Constructing and Monitoring Processes in BPM using Hybrid Architectures

José Martinez Garro
Universidad Nacional de La Plata
UNLP
La Plata, Argentina

Patricia Bazán
LINTI – Universidad Nacional de La Plata
UNLP
La Plata, Argentina

***Abstract*— With the entrance of BPM in the Cloud, a change in the conception and design of Business Processes has been produced. Distributed environments, in this context offer computing possibilities which are advantageous for processes, especially in a decomposition context. This last concept has been introduced in BPM allowing processes to be executed in a cloud environment as well as in an embedded one. This situation takes advantage of both approaches under criteria like sensitive data, high computing performance and system portability. An unexplored aspect in current bibliography is process monitoring over a decomposed environment. In the present article we introduce the analysis of some concepts presented in current bibliography, and we propose also the architecture for a distributed process monitoring system. In this architecture we consider different design factors like location transparency, and the data needed for instance tracking over a cloud system.**

Keywords — BPM; Cloud Computing; Execution; Monitoring.

I. Introduction

In this article we face the problem of including a Business Process Management System (BPMS) in a cloud oriented collaborative environment, with the particularity that it is an external environment to the organization. It is one of the purposes of this work to make a current bibliography analysis in sections II to VII, where we describe the different variants of a cloud model, its benefits and cons, hybrid architectures with embedded systems and the problem of monitoring a distributed process. Then, from section VIII we introduce the architecture of a process monitoring application. Finalizing the document we present some conclusions about the current state of the art and future work proposals in this research line.

II. Related Works

There are different trends in what comes to BPM in the cloud, but they are different if we are talking about research fields or trends in the market. Currently we can find research works tending to analyze the different paradigms of BPM (whether in the cloud or embedded), and how they escalate according to user's needs, connectivity that grows and mobile device incorporation. In [1], [2] and [3] especially we found trends like adaptive workflows and complex events.

These references support the idea of the hybrid architecture and the necessity of monitoring a distributed process using a centralized application. Regarding the other references, we will cite each one of them in every related topic.

In relation to the commercial market, we find fewer advances than in the research area. Most of the available BPMS in the cloud are very similar to the embedded ones, and the concepts introduced in the present work, in [2] and [29] like decomposed processes (or dynamic services) are not present. At the same time, most BPMS support local process monitoring, which is not equivalent to monitor a process instance distributed in different servers. In this paper we introduce further our approach for a monitoring application that gathers information from different servers in a complex architecture and displays it seamlessly.

III. BPM and Cloud

With the fast technological development in the context of application launching and execution using cloud based architectures, companies that began to choose this model are facing new problems. In particular, collaborative business processes with several interaction areas offer an optimization potential through the combination of cloud computing and BPM. A common factor between both paradigms is the flexible and agile approach. The cloud based computing model may be considered as an enabler for an improved combination of service oriented architectures, and also an agile procedure for Business Process Management. But this potential depends on the conditions imposed by the different frameworks, which can be viewed from technical and financial aspects.

A. *Technical view:*

From a technical point of view there are three dimensions in order to design, implement and successfully operate the different BPM tools in a cloud environment. These dimensions are: programming, integration and security.

- Programming: complex and distributed systems are easily reachable in current IT. In connection with obtaining more usability and flexibility, this complexity represents new requirements for Software Engineering. To solve this problem it is necessary to adopt new languages. So, based on new concepts and innovative techniques, the efforts invested in the development phase have been reduced to convert the complexity of these new aspects into a manageable element.
- Integration: this category can be divided in data integration, function integration and process integration. Under the light of the new challenges involved, the current topic plays an important role

in different scenarios. For example, a cloud based workflow can control distributed activities beyond the companies' border, mainly due to its easy accessibility. For a simplified execution of several process instances it is necessary to have integration interfaces and structured methods that allow joining the new components under the considered process.

- Security: this concept can be divided into three categories: functional security, information security and data security. All these categories have a significant relevance for BPM, especially in regard to business process grids and distributed process servers. Functional security specifies how the current status corresponds with the desired functionality status. The information security is focused in unauthorized changes or information extractions, as well as data security is in charge of the process related data.

Even more, from a technical point of view the question on "what processes are more appropriated to be executed in a cloud-based architecture" should be responded. Possible risks, such as insufficient integration options, location and integrity problems as well as programming interfaces should be taken in consideration.

