Implementation of the Platform

This Section describes an implementation of the microservices platform. There are several approaches that can be followed for implementing it, trying to get a practical balance between cloud and microservices issues (portability, elasticity, privacy, and microservices requirements), as commented before.

The CBP platform can be implemented by means of an IaaS infrastructure for supporting the public cloud, where all the services will be available for the organization (Figure 2). Private clouds can be handled by the same or another IaaS provider, or even appealing to a private IaaS model. Then a proper deployment environment more microservices-friendly needs to be configured on those infrastructures, like Kubernetes as an orchestrator for Docker containers. Another way is to directly use a containers service, such as *Amazon Elastic Container Service*[7] (*Amazon ECS*), *Amazon Elastic Kubernetes Service* [8] (*Amazon EKS*), *Azure Kubernetes Service*[9] (*AKS*), or *Red Hat OpenShift Container Platform*[10]. The option chosen here is Amazon EKS for the public cloud services and the Red Hat OpenShift Container Platform for the private clouds, because Kubernetes counts with a proper scaling and load balancing mechanism, providing elasticity,

[7] https://aws.amazon.com/ecs/
[8] https://aws.amazon.com/eks
[9] https://azure.microsoft.com/en-us/services/kubernetes-service/
[10] https://www.openshift.com/products/container-platform

and also offers good implementations for the basic group of microservices.

Spring Boot[11] is used for the implementation of almost all the remaining microservices, integrated with other technologies when they are required. Spring Boot is an open source Java-based framework used for creating microservices. The *frontend services* require a concrete framework for giving the proper interaction with users. One advantage of using Spring Boot is the ability to easily set up web applications with a built-in embedded Apache Tomcat, so the frontend is developed using JavaServer Pages (JSP). The repositories (for CBPs, IBPs, and PAISs) require a database connection. Spring Boot is easily integrable to MySQL databases, the DB management system also used in a previous work where the repositories are fully described [7].

For implementing the *CBPs choreographic service* a technology allowing asynchronous sending/receiving of messages is required, because all IBPs services are autonomous and decentralized. The *Spring AMQP*[12] API is used for interacting with *RabbitMQ*[13] message broker to this end. The *IBPs orchestration service* also needs an additional technology for enacting an entire IBP executable model and perform the interactions with another IBPs (via messages using the *CBPs choreographic service*) and between the different activities and the external services and/or interactions with legacy system that they may require. Any process engine based on microservices would be suitable, but in this case, a workflow engine for microservices orchestration called *Zeebe*[14] is used. In workflows orchestrated by Zeebe, each task is usually carried out by a different microservice and send/receive tasks can interact with the *CBPs choreographic service*.

IV. Case Study

This section describes the functionality, applicability, and interaction between components/services of the proposed microservices-based platform for executing CBPs, by means of an implementation of a case study from a distribution network of electronic products, previously presented in [7]. This distribution network is a collaborative network consisting of: *Megatronic*, a *retailer* with points of sales around the center and east regions of Argentina; and *Philkaw*, *Grundrive*, and *Sanx*, which are assemblers of electronic products and *suppliers* of the *retailer*. *Suppliers* collaborate with the *retailer* in a separate and peer-to-peer way. Each *supplier* establishes an independent inter-organizational collaboration with the *retailer* to carry out a concrete CBP model. *Megatronic* agreed to carry out a *Vendor Managed Inventory* (*VMI*) [29] model with *Philkaw* and *Grundrive*, and a *Collaborative Planning, Forecasting, and Replenishment* (*CPFR*) [28] model with *Sanx*.

The deployment of the platform is as follow: *Megatronic* and *Philkaw* use a public cloud, and *Grundrive* and *Sanx* implement private clouds.

