MBC - Mobile Business Collaboration

Bruno Ramôa Lima
bruno.lima@ist.utl.pt

Instituto Superior Técnico
PT Inovação

ABSTRACT

The inherent mobility of mobility devices is replacing the fixed work environments, making such devices a kind of indispensable good, both personally and professionally. The rise of Web 2.0 allows user interaction and collaboration, employing, for instance, web applications in mobile devices.

Today, SOA (Service Oriented Architecture) is the most widely adopted paradigm for development of distributed applications, increasing the interoperability between heterogeneous systems. The web services technology has become the preferred approach to implement solutions based on SOA paradigm.

Process-oriented business management aims to design and control the organizational structures in a very flexible way. Thus, these organizations can rapidly adapt to changing conditions of the organization's structure and operation. The BPM (Business Process Management) systems make the bridge between business analysts and the software developers.

This document describes a platform which allows the modeling and execution of specific tasks of a domain, using business processes. The activation and update of these business processes can be performed by users, using a client application independent of the used terminal (TV, Smartphone, Laptop, Tablet), or by external systems, using the web services technology.

The MBC project aims the integration of proposed solution with the collaborative platform developed by *PT Inovação*, called PUC.

Keywords

Business Process, Collaboration, Interoperability, SOA, Web Application, Web Services, Workflow.

1. INTRODUCTION

A business process is a sequence of activities performed by humans or systems to complete a business goal. Initially, processes were performed by humans who manipulated physical objects. Today, the information technologies allow systems to automate and dematerialize processes partially or totally. The automation of a business process is called workflow, which is described by a document using a workflow language, and it is executed in a workflow engine [6, 3].

The concept of SOA (Service Oriented Architecture) has been developed to provide interoperability advantages for organizational systems. Web services have been the chosen technology to achieve SOA because they allow the development of loosely coupled distributed business applications that are highly interoperable and cross organizational boundaries [18, 2].

People commonly collaborate with each other in their work life or simple daily social routines, so collaborative software (also called groupware) aims to support this kind of interaction. Groupware can be defined as: *"computer based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment"* [5]. The groupware makes the work more efficient, reduces the time spent in group activities, reduces the cost of carrying out group activities, and allows certain types of group tasks that would be impossible without the computer support.

Nowadays, almost everyone has a mobile device such as mobile phones, smartphones, laptops, and tablets. The inherent mobility of this type of devices is replacing the fixed work environments, making such devices a kind of indispensable good. Mobility by itself implies that a theoretical ideal of access, anytime and anywhere, is achieved. Furthermore, the rise of Web 2.0 also allows the users interacting and collaborating with each other, anytime and anywhere.

In this work, we aim to develop a platform (MBC - Mobile Business Collaboration) which allows the modeling and execution of specific tasks of a domain. The modeling can be made using predefined service nodes (also called work items) performing only the necessary customization, or using other node types. Thus, the system must provide a graphical tool for modeling business processes which allows customizing those service nodes. The business processes may be activated and updated by participants, using a graphical application, or by external systems, using the web services technology. The graphical application must be cross-platform and developed according to the Web 2.0 principles.

The system should provide a mechanism for monitoring the infrastructure since it may be necessary to perform cost accounting, maintain access information, and control processes' execution. Moreover, the solution should be scalable, reliable, and interoperable with either legacy application. We also integrate our MBC platform with the collaborative platform developed by a Telco and IT company (*PT Inovação*), called PUC (*Plataforma Unificada de Colaboração* or Unified Collaborative Platform). The purpose of this integration is to create, manage, and control collaborative sessions using business processes.

Document Roadmap

The remaining document is organized as follows: in Section 2, we make a survey about the state of the art in the areas of web applications, business processes, and enterprise integration; the architecture design of our solution is presented in Section 3, and the implementation is described in Section 4; in Section 5, we describe the results of the solution's evaluation; finally, in Section 6, we close the document with some conclusions of our work.

2. STATE OF THE ART

In order to activate and update the business processes, the developed system must offer web services which will be used either by users through a web application, or external systems through an API (Application Programming Interface). Moreover, the developed system must support the modeling and execution of workflows. Finally, it is necessary ensure that all developed components are interoperable with existent applications of an organization. To address the above issues, we present in this section three themes, which are: Web Applications (Section 2.1), Business Processes (Section 2.2), and Enterprise Integration (Section 2.3).

2.1 Web Applications

Over time, the web applications have evolved; the first-generation of web applications were developed in the early 1990s by using the basic web technologies such as HTML, web browsers, web servers, and CGI (Common Gateway Interface). The most used technology was based on RPC (Remote Procedure Calls). The second-generation web applications started using distributed object technologies, such as CORBA (Common Object request broker architecture) and DCOM (Distributed Computing Object Model) for development of more sophisticated applications. The third and current generation of web applications are taking advantage of developments in semantic web. The keystone of third generation is AJAX;[1] AJAX is a set of technologies (DOM, XML, XSLT, XMLHttpRequest, XMLHttpResponse, and Javascript) that allows increasing the interoperability between applications. AJAX was very important for the rise of Web 2.0 [20].

JSON[2] (JavaScript Object Notation) is a lightweight format used for computational data interchanging. JSON is quite simple, so it is increasingly used by developers. The JSON parsers are simpler than XML parsers, so this is the main advantage of JSON regarding the XML. The AJAX technology accepted the JSON format because JSON can be evaluated using the eval function which it is present in all web browsers.

Web services combine web and distributed objects into a single framework where the user-to-component interactions, as well as component-to-component, are conducted by using standard web technologies. The information is exchanged using SOAP[3] messages, and the network services are described as set of endpoints in a WSDL[4] document. HTTP is commonly used for the transport layer, but it is not a dependency. The basic communication pattern is synchronous, but web services also support asynchronous communications [20, 8].