B. *Financial view*

There are two dimensions from the financial point of view:

- Availability: the services provided by a cloud infrastructure can be accessed at any time because of the high availability model. Based in a high abstraction level, the customization and installation are significantly easier. In addition with this simplification, the final user is capable of working with the service immediately.
- Investment risk: in the context of the different variable billing models (for example "pay per transaction") the use of a cloud based service results in certain charges. These charges contain relevant costs given by transferences and transactions [1] [2] [3] [4].

IV. BENEFITS AND DRAWBACKS

Cloud based BPM provides users the possibility of using software in a "pay per use" way, instead of forcing them to make big investments in BPM software, hardware and maintenance, versus the traditional licensing applications. Systems can escalate up and down according to the user's needs. This means they do not have to worry about the over/ under resource provisioning because of the high adaptability provided currently by cloud service providers, as we can see in Figure 1.

The current model, on the other hand, has several low points. By putting a BPMS in the cloud, users may lose control over sensitive data. This aspect results major considering that business processes inside an organization may manage important information for it and its members. On the other hand, the non-high computational activities' efficiency and effectiveness cannot be increased by putting them in the cloud, but rather these activities may get more expensive. For example, an activity which is not intensively computational could need to process a certain amount of data. The transference of these data to the cloud could take more time than the transmission to an embedded version installed locally. That transference could result bigger than the real necessity of processing. Even more, the cost of the activity may increase due the data transference. This element is one of the billing concepts in a cloud computing system because of the high connection availability [1] [2] [5] [6].

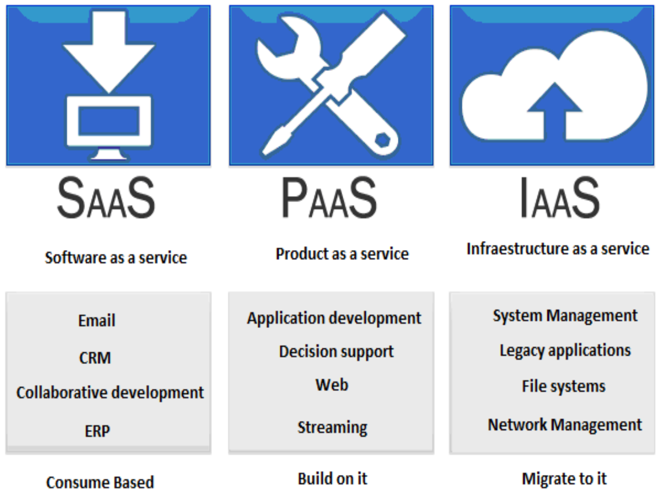


Fig.1. Service model

V. SERVICE MODEL

A. *Infrastructure as a service (IaaS)*

When an application is moved to an IaaS model, the cloud user is responsible for the operating system, the middleware and the applications running on the virtual machine. The action of installing BPM software in an IaaS cloud solution is comparable to installing an embedded BPMS, since everything except the hardware is managed by the cloud user. Furthermore, the user has to make some security decisions in order to avoid intrusions. According to this, possible security measures are: port blocking, access control policies and updating the applications and the operating system frequently.

B. *Product as a Service (PaaS)*

By positioning a workflow based application in a model like "Product as a Service", the responsibilities for the user and the cloud provider are different. The execution engine is assumed as a part of the platform, so it is offered by the service provider. Users must upload their processes to run them in the cloud. The engine can be used by several users since the platform is shared. The responsibility for data storage and management is no longer in charge of the user, who has to deal with several security issues:

- The process models should not be readable by intruders in posession of a description file.
- Process models should not be altered by intruders.
- Process models should not be deployed in other servers.

In order to achieve these requirements, the process model descriptions should be encrypted and signed. The encryption ensures that process models are not readable by intruders. By the action of signing them, it can be assured that a file is only valid for a particular execution engine, and using it to point to another execution engine will provoke an error. This turns into utility considering that the same server can be accessed by different users in a shared environment.

Storing the application database can be an issue also. Data should be encrypted in order to not be readable by intruders. Data encryption in a relational database generates expressivity issues with queries using relational operators. For example, joins can have problems in an encrypted data context.

C. Software as a Service (SaaS)

By moving an application to a SaaS model, the cloud provider is now responsible for the application itself. The application is no longer an asset of the enterprise cloud user but it is offered by the cloud provider. The application may be given to multiple cloud users in a single or multiple tenant architecture. In a single tenant paradigm, an execution engine is installed for each process model. In a multi-tenant environment, multiple users and process models are served by a unique engine. The data stored by the cloud provider should be assured in order to prevent unintended accesses, both by the service provider or other users in the cloud. The same measures we have mentioned in the previous subsection related with signing and encryption can be applied to solve this problem.