Now to use the platform, first, the *retailer* accesses to the *frontend services* of the public cloud (which make use of the *identity services* for authentication) and creates the collaborative network *Electronic Products Collaborative Distribution* using the *CBPs management service*, which is saved via the *CBPs repository service*. Then the *retailer* looks for *suppliers* in the platform and sent them an invitation to join this network. Afterwards, the *suppliers* join in the collaborative network created by the *retailer*. Then, the *retailer* creates three inter-organizational collaborations which refer to collaborations defined by the *retailer* with each *supplier* (all by means of the *CBPs management service*). Each collaboration indicates the business model adopted and the role the organizations fulfill. For each one of the created collaborations, organizations can manage the CBP models by using the *CBPs management service*, which interacts with the *CBPs repository service*. As an example, the *retailer* added several CBP models to the CBPs model repository as part of the CPFR-based collaboration with *Sanx*, for instance the *Collaborative Replenishment Plan Management* model (*CPFR process*). Figure 3 shows a choreography diagram in BPMN which defines the behavior of this CBP model.

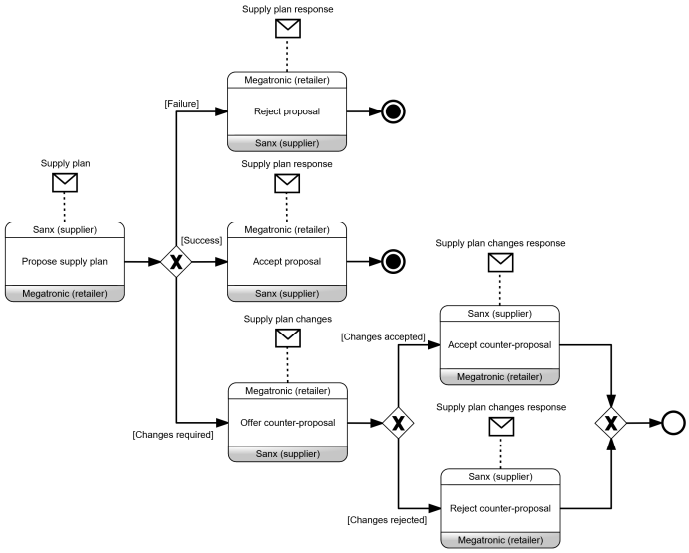

Fig. 3. Collaborative Replenishment Plan CBP model (adapted from [7]).

This model manages a simple negotiation process between the *retailer* and the *supplier* for agreeing on a replenishment plan of several products for a short-time period. It starts with the *supplier* that proposes a *supply plan* to the *retailer*, who evaluates it and decides to reject, accept or make a counter-proposal. The decision is sent to the *supplier*. In case of rejection or acceptance, the process finishes. In case of a counter-proposal, the *supplier* evaluates it and responds to the *retailer* with an acceptance or rejection. Having all organizations agreed the collaboration (and the CBP model), the collaboration status is automatically set to "*active*" and it can be accessed via the *CBPs monitoring service*.

What happens next is illustrated with the sequence diagram of Figure 4.a. The *CBPs management service* calls the *CBPs repository service* to get the final CBP model. Next, it calls the *CBP-to-IBPs transformation service* to generate the public behavior of the IBP for each organization (*Megatronic*, *Philkaw*, *Grundrive*, and *Sanx*) from the CBP as a template (if it does not exist previously in the correspondent IBPs repositories) and saves it in the IBPs repositories with the *IBPs repository service*. To perform its actions, the *CBP-to-IBPs*

[11] https://spring.io/projects/spring-boot
[12] https://spring.io/projects/spring-amqp
[13] https://www.rabbitmq.com
[14] https://zeebe.io/

transformation service employs the method and tool proposed in [24].

From the generated IBP template models, and by means of the *frontend services* and the *IBPs management service*, organizations complete their corresponding IBP models and add all the private activities they consider for carrying out the collaboration properly (Figure 4.b, *complete IBP model* interaction). This is still made in terms of a technology-independent language, such as BPMN. The completed IBP models are also saved to the IBPs repositories with the *IBPs repository service*.