However, web services introduce some overhead which cannot be ideal for mobile devices to process. Thus, Roy Fielding proposes another architecture style of networked systems, called REST[5] (Representational State Transfer). This architecture style is not a standard, but it uses some standards such as HTTP, URL, XML (resource representation), and MIME types (text/xml, application/json, etc.). REST is an architectural style where each of the system's available resources are represented in an unique URL. The purpose is to use standard HTTP functions (Get, Post, Put, and Delete) to access and modify those resources. Nevertheless, the implementation of web services using REST suffers some disadvantages such as the difficult management, security, and discovery of these web services given that no contract is established, as it happens with SOAP and the WSDL format.

2.2 Business Processes

The general function of a workflow system is to support the modeling and execution of business processes. Some standards like BPMN[6] and BPEL[7] have been proposed which are being adopted by industry [12]. Despite of these standards are being adopted, there are many other workflow solutions such as WS-CDL [14], YAWL [21], Triana [19], Taverna [16], and jBPM [10].

BPEL (Business Process Executing Language) is an executable workflow process that defines the flow of control and data between participant web services. It is an XML-based language, with support to handle variables with scopes, loops, conditional branches, synchronous and asynchronous communications, concurrent activities with correlated messages, transactions, and exceptions. BPEL is the standard language adopted to describe the business processes based on web services composition [18, 4]. According to [2] there are two complementary types of web services composition, orchestration and choreography. Orchestration is the composition of business processes via web services where there is a central process that controls and coordinates the other processes, on the other hand, with choreography composition, there is not a master process that controls and coordinates the other processes.

BPMN (Business Process Model and Notation) was developed by BPMI (Business Process Management Initiative), and it is currently maintained by the OMG (Object Management Group) since the two organizations merged in 2005. The current version of BPMN specification is 2.0, it not only defines a standard on how to graphically represent a business process like BPMN 1.x, but also includes execution semantics for the elements defined, and an XML format on how to store process definitions. A process defined with a graphical tool is a graph that describes the order in which a series of steps need to be executed using a flow chart [13].

WS-CDL (Web Services - Choreography Description Language) is a W3C candidate recommendation for web services choreography composition. Likewise, BPEL is the proposed standard to achieve the web services orchestration. WS-CDL is used in conjunction with BPEL, thus the business

[1] AJAX Tutorial - http://www.w3schools.com/ajax/default.asp
[2] JSON Project - http://json.org/
[3] SOAP Specifications - http://www.w3.org/TR/soap/
[4] WSDL Specifications - http://www.w3.org/TR/wsdl
[5] REST Way - http://www.xfront.com/REST-Web-Services.html
[6] BPMN Specifications - http://www.omg.org/bpmn/
[7] BPEL Specifications - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

process becomes more collaborative since the choreography is more collaborative than orchestration [14].

YAWL[8] (Yet Another Workflow Language) is a new proposal of a workflow solution which supports a concise and powerful workflow language, and handles complex data transformations, as well as web services integration. This solution defines the twenty most used workflow patterns divided in six groups: basic control-flow, advanced branching and synchronization, structural, multiples instances, state-based, and cancelation. Moreover, it supports tools like editor and engine which are freely available [21].

Triana[9] is a visual programming environment that allows users to compose applications from programming components, by dragging and dropping them into a workspace and connecting them together to build a workflow graph. It is good for automating repetitive tasks such as performing a find-and-replace on all the text files in a particular directory, or continuously monitoring the spectrum of data that comes from an experiment that runs for days or even years [19].

Taverna[10] is a powerful scientific workflow management application that allows designing and executing workflows. It provides a graphical workbench tool for both creating and running workflows. In Taverna, a workflow is considered to be a graph of processors, each of which transforms a set of data inputs into a set of data outputs. These workflows are represented in the Scufl language [16].

JBPM[11] (JBoss Business Process Management) is the open source WfMS (Workflow Management System) suited by JBoss. Initially, jBPM processes are created using a proprietary language of process definition called jPDL (jBPM Process Definition Language). jBPM5 is the latest version of the jBPM suite which is based on the BPMN 2.0 specification and supports the entire life cycle of the business processes (modeling, executing, monitoring, and management) [10].

2.3 Enterprise Integration

Enterprise Application Integration aims *"to connect and combine people, processes, systems, and technologies to ensure that the right people and the right processes have the right information and the right resources at the right time"* [1]. A typical scenario is that of an enterprise that runs hundreds or thousands of applications, which should be able to communicate and exchange data with each other, in order to work together for the business of the organization. In this section, we will address three approaches to solve enterprise integration issues, that are: ERP (Enterprise Resource Planning), EAI (Enterprise Application Integration), and ESB (Enterprise Service Bus).

ERP (Enterprise Resource Planning) was pioneering in enterprise integration software. It offers a system that accomplishes the integration of different functions in an organization. These functions assist the businesses in managing the important parts of the business, including product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service, and tracking orders. However, ERP has a centralized management model and it focused historically on integration of internal business functions. Moreover, when the applications interact with each other using a point-to-point model, the maintenance costs increase and the reuse of the applications becomes more difficult [11].

EAI (Enterprise Application Integration) is a framework used to perform integration of systems and applications across the enterprise. It is composed of a collection of technologies and services which form a middleware. EAI comprises message acceptance, transformation, translation, routing, guaranteed delivery, and business process management. Previously, integration of different systems required rewriting code on source and target systems, which in turn, consumed much time and money. Unlike traditional integration, EAI uses special middleware that serves as a bridge between different applications for system integration. Typically, the solution for EAI is to use Message-Oriented Middleware (MOM), this means the communication is asynchronous, but it can be synchronous as well. MOM products are usually built around a central message queue system, often called message broker, and all applications are connected to it [11, 17, 15].