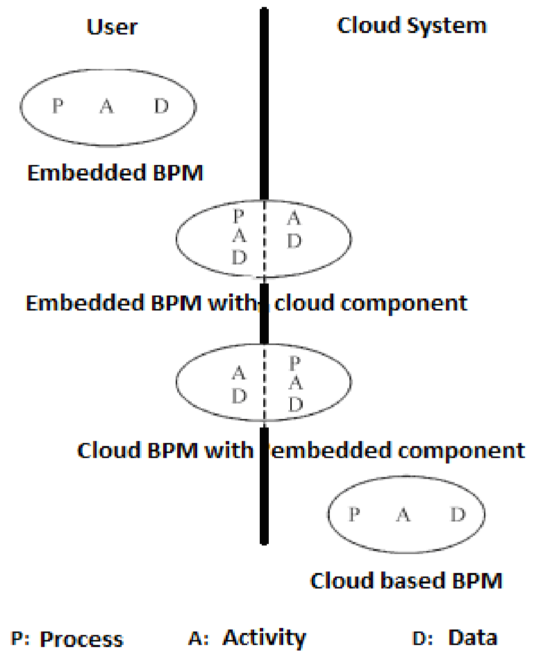
In a multi-tenant architecture, different users access the same execution engine. The data used by one user should not be accessible to other cloud users. There are two possible solutions for this problem: in the first place, a database for each cloud user can be created. As an alternative, a column to each table where the user identifier is saved can be added. It is necessary to observe the scalability of both solutions: the amount of users could increase, and because of that, the need of resources too [1] [6] [7] [8].

VI. COMBINATION OF EMBEDDED AND CLOUD SCHEMES

Privacy protection is one of the barriers to execute BPM in a cloud environment. Not all users desire to put their sensitive data outside the organization. Besides, it is necessary to observe product's portability and versions, and their availability in a cloud system. Another not minor problem is the efficiency.

The intensive computing activities may obtain benefits in the cloud due to the scalability and the computing force high availability. The non intensive computational tasks, on the other hand, not always take advantage of this context. The performance of one activity running in an embedded environment should be better than in the cloud because of the data that are transferred in order to execute the activity. These activities could also result expensive due to the fact that data transference is a billing criterion in the cloud [11] [12].

- Architecture: in most BPM solutions, the process engine, the activities and the process data are located in the same side, even in an embedded or cloud solution. There are some papers introducing the PAD model (Process - Activity - Data) of Figure 2 as a distribution possibility for BPM in the cloud. In this approach, the process model, the involved activities and the data are separately distributed. The PAD model defines four possibilities of distribution:



-

Fig.2. PAD Distribution Schema [6]

1) The first pattern is the traditional alternative where all elements are distributed over the final user side.

2) The second pattern is useful when the user already has a BPMS, but the high computing activities are located in the cloud to increment their performance.

3) The third pattern is useful for the users who still do not have a BPMS, so they can use the cloud system in a "pay per use" way. In this approach the activities with low computing intensity or the ones with sensitive data management can be located on the final user side.

4) The fourth pattern is the cloud based model where all the elements are located in the cloud.

- Business processes consist of two kinds of flows: control and data. Control flows regulate the execution of activities and their sequence, while data flows determine how the information is transferred from one activity to the other inside the process. BPM engines must deal with the control of both kinds of flows. A data flow could contain sensitive data, so when a BPMS is deployed in the cloud, the content of those flows should be protected. An example of the proposed architecture could be a scenario where the engine in the cloud only deals with data flows using reference identifiers instead of real data. When an activity needs sensitive data, the data being transferred to the activity are managed under user supervision in an encryption tunnel. Sensitive data are saved in the final user side, and non sensitive data are saved in the cloud. This schema allows that sensitive data do not travel indiscriminately through the web.

- Optimal distribution: the cloud system costs have been an object of study in different articles. There are several formulas to calculate the optimal distribution of activities, since they can be located in the cloud or in an embedded system. The calculation takes in consideration time costs, monetary costs and privacy risk costs. By using the formulas, users can make cost estimations about deploying part of their applications in an embedded or cloud system alternatively [2] [5] [6] [9].