Finally, organizations are now able to configure the complete IBP models to generate the executable specification. Again, with help of the *frontend services* and the *IBPs management service*, organizations retrieve their corresponding complete IBP models and generates the specification for the PAISs, with all the information and resource links necessary to interact with external services and/or legacy systems, using the *IBPs configuration service* (Figure 4.b, *generate IBP specification* interaction). The specifications of the IBP models are saved to the PAISs repositories by means of the *PAISs management service* and *PAISs repository service*.

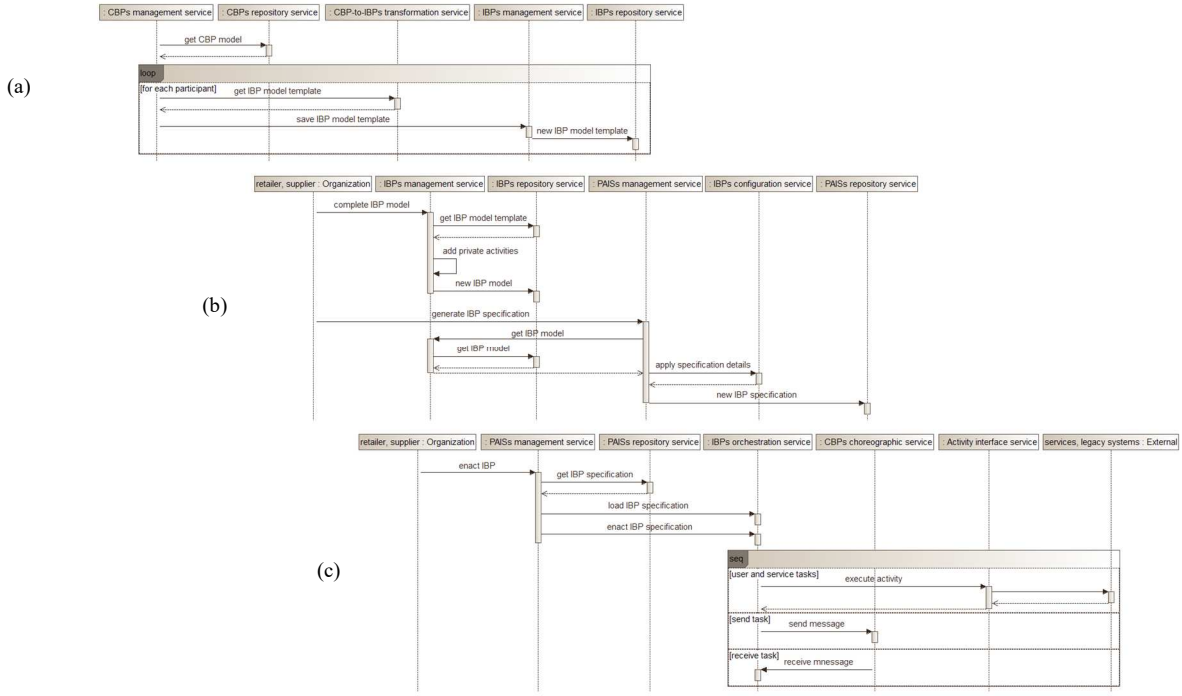

Fig. 4. Interaction between the components/services of the platform. (a) Generation of the IBP model templates for each organization from the CBP model. (b) Completing an IBP model template with the required private activities to get a complete IBP model, and from this completed IBP, getting an IBP specification. (c) Execution of the CBP by means of the decentralizes execution of the individual IBPs.

When the code of the PAIS is available, each organization is able to enact their IBPs and start the decentralized execution of the CBP (Figure 4.c). Again, this is done in the same way either in the public cloud and the private clouds. The execution implies accessing the PAISs repositories (via the *PAISs management service* and *PAISs repository service*) and retrieving the executable IBPs specifications. These specifications are made available to the *IBPs orchestration service*, which will enact them. During the execution of the IBPs, the *IBPs orchestration service* makes use of the *CBPs choreographic service* to interchange messages between IBPs, according to the CBP, and also uses the *activity interface service* to communicate each activity of the IBP models with the proper external services and/or legacy systems. The collaboration status is now set to "*in execution*".