ESB (Enterprise Service Bus) is a open standard, message based, distributed integration infrastructure that provides routing, protocol conversion, message format transformation, accept and deliver messages from various services and applications which are linked to ESB. Hereupon, the question on what exactly is the difference between ESB and EAI arises. Beyond the cost, which is significantly lower for ESB because it is based on open standards (no more technology lock-in), the ESB was developed to support the SOA paradigm, so it has all advantages offered by SOA [11, 15].

SOA (Service-Oriented Architecture) is a relatively newer form of information systems architecture and a real buzzword in the last few years in the domain of information system development. It can be defined as *"an architecture or development style that builds on distributed, loosely coupled, and interoperable components of software agents called services"* [9]. The applications are built of a set of collaborative services running in a distributed environment. Hence, it can be considered a peer-to-peer architecture, totally distributed, with the services distributed between different resources. The adoption of SOA may be unacceptable for some traditional enterprises because of the lack of sufficient control over the information assets. However, components are reusable, interoperable, and satisfy the needs and demands of dynamical business processes [22, 7].

3. ARCHITECTURE

In this section, we present the architectural design model for our solution. This architecture leverages the insights and experiences of the state of the art presented above (Section 2). Thereby, all systems are connected through a central bus which is based on ESB architecture. The developed system must support the modeling and execution of business processes, thus we chose the jBPM suite to support the entire life cycle of the business processes. Furthermore, our architecture embraces a central repository to allow the collaborative modeling of business processes. We develop a web application which is employed by users to activate and update the business processes. However, the business processes can also be activated and updated by external systems. At the end of this section, we detail our prototypical examples that allow executing specific tasks of a domain (Section 3.3).

[8] YAWL - http://www.yawlfoundation.org/

[9] Triana - http://www.trianacode.org/

[10] Taverna - http://www.taverna.org.uk/

[11] jBPM - http://www.jboss.org/jbpm/

3.1 Overview

The proposed solution comprises several entities that are involved in the communication process. Solution's architecture overview is depicted in Figure 1.

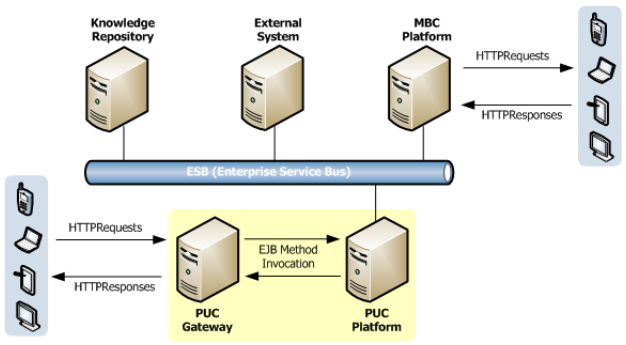

Figure 1: Architecture Overview

PUC is a collaboration platform developed by *PT Inovação*, so our solution assumes that the core groupware functionality is hereby deployed as well as the PUC Gateway which acts as a proxy server for the terminals, mediating access to the provided core collaborative functionality residing in the PUC.

ESB is the communication channel between the several entities. It receives the data from one system, if necessary it performs some mediating, and then it delivers information to the target system. Thus, the interoperability is achieved through some format conversion, however it is necessary to ensure that there is no incompatibility of features. This entity provides three types of connectors: EJB Connector, WS Connector, and REST Connector. The involved systems in the communication process can use different connectors, for instance, system A can use EJB Connector to send and receive information from bus, and likewise the system B can use WS Connector to perform the same operations, thus the system A and B can communicate with each other despite they use different communications ways.

Knowledge Repository stores the business processes of the organization. Thereby, it is possible for a partner to get a business process from another partner, and it is also possible to reuse business processes in different scopes. Moreover, it allows some collaboration in the modeling of business processes since it can be integrated with Oryx editor. Oryx[12] is a web-based editor for modeling business processes hosted at Google Code.

The MBC (Mobile Business Collaboration) platform provides a web application which can be used by users to update the business processes. Moreover, it also provides web services which can be used by external systems. The execution of business processes is carried out by the workflow engine which is provided by jBPM suite. MBC uses a knowledge agent to get processes from repository. If a process is already loaded in the knowledge base, it is not necessary get it from repository, unless its version has changed. This platform uses an external database to assure the persistence of the data. MBC uses an external resource (e.g. Email Resource) to send notifications to the users.

[12]Oryx editor - `http://code.google.com/p/oryx-editor/`

3.2 System Components

We now describe the components of ESB, Knowledge Repository, and MBC. We do not describe the PUC platform's components because it is irrelevant for the scope of this document.

ESB's Components. ESB comprises two components, Connector and Adapter. The Connector is used to get an interface to communicate with the target system. On the other hand, the connector uses the Adapter to perform some mediation before it sends the data to the other connector. As stated before, we provide only three types of connectors because our solution just needs these types. We do not use an existing implementation of ESB because this entity is quite simple for the scope of this work, and the existing implementations are quite complex which are used in other contexts. ESB's architecture is depicted in Figure 2.