VII. Process Decomposition

It is possible to generalize the distribution and identify a fifth pattern where the process engine, the activities and data are deployed in the cloud and in the final user. This solution presents two potential benefits:

1) The process engine regulates control and data flows. One activity receives data from the process engine and after its execution the produced data are passed again to the process engine. Consider now a sequence of activities located in the cloud, while the process engine is deployed in the final user. Each activity uses data produced by the previous activity as an income. Data are not passed directly from one activity to the other but they are sent to the process engine first. Since data transference is one of the billing factors in this model this kind of situations could become more expensive when large amounts of data are transmitted between activities. To avoid this problem a process engine can be added to the cloud, in order to regulate the control and data flows between activities located inside it. When a sequence of activities is located in the cloud, data are regulated by the process engine in the cloud. This reduces the amount of data to be transmitted between the cloud and the embedded system.

2) When the cloud is not accessible, users can execute business processes in a complete way in the embedded system until the former one is available again.

In order to run a single business process between two separated engines, it should be split into two individual processes. It could be convenient for the users to take a distribution list of the process and its activities. The process can be automatically transformed into two business processes, one in the cloud and the other in the embedded system. The communication between both systems can be described using a choreography language, like BPEL. Besides, the distribution list can be created automatically according to the optimal distribution formulas mentioned in subsection VI [13] [14].

Business process monitoring is more complicated now, since the process has been divided into two or more parts. As a solution, a monitoring tool can be developed for the original process, through the combination of the individual process monitoring details. This point will be analyzed further.

A possible approach to manage the process decomposition is to identify its structure and semantics. When the control and data dependencies are identified, the consequences of moving some activities from the embedded system to the cloud and vice versa can be researched. When the activity distribution consequences are known, a transformation model can be created.

Then, a business process and a list with marks are used to create two separated processes, one for the cloud and another for the final user. Also, a choreography description can be generated in order to describe the communication between both processes using some standard language, like BPEL [6] [10].

VIII. Hybrid SchemEs Implementation

The possibility of locating a BPMS in an external space to the organization (for example in a cloud computing architecture with a SaaS model) makes feasible to access it from inside the organization through an Internet connection, as well as from any other external point. Considering this fact, besides the possibility of having clients accessing from mobile devices, the access points to the cloud are incremented.

This generates the following issues about process execution, and their corresponding proposed solutions:

- **Process Decomposition:** as exposed in Section V, the fact of putting a BPM server in the cloud generates the problem of what to do with sensitive data management. Facing this problem, this solution can be enounced: in case of publishing the corporative database (or at least part of it) in a cloud environment is not a viable choice according to the organizational security policies, the decomposition of the process is going to be necessary in order to implement a hybrid scheme. In this scenario, the high computing activities can be located inside the cloud in order to take advantage of the computing performance, and the activities that make use of corporative sensitive data are located inside the organization in an embedded installation.

- **Decomposed process synchronization:** the disaggregated process is formally divided into sections according to the amount of involved servers. According to this, it is going to be necessary to solve how to synchronize the servers in order to ensure the execution sequence. There are, in theory, different ways to implement the synchronization, such as by using messages or event monitoring. Using messages, the end event of each process part invokes the start event of the next one. This can be made through start and end message type events, included both in the last version of standard BPMN (Business Process Management Notation), where the execution of the end event of a process throws a message to the BPMS in order to notify the finalization, and require the execution of a process previously parameterized. The notifications can be implemented by using a message queue and a daemon for pooling. This daemon receives messages and initiates instances of the required process. In this way, each server in this hybrid model (the embedded and also the cloud based ones) must have a copy of the pooling service in order to receive the finalization notifications and later notify the process engine. The result of this is to initiate instances corresponding to the requested definition [15] [16].

- **Decomposed process monitoring:** the biggest problem of having a process partitioned orientation is to monitor the different distributed instances, and at the same time to accomplish an integrated model of them under the optic of the "real process" which they belong to. In order to solve this inconvenient the following solution can be analyzed: in first place it is necessary to associate the different instances with the original process, in order to recover them from the existing servers. Once they are recovered, some kind of application in charge of gathering data and showing them seamlessly should be provided. The most important thing in this aspect is to accomplish monitoring transparency for the user, without forcing him to distinguish the server where each activity has been executed. This fact provides thus an integrated visualization of the different instances by seeing them as a unique entity. The implementation of the current feature should be made by a cloud resident web application, located there in order to access every involved server, whether cloud or embedded, and to ensure user access from any point. For this purpose it is important for the application to have a catalog with every existing server in the architecture, with their location information updated. Each involved server will have a copy of a web service which receives a process definition identifier and returns information about every existing instance associated with the sent definition. The returned information includes instance identification, current status (running, completed, suspended), current activity in case of non-completion status, start and end date. According to this, the cloud resident web application sends an invocation of the web service with the selected process definition as a parameter to each server, and receives the information of the associated instances. Then this information will be visualized in a web interface where the user can select a particular instance and observe its details. For this purpose the application contains a web service to require to each server the details of the associated activities. The information returned includes identification of the activity, associated participant, start date, current status and end date. After receiving this information the web application will allow the user to observe some activity details transparently, without indicating the server information where they were executed. This helps to accomplish location transparency [6] [17] [18].