Once in execution the IBPs (and the CBP, in consequence), each organization can follow the state of the collaboration using the *CBPs monitoring service*. This is particularly useful for the *retailer*, which can be aware of the progress of the collaboration established with each *provider*.

V. Related Work

Existing approaches for offering BPM in cloud environments (BPaaS) mainly focus on fulfilling elasticity issues [30-33]. Authors of [34] propose a PaaS model for cloud applications implemented in terms of active components to accomplish non-functional requisites. Other works focus on security and privacy in the cloud. In [35] authors propose an anonymization-based approach to preserve client business activity while sharing process fragments between organizations on the cloud. In [36] the author exposes that although outsourcing (the execution of business processes) harbors an economic potential, cloud consumers lose control over their data and executions, so they review the role of remote auditing as a mean to address this issue. Although these works provide interesting approaches for privacy and security, they do not focus on privacy and security in CBPs that are executed in a decentralized way in the same or different clouds. About portability, several works suggest the attachment to cloud standards definitions are a way to achieve portability. Most standards available depend of the cloud service model: *Topology and Orchestration Specification for Cloud Applications* (*TOSCA*) is an IaaS standard from OASIS [37-39]; *Open Cloud Computing Interface* (*OCCI*) is a RESTful protocol and provides APIs for all kinds of management tasks in cloud environments [40-42]; *Cloud Application Management for Platforms* (*CAMP*) is a

standard that addresses the problem of portability of artifacts and interoperability of APIs in PaaS environments [43-44].

There are few proposals that support cloud services for CBPs, such as [45-47]. In [46] authors present a SOA architecture based on an *Enterprise Service Bus* (*ESB*) for supporting collaborative processes. However, the architecture does not offer a clear vision of a cloud service model that could be used for supporting it. It is focused on offering interoperability (not portability) and agility not in the sense of dynamism for supporting CBPs on-demand. All of these proposals have still an important shortcoming for collaborative networks: they provide a centralized approach for CBPs being their execution driven by one organization, and do not deal with autonomous process execution.

In the microservices world, approaches for BPaaS include service orchestration, where a single executable process uses a flow description (such as WS-BPEL) to coordinate service interaction orchestrated from a centralized point. In [48], authors describe business process modeling integration with an automatic lightweight declarative approach for the workflow-centric orchestration of semantically-annotated microservices. Zeebe is a horizontally scalable and fault tolerant workflow engine developed by Camunda that uses BPMN to specify orchestration logic, simplifying collaboration within the teams in organizations [49]. Another microservices based architectural approach mainly for offering Industry 4.0 B2B and IoT (Internet of Things) support is the NIMBLE Collaborative Platform, which performs IoT-based real-time monitoring, optimization and negotiation in manufacturing supply chains [50], but it seems not suitable for general collaborations support.

Finally, outside the area of cloud computing, there are also software agent-based platforms proposed for executing CBPs [51-52]. In particular, in [51] a platform that deals with the issues of dynamic collaborations is proposed. However, these proposals require organizations to deploy PAISs in their own private infrastructures, which implies more complexity, costs, and poor agility for managing collaborations. This results in a more difficult adoption of these solutions by the organizations that are interested in the implementation of collaborative networks.

VI. Conclusions and Future Works

This work proposed a microservice-based architecture for a cloud-based CBP management platform. By using microservices and the deployment of them in orchestrated containers it provides a more suitable approach to deal with the issues of portability, elasticity, and privacy of CBPs executed in the cloud. In addition, the proposed architecture brings a more flexibility for the deployment of the microservices, since microservices that support the management of the private IBPs and their data can be deployed into a public cloud, or in a private cloud if an organization prefers it. This is enabled by the defined specific microservices that allow the communication between the global services deployed in the public cloud with the private services deployed in private clouds.