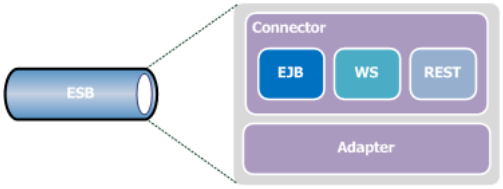

Figure 2: ESB's Architecture

Knowledge Repository's Components. In order to store the business processes, we use the Drools Guvnor[13] that is a centralised knowledge repository developed and maintained by JBoss Community.[14] The Guvnor provides a rich web-based GUI that allows managing the knowledge bases. Moreover, it can be integrated with the Oryx editor which allows modeling business processes that are stored in the repository. The changes in business processes are saved in repository which provides a version control system. When a user is editing a resource (business process), the Guvnor locks this resource, and thus it does not allow that other users edit the same resource at the same time. Figure 3 shows the major components of the system and how they are integrated and deployed.

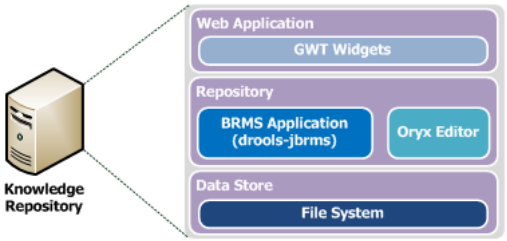

Figure 3: Knowledge Repository's Architecture

MBC's Components. We will now describe the MBC platform's components of each layer, whose architecture is depicted in Figure 4. The solution's architecture proposed comprises four layers, which are: Domain Layer, Business

[13]Drools Guvnor - `http://www.jboss.org/drools/drools-guvnor.html`
[14]JBoss Community - `http://community.jboss.org/`

Layer, Service Layer, and Presentation Layer. Multi-tier application architecture provides a model for developers to create a flexible and reusable application.

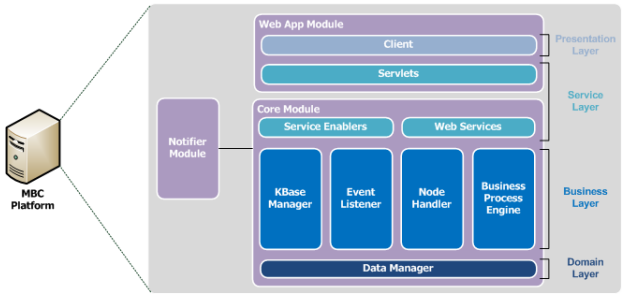

Figure 4: MBC's Architecture

Web application module is divided in two components, client and server. The client component is the graphical interface that allows, for instance, the users choose and start a business process. The client architecture is depicted in Figure 5. The server component provides servlets which are used by web browsers. On the other hand, the server component uses the API provided by the Services Enablers, likewise, the external systems use the API provided by the Web Services. Service Enablers implement the services in order to explore the core functionalities.

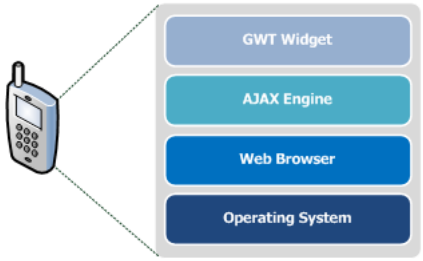

Figure 5: Client Component's Architecture

The jBPM suite provides an API to interact with the process engine. However, the system needs to set up a session to communicate with the engine. On the other hand, the session needs to have a reference to a knowledge base. Hence, when the system receives a request to start a process, it uses the agent, called KBase Manager, to get the process's definitions and to create the session. KBase Manager is used to look up the process definitions whenever necessary

Whenever the engine starts a new process, it creates a new instance for that process definition and maintains the state of that specific instance. Data Manager is responsible for ensuring the persistence of the process instances and all other data associated with the business processes. Moreover, it stores in persistent way all other data of the system. This component loads data from database whenever necessary. Likewise, it stores the new data in the database also whenever necessary.

The jBPM suite provides methods for registering and removing listeners. A listener can be used to listen process-related events, like starting or completing a process, entering and leaving a node, etc. An event object provides access to related information, like the process instance and node instance. The Event Listener component just receives the events and processes it. At the moment, we use this component to remove unused objects from the database, when a process instance ends.

The domain-specific tasks are targeted to one particular application domain. However, we proposed the development of work items that can be used across domains. All work items are driven like a state machine. So, the work item has at least two stages, First and Last. Basically, in the first stage we read the work item parameters (input data), and in the last state we write the results (output data) in the database. In the last stage, we also send the *Complete Command* to the business process engine. All other stages are specific and optional for each work item.

All work items have the same Node Handler associated. The Node Handler is responsible for selecting the appropriated service for the running work item. On the other hand, the service is responsible for the implementation of the functionalities provided by the work item. When the service completes the task, the work item is completed, so the service sends this information to the engine. The engine goes to the next node. If there is no next node, the business process ends.

3.3 Prototypical Examples

As stated before, we developed work items that can be used across domains. Work items represent atomic units of work that need to be executed. These nodes specify the work that should be executed in the context of a process in a declarative way, specifying what should be executed (and not how) on a higher level (no code), and hiding implementation details. Table 1 shows the developed work items as well as the location of the services' implementation.

Work Item Name	Service's Implementation
ApproverNode	MBC (core)
SelectUserNode	MBC (core)
SelectFeatureNode	MBC (core)
PUCNode	PUC (resource)
WSNode	ESB

Table 1: Developed Work Items

ApproverNode - This node approves some deal or management decision, for which is necessary a rate of positive votes. The participants have different vote weight according to the its privilege level. If the rate of positive votes is equal or greater than a selected rate, the work item goes to next stage. Otherwise, the business process is aborted after the total weight of negative votes is found that prevents deal approvement. At each stage, the users receive a notification using, for instance, the email resource. The purpose of the notification is alert the users that they have a pending task in the MBC portal (web application). When a business process is aborted, the actors (users) receive a notification informing them that the task was cancelled. If the actors of some stage do not answer to notification, the system resends another notification (repeats the stage). The system tries N times and then, if the work item does not have yet the necessary information so it can go to the next stage, the process is aborted.