IX. MONITORING PROCESSES IN THE CLOUD

As we have seen previously, the biggest problem about using a partitioned process model is to gather and monitor the different distributed instances (either in an embedded system or in the cloud), and at the same time to accomplish an integrated view under the optic of the "original process" which they belong to. To face this inconvenient we have designed a solution considering distributed and intercommunicated components forming an architecture, which is described as follows.

On the one hand, it is going to be necessary to associate the different process instances initiated in a chain, with the purpose of gathering information about them accessing the different involved servers. The execution model of decomposed processes consists of linking each instance flow to the corresponding partitioned processes. Thus, when an instance finishes in a server, it initiates automatically a new instance corresponding to the next process partition, depending on the distribution architecture. For this purpose, each node in the architecture should be capable of establish communication with the next node in order to initiate new instances, and gather in this way information about them. Namely, given a new instance which was initiated in a node of the architecture, we should be able to obtain, not only its data but every instance generated by it in another server [29].

A. Bonita Open Solution: API and connectors.

There are several ways of implementing instance flow linking. In our case we have selected Bonita Open Solution [30] as the BPMS. In this way, once the original process was partitioned over the servers, following criteria like sensitive data storing, data transferring and application portability, we have used the API and connectors provided by the BPMS in order to create instances and recover their information using Java classes. These classes use the API as libraries, including functions like server authentication, instance launching, instance information gathering and process variable setting. These classes are invoked from the process definition using connectors.

It was also included in each process definition the information needed for the communication with another Bonita server inside the architecture, and in this case, by using connectors, launch new instances in that server. Thus, every instance when is finished will execute the connector which allows initiating a new instance by using the API, linking in this way automatically the process execution flow [19] [20].

B. Centralized front-end

As it was described initially in section VIII, a monitoring application must be developed in order to show integrated data related with distributed instances. Facing the execution link, it is very important for each instance to be able of storing, not only their own information but the one associated with the instances created by them over other servers. In this way, by accessing the initial instance of the process, it is possible to recover the information associated to the next instance, and so on in order to obtain the complete flow of the process. Once recovered the execution chain in the different servers, it must be provided an application for visualization in charge of gathering data and show them seamlessly.

The most important thing in this aspect is to accomplish monitoring transparency for the user: he should not be forced to distinguish the server where the activity was executed but he should visualize seamlessly the different instances and observe them as a unique entity. The implementation of this feature was made through a web application located in the cloud, following the criteria established in Section VII. This application was placed there in order to access each involved server, being them cloud or embedded, guaranteeing in this way user access from every point. For this purpose it is important for the application to have a catalog with the existing servers in the architecture considering their location information updated. Each of these servers has a copy of a web service (*getInstanceService*), which receives a process definition id and returns information of each instance existing in the server associated with the definition sent as a parameter. The information returned includes instance id, current status (executing, completed, suspended), current activity if the instance is not finalized, start and end date. In this way, the application located in the cloud sends to each server a web service invocation with the selected process definition as a parameter, and receives the information of the associated instances. Then, this information is visualized in a web interface, where the user can select a particular instance and observe its details. In order to make this, the application has another web service (*getInstanceActivityService*) used to get from each server the details of each activity associated to the instance. The returned information includes activity id, participant, start date, current status and end date. Once ended this collection phase, we need to remember that each instance contains also the information of the different instances initiated over the different servers in the architecture. In this way, the web application will have to concatenate the information received about the different instances and allow to the user to observe the monitoring details in a transparent and integrated way, without indicating him (unless he asks for it, for administration purposes) the information of the server where each activity was executed, accomplishing in this way location transparency [21] [22] [23].

C. *Application's architecture*

We can observe in Figure 3 the different distributed components identified in the architecture design, as well as the internal relationship between them and the user.

The solution is composed by three main nodes: the cloud, the embedded or traditional system and the monitoring application. The cloud works as the container of several elements: the BPMS, the monitoring application, the REST API used by the developers in order to integrate the applications with the process engine, and eventually a geolocation service which allows assigning to mobile clients the most convenient version of the service according to where they are.