The proposed architecture fulfills the requirement of autonomy of the organization for executing their IBPs, along with the requirement of decentralized execution of CBPs. This is achieved by the services that support the execution of IBPS by means of the PAISs each organization implement into the platform, and the communication and synchronization of the IBPs by means of an asynchronous message-based microservices to support the decentralized execution of CBPs, as well as the monitoring of these process to provide a global view of the message exchange of CBPs.

In addition, the architecture provides the microservices that allow the on-demand generation of PAISs and IBPs that organizations require to be part of the execution of CBPs. This enhances the platform by reducing the effort curve required for the organization to engage in collaborations and execute CBPs. This also enables organizations can join in dynamic collaborative networks that do not have preset the CBPs that can be executed (i.e. implementations of CBPs and their corresponding IBPs must not be predefined in the platform), which allows the support to offer on-demand services to build collaborations in a dynamic and agile way.

A concrete deployment of this architecture with the proper technologies was given along with a case study illustrating the interactions between services. Next steps focus on fully conclude and validate the platform through the implementation of real cases from different domains such as supply chain, e-healthcare, or e-government.

References

[1] Chituc, C. M., Azevedo, A., & Toscano, C. (2009). "A framework proposal for seamless interoperability in a collaborative networked environment". Computers in industry, 60(5), 317-338.

[2] Camarinha-Matos, L. M., Afsarmanesh, H., Galeano, N., & Molina, A. (2009). "Collaborative networked organizations–Concepts and practice in manufacturing enterprises". Computers & Industrial Engineering, 57(1), 46-60.

[3] Andres, B., Macedo, P., Camarinha-Matos, L. M., & Poler, R. (2014, October). "Achieving coherence between strategies and value systems in collaborative networks". In Working Conference on Virtual Enterprises, 261-272, Springer Berlin Heidelberg.

[4] Villarreal, P. D., Salomone, E., & Chiotti, O. (2007). "Modeling and Specification of Collaborative Business Processes with a MDS Approach and a UML Profile". In Enterprise modeling and computing with UML, 13-44, IGI Global.

[5] Weske, M. (2012). "Business process management: concepts, languages, architectures" (2nd. Edition). Springer Publishing Company, Incorporated.

[6] Object Management Group, OMG. (2011). "Business Process Model and Notation version 2.0". Specification "formal/2011-01-03". Object Management Group. URL: http://www.omg.org/spec/BPMN/2.0/PDF/.

[7] Lazarte, I. M., Thom, L. H., Iochpe, C., Chiotti, O., & Villarreal, P. D. (2013). "A distributed repository for managing business process models in cross-organizational collaborations". Computers in Industry, 64(3), 252-267.

[8] Gupta, P., Seetharaman, A., & Raj, J. R. (2013). "The usage and adoption of cloud computing by small and medium businesses". International Journal of Information Management, 33(5), 861-874.

[9] Pallis, G. (2010). "Cloud computing: the new frontier of internet computing". IEEE internet computing, 14(5), 70-73.

[10] Lin, A., & Chen, N. C. (2012). "Cloud computing as an innovation: Perception, attitude, and adoption". International Journal of Information Management, 32(6), 533-540.

[11] Dillon, T., Wu, C., & Chang, E. (2010, April). "Cloud computing: issues and challenges". In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 27-33, IEEE.

[12] Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., & Kunze, M. (2011). "Cloud federation". Cloud Computing, 2011, 32-38.

[13] Esposito, C., Castiglione, A., & Choo, K. K. R. (2016). "Challenges in delivering software in the cloud as microservices". IEEE Cloud Computing, 3(5), 10-14.

[14] Bocciarelli, P., D'Ambrogio, A., Paglia, E., & Giglio, A. (2018, July). "A service-in-the-loop approach for business process simulation based

on microservices". In Proceedings of the 50th Computer Simulation Conference (p. 24), Society for Computer Simulation International.

[15] Lewis, J., & Fowler, M. (2014). "Microservices: a definition of this new architectural term" (2014). URL: http://martinfowler.com/articles/microservices.html.

[16] Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). "Microservices: The journey so far and challenges ahead". IEEE Software, 35(3), 24-35.