SelectUserNode - The purpose of this node is to create a list of users (participants), which can be used by the other work items of the same process. At each stage, the users also receive a notification. The flow of execution is very similar to the *ApproverNode*. Unlike the *ApproverNode*, *SelectUserNode* typically is used to gather the list of participants and save it in the database. Hence, in this situation a business process must have more than one work item.

SelectFeatureNode - The purpose of this node is to create a list of features, which can be used by other work items of the same process. At each stage, the users also receive a notification. The flow of execution is also very similar to the *ApproverNode*. Again, unlike the *ApproverNode*, *SelectFeatureNode* typically is used to gather the list of features and save it in the database. Hence, in this situation a business process must have more than one work item.

PUCNode - This work item has a flow of execution very similar to the previous work items. The purpose of this node is to schedule a session in the PUC platform. If the work item opens a session successfully in the PUC platform, it completes the job. Otherwise, it aborts the process. The session is open at the specified date. At each stage, the users also receive a notification.

WSNode - This node is used for web services composition. Basically, we use this work item to invoke web services deployed elsewhere. This work item can save and share data with other work items of the same process. Thus, it is possible to use the results of the invocation of a web service, as input data for invocation of another web service. The purpose of this work item, in this project, is to open a session in the PUC platform using the web services composition.

4. IMPLEMENTATION

As seen before, the MBC platform provides a web-based GUI which allows users interact with the system. The GUI widgets for the web front end were developed with GWT (Google Web Toolkit). GWT[15] is a toolkit for building complex Web applications. Its main singularity is that applications are developed with a set of core Java APIs and pre-defined widgets, making it possible to write AJAX applications in Java and then compile the produced source code to a highly optimised JavaScript that runs across all modern desktop browsers, including even, some mobile browsers. GWT also makes the interaction with handwritten JavaScript possible by using its JavaScript Native Interface (JSNI).[16] In order to support communication with back-end servers in its applications, GWT also provides the GWT RPC framework that transparently makes calls to GWT Java servlets through a customised RPC protocol, and takes care of object serialisation and other details. However, we use REST style architecture to support communication purposes because it is the most lightweight and interoperable alternative.

The servlets respond to HTTP requests performed by clients, using some method provided by HTTP protocol (GET, POST, PUT, DELETE). Each method executes a different service, for instance, the GET method is used to get an object from server. The payload of the HTTP protocol is filled with a JSON object. The system provides a servlet for each domain object that is used in the communication between client and server. Nevertheless, the system also provides a servlet that only accept requests using POST method. The service name, as well as other parameters, are described using the JSON representation. This servlet is used to perform generic business tasks, for instance, the *Sign-in Service*.

Core module represents the main work of our implementation. This module comprises components from service, business, and domain layer (Figure 4). These components were developed based on EJB technology. For instance, the Service Enablers component was developed using stateless beans, which explore core functionalities. We use stateless beans because the client's state does not need to be saved between the method invocations, also favouring scalability. Furthermore, these beans can be executed in any application server.

KBase Manager is a singleton bean, thus the container only has one instance for this bean, which can be invoked concurrently by multiple threads (clients). This bean stores the knowledge base which is shared between sessions (clients). The task of creating a knowledge base can be rather heavyweight. This component uses a map to control the validity of a resource (business process). The validity period is defined in seconds, and it can be changed in the ESB's configuration file.

The engine, by default, does not persist the business processes instances, if unexpected failure occurs, all running instances are lost. Moreover, without data persistence the system cannot remove the running instances from memory and restore them at some later time. Therefore, we set up the engine to store the data in database through the entity manager defined in the session's environment by KBase Manager component.

The communication with the repository is performed using the central bus (ESB). Basically, the ESB provides a bean that can be accessed remotely. This bean uses an HTTP client to access the Guvnor since this platform provides a REST API.

Data Manager uses a container-managed entity manager. Thus, the container is responsible for the opening and closing of the entity manager (this is transparent to the application). It is also responsible for transaction boundaries. The persistence of the data was implemented using the Hibernate[17] framework. Furthermore, the data manager component uses an external object-relational database system, which is an open source implementation provided by PostgreSQL[18] community, to store the data.

Notifier module is used to send notifications (email, sms, etc.) to the MBC's users. However, we only develop the Email Notifier.

The Email Notifier uses the email service provided by JBoss AS.[19] We configured this service in order to use a Gmail account to send the emails to the users. The service is retrieved from JNDI (Java Naming and Directory Interface) lookup.

[15] GWT overview - `http://code.google.com/intl/pt-PT/webtoolkit/overview.html`

[16] JSNI overview - `http://code.google.com/intl/pt-PT/webtoolkit/doc/latest/DevGuideCodingBasicsJSNI.html`

[17] Hibernate - `http://www.hibernate.org/`

[18] PostgreSQL - `http://www.postgresql.org/`

[19] JBoss AS - `http://www.jboss.org/jbossas`

5. EVALUATION

Now, in this section, we present the evaluation of our solution, which has been built in hopes of ultimately being a usable and commercially viable solution that aims to be deployed in *PT Inovação*'s context. Hereby, we will analyse our test results and try to make a realistic evaluation, considering this ultimate objective. Our solution's assessment is based on qualitative (Section 5.1) aspects and quantitative (Section 5.2) metrics.