On the other side we find the embedded type components, namely traditional BPM applications which belong to the organization, and because of different reasons like data sensibility or application portability, it could be decided not to locate them in the cloud. These nodes, functionally talking, take a role which is equivalent to the cloud node's behavior, even when they have access restrictions and lower computing force compared with the first ones.

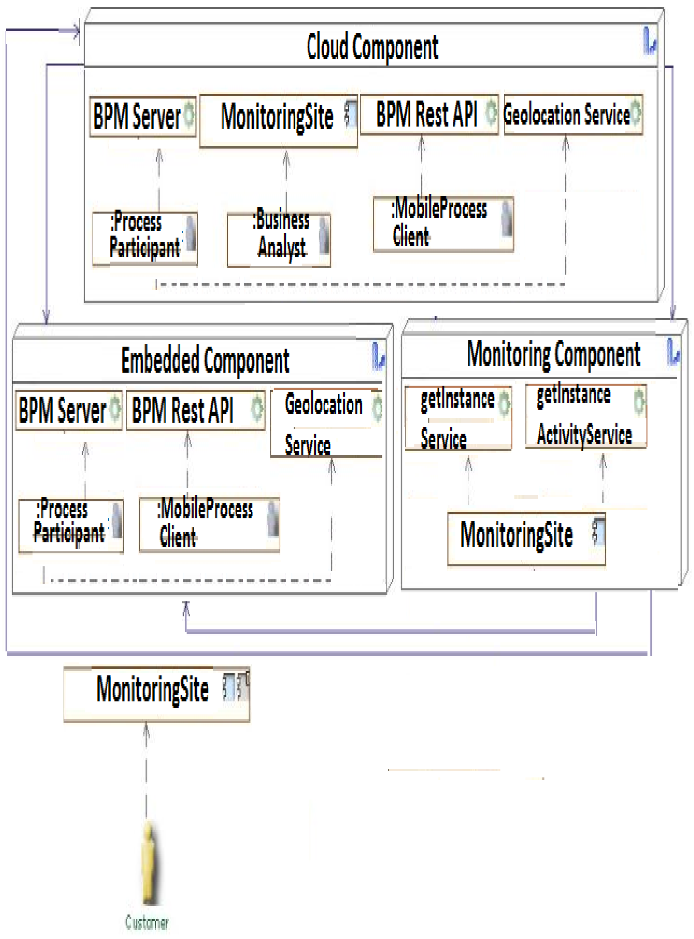


Fig.3. Application architecture and user location

The third component is related with monitoring. It is used by the monitoring application, and is in charge of returning information about instances and activities which were executed in every node of the distributed architecture. The web services getInstance and getInstanceActivity were constructed jointly with the monitoring application, and are executed on demand by this one. They are communicated with the process servers through an API (in our case, the Bonita one), and are in charge of returning, in first place, information about the instances initiated on each servers, and once these were accessed, return data about the activities that compose them [24] [25] [26].

D. *Component communication*

If we consider every component present in the architecture, we have analyzed the communication between each one of them through an application communication diagram. There we can observe the most important involved applications, their main actors and the interaction of the different distributed software components.

We can see at the same time the different user profiles involved in the execution of the components represented in the architecture.

While the preponderant role in the process execution is the activity's participant, the monitoring site results are important for the business analyst, as well as for the architecture administrators which can optimize the services or process components (Figure 4).

A feature in common between the process execution application and the monitoring one is the location transparency. Users should not be necessarily notified about the execution environment change, in case we are considering a decomposed process where the activities are located in different servers. This is very useful in order to allow users to have a unified vision of the process, more than a partitioned one, which main existence reason is related with taking advantage of technical resources.

We can also visualize in Fig. 4 how both the execution and the monitoring components access indistinctly to the cloud or embedded nodes, in order to gather information about each instance initiated in the distributed servers [27] [28].