[17] Salah, T., Zemerly, M. J., Yeun, C. Y., Al-Qutayri, M., & Al-Hammadi, Y. (2016, December). "The evolution of distributed systems towards microservices architecture". In 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 318-325, IEEE.

[18] Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2015, September). "Migrating to cloud-native architectures using microservices: an experience report". In European Conference on Service-Oriented and Cloud Computing, 201-215, Springer, Cham.

[19] Larsson, M. (2019). "Hands-on microservices with Spring Boot and Spring Cloud: build and deploy Java microservices using Spring Cloud, Istio, and Kubernetes", Packt Publishing.

[20] Hasselbring, W., & Steinacker, G. (2017, April). "Microservice architectures for scalability, agility and reliability in e-commerce". In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 243-246, IEEE.

[21] Dewi, L. P., Noertjahyana, A., Palit, H. N., & Yedutun, K. (2019, December). "Server Scalability Using Kubernetes". In 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), 1-4, IEEE.

[22] Villarreal, P. D., Salomone, E., & Chiotti, O. (2006). "A MDA-based development process for collaborative business processes". Milestones, Models and Mappings for Model-Driven Architecture, 17.

[23] Villarreal, P. D., Lazarte, I. M., Roa, J., & Chiotti, O. (2009). "A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language". In Business Process Management Workshops, 43, 318-329.

[24] Lazarte, I. M., Tello-Leal, E., Roa, J., Chiotti, O., & Villarreal, P. D. (2010, October). "Model-driven development methodology for B2B collaborations". In Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International, 69-78, IEEE.

[25] del–Río–Ortega, A., Gutiérrez, A. M., Durán, A., Resinas, M., & Ruiz–Cortés, A. (2015, June). "Modelling service level agreements for business process outsourcing services". In International Conference on Advanced Information Systems Engineering, 485-500, Springer, Cham.

[26] Cocconi, D., Roa, J., Villarreal, P. (2018). "Collaborative Business Process Management Through a Platform Based on Cloud Computing". CLEI Electronic Journal, 21(2). DOI: https://doi.org/10.19153/cleiej.21.2.6.

[27] Cocconi, D., Roa, J., Villarreal, P. (2019). "eBPSim: A Simulation Tool for Testing Elasticity Strategies in Cloud-based Business Process Solutions". In proceedings of the XXII Iberoamerican Conference on Software Engineering, CIbSE 2019, La Habana, Cuba, 377-390.

[28] Min, H., & Yu, W. B. V. (2008). "Collaborative planning, forecasting and replenishment: demand planning in supply chain management". International Journal of Information Technology and Management, 7(1), 4-20.

[29] Franke, P. D. (2010). "Vendor-managed inventory for high value parts: results from a survey among leading international manufacturing firms", 3, Univerlagtuberlin.

[30] Mohamed, M., Amziani, M., Belaïd, D., Tata, S., & Melliti, T. (2015). "An autonomic approach to manage elasticity of business processes in the Cloud". Future Generation Computer Systems, 50, 49-61.

[31] Han, Y. B., Sun, J. Y., Wang, G. L., & Li, H. F. (2010). "A cloud-based bpm architecture with user-end distribution of non-compute-intensive activities and sensitive data". Journal of Computer Science and Technology, 25(6), 1157-1167.

[32] Schulte, S., Janiesch, C., Venugopal, S., Weber, I., & Hoenisch, P. (2015). "Elastic business process management: state of the art and open challenges for BPM in the cloud". Future Generation Computer Systems, 46, 36-50.

[33] Woitsch, R., & Utz, W. (2015, October). "Business process as a service: Model based business and IT cloud alignment as a cloud offering". In Enterprise Systems (ES), 2015 International Conference on (pp. 121-130). IEEE.

[34] Pokahr, A., & Braubach, L. (2015). "Elastic component-based applications in PaaS clouds". Concurrency and Computation: Practice and Experience.

[35] Bentounsi, M., Benbernou, S., & Atallah, M. J. (2012, June). "Privacy-preserving business process outsourcing". In Web Services (ICWS), 2012 IEEE 19th International Conference on, 662-663, IEEE.