Concerning our server-side implementation (MBC platform), we analyse the *performance* and *scalability*. The performance includes the amount of memory and CPU load used, according to the number of users per business process, and the number of business processes per user.

Moreover, we also analyse the behaviour in the database server, when the MBC platform receives several requests, at the same time. Basically, we monitor the number of open connections between the database server and MBC platform.

In the client-side, we measured the application loading time and content length. Furthermore, we also measured the time to serve the requests to the server and their respective content length. For an analysis of the application's *presentation* and *usability*, we also present screenshots of own prototype.

5.1 Qualitative Evaluation

For qualitative evaluation, we considered the *presentation* and *usability* aspects, using for this purpose the *PUCNode* once the widgets for the other nodes are similar. Furthermore, in the implementation phase, we aimed to maximise extensibility and interoperability of all developed code.

Figure 6 shows the second stage of the *PUCNode* that is used by the users to fill and select some parameters, which are used to create a new session in the PUC platform.

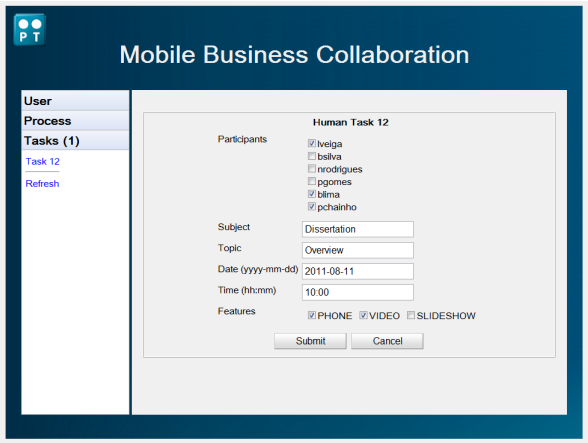

Figure 6: *PUCNode* Widget in the 2nd stage

5.2 Quantitative Evaluation

The quantitative evaluation is divided in three parts: MBC platform; Database; Client. However, our *performance* and *scalability* evaluation was mainly focused on the MBC platform. We performed load tests to measure the memory and CPU consumption using the *ApproverNode*.

MBC Platform

We performed two kinds of tests in order to evaluate the performance and scalability of the MBC platform. The tests are defined by the number of clients in simultaneous execution, and the number of business processes per user as well as the number of users per process.

We used a uniform random distribution (u.r.d.) because it enables a good characterization that spans different user population sizes and amount of business processes. Furthermore, it allows a better simulation of the real world. We used a function to generate random numbers (integers), which generates sequences of numbers with different probabilities. Hence, if we generate a sequence of numbers, defined in a finite interval, the average of the generated numbers may not be the median value of the interval. For instance, in the interval [2; 6] the median value is 4, so the average of the generated sequences should be 4, if the numbers had the same probability and the sequence size was unbounded. Thus, in the analysis of the tests' results, we took this aspect into account.

In the first test, we increase the number of running business processes with a variation of the number of users per business process. The number of users per process varies randomly between 2 and 6 (u.r.d. with average 4). Thus, for 400 business processes, the system may have at maximum 2400, and on average 1600 users connected, since the same user is not associated with more than one business process. The results are depicted in Figure 7 and 8.

The average CPU utilization grows linearly with the number of running business processes, and it never exceeds 65% (Figure 7). The growth is not completely linear because we used a u.r.d., and the total of users has slight random variation in each sample, for reasons stated before. However, these values are substantially the same.

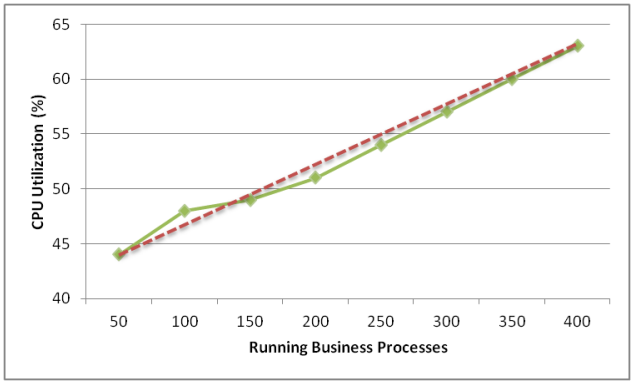

Figure 7: CPU utilization growth with the number of running business processes

Likewise, the memory utilization grows with the number of running business processes (Figure 8). Again, the growth is not completely linear due to the reasons stated above.

In the second test, we increase the number of active users with a variation of the number of business processes per user. The number of business processes per user varies randomly between 1 and 3 (u.r.d. with average 2). Thus, for 100 users, the system may have at maximum 300, and on average 200 running business processes since the same business process is only associated with one user. The results are depicted in Figure 9 and 10.

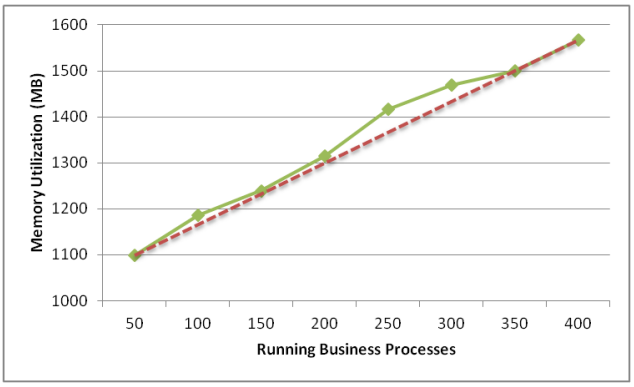

Figure 8: Memory utilization growth with the number of running business processes

The measured values of CPU utilization are lower than in the first test because the number of users is much lower. In the second test we have a maximum of 100 users, while in first test the system may have 2400 users (Figure 9).