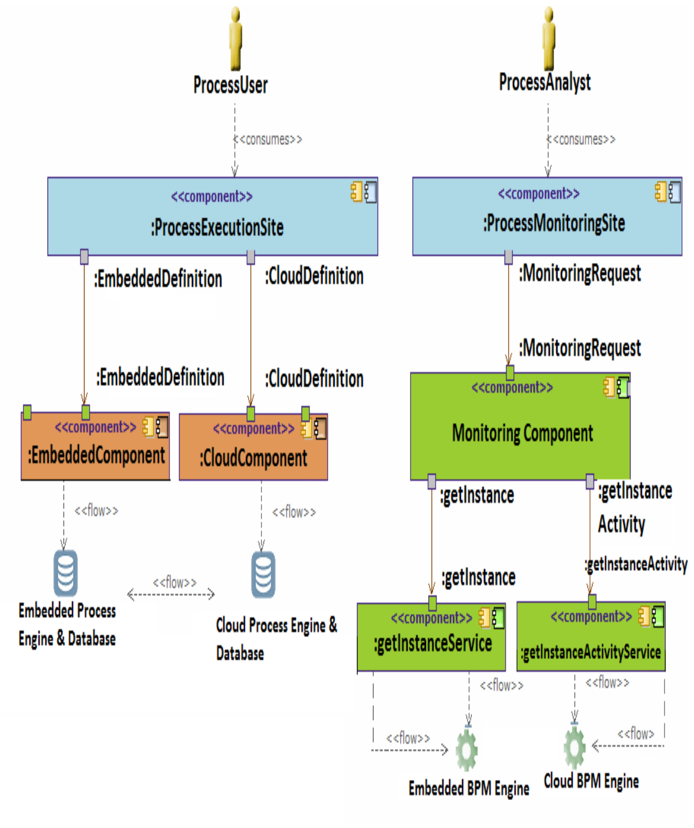


Fig.4. Application Communication Diagram

X. CONCLUSIONS

As we could observe, BPM as well as many other specialties in IT, have suffered changes due to the different service models in the cloud. This has forced specialists to consider new process design and implementation variants which allow using different advantages offered by the quoted paradigm. Facing the possibility of using unlimited computing force and high availability, some new decomposition process schemes appear in order to divide a process along some distributed server architecture.

Even when this approach allows using efficiently technological resources and protecting the organization's sensitive data, it is not necessarily easy to implement, and many times depends on the subjacent cloud infrastructure and the selected process server. In the present article we have used Bonita Open Solution because it is open source, and has an API which allows, through using connectors, accessing the different servers of the architecture. Without this last component it is very difficult to initiate new instances in different servers, and accomplish in this way the execution link of a decomposed and distributed process.

On the other side, as we said previously, even when process decomposition is a highly explored subject in current literature, the scenario is not the same with distributed process monitoring. This topic, at a glance, is not easily soluble. In a traditional business process model, the information source to monitor is in the same node that executes and monitors processes, while in a distributed environment instances are located in different servers. For this reason, different mechanisms are needed in order to gather data about executed instances, as well as to link them and provide an integration perspective under the light of the original process.

Currently, our research interest is focused on improving the monitoring application, allowing different filters for the users. The objective of this is to monitor efficiently each node of the architecture and optimize eventually the performance in some of them.

On the other side, it results important also to analyze different modifications to the BPMN notation present in current bibliography, which would allow including in process models semantic associated with decomposition, as well as interconnections between distributed servers.

REFERENCES

[1] T. Kirkham, S. Winfield, T. Haberecht, J. Müller, G. De Angelis, "The Challenge of Dynamic Services in Business Process Management", University of Nottingham, United Kingdom, Springer, 2011

[2] M. Minor, R. Bergmann, S. Görg, "Adaptive Workflow Management in the Cloud – Towards a Novel Platform as a Service", Business Information Systems II, University of Trier, Germany, 2012

[3] M Mevius, R. Stephan, P. Wiedmann, "Innovative Approach for Agile BPM", eKNOW 2013: The Fifth International Conference on Information, Process, and Knowledge Management, 2013.

[4] Dr. Manuel Goetz, "Integration of Business Process Management and Complex Event Processing", Germany, 2012.

[5] M. Gerhards, V. Sander, A. Belloum, "About the flexible Migration of Workflow Tasks to Clouds -Combining on and off premise Executions of Applications", CLOUD COMPUTING 2012: The Third International Conference on Cloud Computing, GRIDs, and Virtualization, 2012.

[6] Evert Duipmans, Dr. Luis Ferreira Pires, "Business Process Management in the cloud: Business Process as a Service (BPaaS)", University of Twente, April, 2012.

[7] S. Aleem, S. Molnar, and N. Mohamed, "Collaborative Business Process Modeling Approaches: A Review", In Proc. of the 2012 IEEE 21st International workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 274-279, June 2012.

[8] Dirk Fahland, Wil M.P. van der Aalst Eindhoven, "Simplifying Discovered Process Models in a Controlled Manner", University of Technology, The Netherlands, 2012.

[9] Hubert Scheuerlein, Falk Rauchfuss, Yves Dittmar, Rüdiger Molle, Torsten Lehmann, Nicole Pienkos, Utz Settmacher, "New methods for

clinical pathways –Business Process Modeling Notation (BPMN) and Tangible Business Process Modeling (t.BPM)". Springer-Verlag 2012.