[36] Accorsi, R. (2011, July). "Business process as a service: Chances for remote auditing". In Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, 398-403, IEEE.

[37] Binz, T., Breitenbücher, U., Kopp, O., & Leymann, F. (2014). "TOSCA: portable automated deployment and management of cloud applications". In Advanced Web Services, 527-549, Springer, New York, NY.

[38] Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., & Wagner, S. (2013, December). "OpenTOSCA–a runtime for TOSCA-based cloud applications". In International Conference on Service-Oriented Computing, 692-695, Springer, Berlin, Heidelberg.

[39] Carrasco, J., Cubo, J., Durán, F., & Pimentel, E. (2016, June). "Bidimensional cross-cloud management with TOSCA and Brooklyn". In Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on, 951-955, IEEE.

[40] Ahmed-Nacer, M., Gaaloul, W., & Tata, S. (2017, June). OCCI-Compliant Cloud Configuration Simulation. In Edge Computing (EDGE), 2017 IEEE International Conference on, 73-81, IEEE.

[41] Metsch, T., Edmonds, A., Nyren, R., & Papaspyrou, A. (2010, November). "Open cloud computing interface–core". In Open Grid Forum, OCCI-WG, Specification Document. URL: http://forge.gridforum.org/sf/go/doc16161.

[42] Parák, B., Šustr, Z., Feldhaus, F., Kasprzakc, P., & Srbac, M. (2014, March). "The rOCCI Project–Providing Cloud Interoperability with OCCI 1.1". In International Symposium on Grids and Clouds (ISGC), 23(28).

[43] Karmarkar, A. (2014, October). "CAMP: a standard for managing applications on a PaaS cloud". In Proceedings of the 2014 Workshop on Eclipse Technology eXchange, 1-2, ACM.

[44] "Cloud Application Management for Platforms". Version 1.1. Committee Specification 01. URL: docs.oasisopen.org/camp/camp-spec/v1.1/camp-spec-v1.1.html.

[45] Camarinha-Matos, L. M., Juan-Verdejo, A., Alexakis, S., Bär, H., & Surajbali, B. (2015, February). "Cloud-based collaboration spaces for enterprise networks". In Computing and Communications Technologies (ICCCT), 2015 International Conference on, 185-190, IEEE.

[46] Benaben, F., Mu, W., Boissel-Dallier, N., Barthe-Delanoe, A. M., Zribi, S., & Pingaud, H. (2015). "Supporting interoperability of collaborative networks through engineering of a service-based Mediation Information System (MISE 2.0)". Enterprise Information Systems, 9(5-6), 556-582.

[47] Sun, Y., Su, J., & Yang, J. (2014, September). "Separating execution and data management: A key to business-process-as-a-service (BPaaS)". In International Conference on Business Process Management, 374-382, Springer, Cham.

[48] Oberhauser, R., & Stigler, S. (2017). "Microflows: enabling agile business process modeling to orchestrate semantically-annotated microservices". In Proceedings of the Seventh International Symposium on Business Modeling and Software Design (BMSD 2017), 19-28.

[49] Hamidehkhan, P. (2019). "Analysis and evaluation of composition languages and orchestration engines for microservices" (Master's thesis).

[50] Innerbichler, J., Gonul, S., Damjanovic-Behrendt, V., Mandler, B., & Strohmeier, F. (2017, June). "Nimble collaborative platform: Microservice architectural approach to federated iot". In 2017 Global Internet of Things Summit (GIoTS), 1-6, IEEE.

[51] Tello-Leal, E., Chiotti, O., & Villarreal, P. D. (2014). "Software agent architecture for managing inter-organizational collaborations". Journal of applied research and technology, 12(3), 514-526.

[52] Küster, T., Lützenberger, M., Heßler, A., & Hirsch, B. (2012). "Integrating process modelling into multi-agent system engineering". Multiagent and Grid Systems, 8(1), 105-124.