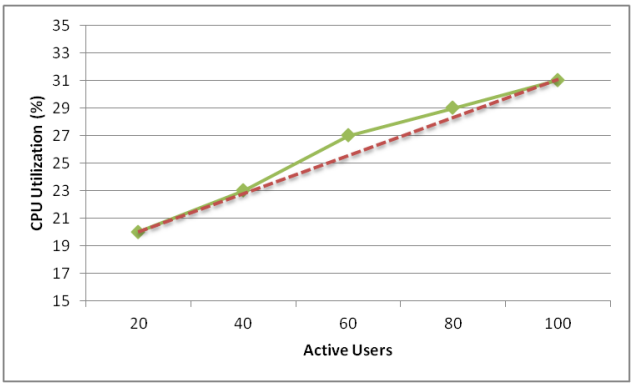

Figure 9: CPU utilization growth with the number of active users

Regarding memory, the measured values are very similar because we use an external database, and it allows removing the running instances of business processes from memory and restoring them at some later time (Figure 10).

In both tests, we measured values that are acceptable according to the typical features of an enterprise server.

In a scalable system we expect that the CPU and memory utilization grows linearly with the number of running business processes and active users, so we may consider our solution as a scalable system.

Database

In order to analyse the usage of database local connections, we periodically gathered the number of open connections using a SQL command. We used the same tests described above. The tests' results are depicted in Figure 11 and 12.

We verified that the system has some limitations regarding open connections because we are using an open source database, which has a threshold of 100 connections. Although this value can be raised, this is not recommended since the management of connections may not be safe.[20]

[20] `http://www.postgresql.org/docs/8.2/static/runtime-config-connection.html`

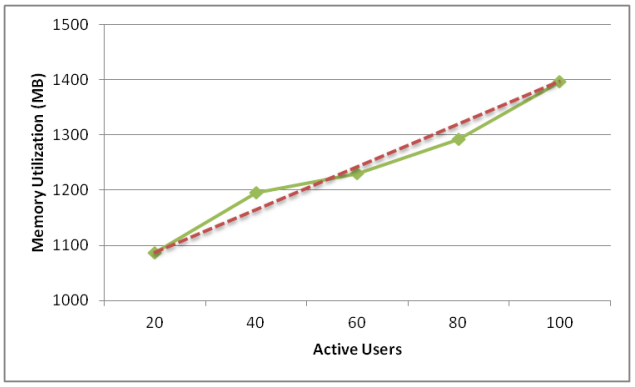

Figure 10: Memory utilization growth with the number of active users

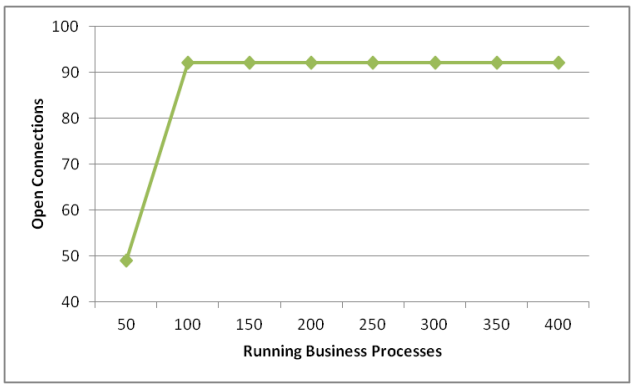

Figure 11: Open connections growth with the number of running business processes

When the system does not have more available connections in the pool (set of available connections), it waits by one (30 seconds). After this time, if the pool does not have yet one available connection, the request is aborted and one exception is thrown. In the third test (Figure 11) in spite of the threshold of open connections having been reached, the waiting time typically was enough so that all requests finished successfully. However, in some situations, this time was not enough and some requests were indeed aborted.

Client

In the client-side, we used the *PUCNode* to measure the application loading time and content length as well as the time to serve the requests to the server and their respective content length. These results are depicted in Table 2.

The measured values are different between the first times and following times because in the following times there is static content, like images, which are not downloaded from server, as they are saved in cache. All of these values are acceptable according to the typical features found today for our target devices (TV, Smartphone, Laptop, Tablet).

The content length varies according to the type of request performed. Furthermore, the widgets have fields with variable length, so the resultant length of JSON strings also varies. The time for sending data increases according to the length of the resultant message (JSON string).

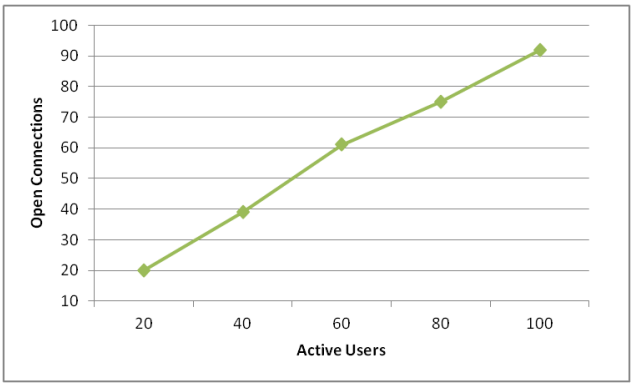

Figure 12: Open connections growth with the number of active users

We tested our web application in several browsers, including Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and Apple Safari. Our application works fine in all of these browsers, using a laptop device running Windows 7 operating system. Moreover, we also tested the web application in a smartphone device running Android 2.3 operating system. Again, our application operates correctly. However, the usability in the Smartphone device is poorer than Laptop because of the screen size.