[10] Marielba Zacarias, Paula Ventura Martins, "Collaborative methods for Business Process Discovery", Portugal, Springer-Verlag 2012.

[11] Jiri Kolar, Tomas Pitner, "Agile BPM in the age of Cloud technologies", Scalable Computing: Practice and Experience, 2012.

[12] Andreas Lehmann and Dirk Fahland , "Information Flow Security for Business Process Models - just one click away", University of Rostock, Germany, 2012.

[13] Rafael Accorsi, Thomas Stocker, Günter Müller, "On the Exploitation of Process Mining for Security Audits: The Process Discovery Case", Department of Telematics, University of Freiburg, Germany, 2012.

[14] Aleš Frece, Gregor Srdić, Matjaž B. Jurič, "BPM and iBPMS in the Cloud", Proceedings of the 1st International Conference on Cloud Assisted ServiceS, Bled, 25 Octubre 2012

[15] Dr. Luis Ferreira Pires, "Business Process Management in the cloud: Business Process as a Service (BPaaS)", University of Twente, April, 2012.

[16] S Balzert, P Fettke, P Loos, "A Framework for Reflective Business Process Management", 45th Hawaii International Conference on System Sciences, USA, 2012.

[17] Marco Brambilla, Piero Fraternali, and Carmen Vaca, "BPMN and Design Patterns for Engineering Social BPM Solutions", Politecnico di Milano, Piazza L. da Vinci 32, Milano, Italy, 2012

[18] Marco Brambilla, Piero Fraternali, Carmen Vaca, Stefano Butti, "Combining Social Web and BPM for Improving Enterprise Performances: the BPM4People Approach to Social BPM", WWW 2012, European Projects Track, Abril 16–20, Lyon, France, 2012.

[19] S. Balzert, P. Fettke, P. Loos, "Enhancement of traditional Business Process Management with reflection – a new perspective for Organizational Learning", Institute for Information Systems (IWi) at German Research Center for Artificial Intelligence (DFKI), Germany, 2012.

[20] Huang Hua, Zhang Yi-Lai, Zhang Min, "A Survey of Cloud Workflow", Jingdezhen Ceramic Institute, Jingdezhen, Jiangxi, 333001, China, Proceedings of the 2nd International Conference On Systems Engineering and Modeling (ICSEM-13), 2013.

[21] Toàn Nguyên, Jean-Antoine-Désidéri, "Resilience Issues for Application Workflows on Clouds", Project OPALE, INRIA Grenoble Rhône-Alpes, ICNS 2012: The Eighth International Conference on Networking and Services, Grenoble, France, 2012.

[22] Markus D□bhring and Birgit Zimmermann, "vBPMN: Event-Aware Workflow Variants by Weaving BPMN2 and Business Rules", SAP Research, Darmstadt, Germany, 2011.

[23] Zhenyu Fang, Changqing Yin, "BPM Architecture Design Based on Cloud Computing", School of Software Engineering, Tongji University, Intelligent Information Management, Shanghai, China, 2010.

[24] Duipmans E Pires L. da Silva Santos L. "Towards a BPM Cloud Architecture with Data and Activity Distribution" 2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops. ISBN 978-0-7695-4786-2/12

[25] T. Anstett, F. Leymann, R. Mietzner, and S. Strauch, "Towards bpel in the cloud: Exploiting different delivery models for the execution of business processes," in Proceedings of the 2009 Congress on Services - I. Washington, DC, USA: IEEE Computer Society, 2009, pp. 670–677.

[26] Roder, A.; Lehmann, M.; Kabitzsch, K., "Monitoring service choreographies," Industrial Informatics (INDIN), 2011 9th IEEE International Conference on , vol., no., pp.142,147, 26-29 July 2011. doi: 10.1109/INDIN.2011.6034852

[27] T. Dornemann, E. Juhnke, and B. Freisleben, "On demand resource provisioning for bpel workflows using amazon's elastic compute cloud," in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 140–147.

[28] W. Fdhila, U. Yildiz, and C. Godart, "A flexible approach for automatic process decentralization using dependency tables," in *ICWS*, 2009, pp. 847–855.

[29] J.Martinez Garro, P.Bazan "Constructing hybrid architectures and dynamic services in Cloud BPM¨ Science and Information Conference 2013 October 7-9, 2013 | London, UK.

[30] Bonita Open Solution http://es.bonitasoft.com/. October, 2013.