Loading time (1st time)	532 ms
Loading content length (1st time)	459 KB
Loading time (next time)	113 ms
Loading content length (next time)	1.26 KB
Time for sign-in service (1st time*)	31ms
Content length for sign-in service (1st time*)	6.33 KB
Time for sign-in service (next time*)	21 ms
Content length for sign-in service (next time*)	327 B
Time for sign-in service (next time**)	39ms
Content length for sign-in service (next time**)	894 B
Time for sending data - 1st stage	414 ms
Content length for sending data - 1st stage	149 B
Time for sending data - 2nd stage	95 ms
Content length for sending data - 2nd stage	149 B

* without pending tasks
** with three pending tasks

Table 2: Results of the client-side evaluation

6. CONCLUSIONS

Our objectives led us in an elaboration of an extensive survey in topics related with web applications, modeling and execution of business processes, and enterprise integration (interoperability between enterprise applications). This allowed us to build a necessary knowledge base for our solution's design phase. It should be clear by now that the accomplishment of work's goals requires special attention to all of the aforementioned issues. The survey of these topics was presented in the state of the art section (Section 2).

The implementation (Section 4) aimed to maximise extensibility and interoperability of all developed code. Moreover, we aimed the integration of the proposed solution with external entities, such as the PUC platform.

The developed solution allows modeling business processes collaboratively, using the Oryx editor. The business processes are stored in the central repository. Thereby, it is possible for a partner to get a business process from another partner, and it is also possible to reuse business processes in different scopes.

The activation and update of the business processes can be performed by users, using a web application that is independent of the used terminal (TV, Smartphone, Laptop, Tablet), or by external systems, using the web services technology.

The modeling of the business processes can be made using service nodes performing only the necessary customization, or using default node types. We developed several work items, which allow, for instance, an organization that wants to gather a rate of positive votes to approve a deal or management decision. We also developed a work item that allows to schedule a new session in the PUC platform, and still, we developed a work item that allows to invoke web services deployed elsewhere. All developed work items can save and share data with other work items of the same process.

We use the Email Notifier to send notifications to the MBC's users. The purpose of the notifications is alert the users that they have a pending task in the MBC portal (web application).

In the end, we find that the system had a stable behaviour for the used load in the tests, and it is a good solution to use with mobile devices since the measured values are compatible with the capacity and communication costs of these types of devices.

We hope that the gained knowledge base, during the course of this work, is sufficient for the future development of functional requirements and desired properties, and that our solution, as it is today, becomes useful for future commercial developments in *PT Inovação*'s context.

7. REFERENCES

[1] W. Brosey, R. Neal, and D. Marks. Grand challenges of enterprise integration. In *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, volume 2, pages 221–227. IEEE, 2002.

[2] A. Charfi and M. Mezini. Ao4bpel: An aspect-oriented extension to bpel. *World Wide Web*, 10(3):309–344, 2007.

[3] Q. Chen and M. Hsu. Inter-enterprise collaborative business process management. In *icccn*, page 0253. Published by the IEEE Computer Society, 2001.

[4] C. Courbis and A. Finkelstein. Towards an aspect weaving BPEL engine. In *The Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Lancaster, UK*. Citeseer, 2004.

[5] C. Ellis, S. Gibbs, and G. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1):39–58, 1991.

[6] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.

[7] S. Geric. The potential of service-oriented architectures. In *Information Technology Interfaces (ITI), 2010 32nd International Conference on*, pages 471–476. IEEE, 2010.

[8] N. Ibrahim. A survey on different interoperability frameworks of SOA systems towards seamless interoperability. In *Information Technology (ITSim), 2010 International Symposium*, volume 3, pages 1119–1123. IEEE, 2010.

[9] K. Kanchanavipu. An Integrated Model for SOA Governance. *Report/IT University of Goteborg 2008: 002*, 2008.

[10] J. Koenig. Jboss jbpm. *White Paper, JBoss Inc., Available from*, 2004.

[11] J. Lee, K. Siau, and S. Hong. Enterprise Integration with ERP and EAI. *Communications of the ACM*, 46(2):54–60, 2003.

[12] C. Liu, Q. Li, and X. Zhao. Challenges and opportunities in collaborative business process management: overview of recent advances and introduction to the special issue. *Information Systems Frontiers*, 11(3):201–209, 2009.

[13] A. Lonjon. Business process modeling and standardization. *BPTrends, in http://www. bptrends. com*, 2004.

[14] J. Mendling and M. Hafner. From inter-organizational workflows to process execution: generating BPEL from WS-CDL. In *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, pages 506–515. Springer, 2005.

[15] F. Menge. Enterprise service bus. In *Free And Open Source Software Conference*, 2007.

[16] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 2004.

[17] A. Parr and G. Shanks. A taxonomy of ERP implementation approaches. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, page 10. IEEE, 2002.

[18] J. Pasley. How BPEL and SOA are changing Web services development. *Internet Computing, IEEE*, 9(3):60–67, 2005.

[19] I. Taylor, M. Shields, I. Wang, and R. Philp. Grid enabling applications using triana. In *Workshop on Grid Applications and Programming Tools*. Citeseer, 2003.

[20] A. Umar. The emerging role of the web for enterprise applications and ASPs. *Proceedings of the IEEE*, 92(9):1420–1438, 2004.

[21] W. Van Der Aalst and A. Ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[22] A. Zalewski. A FMECA framework for Service Oriented Systems based on Web Services. In *Dependability of Computer Systems, 2007. DepCoS-RELCOMEX'07. 2nd International Conference on*, pages 286–293. IEEE, 2007.