Computer Science • 13 (4) 2012 http://dx.doi.org/10.7494/csci.2012.13.4.115

Maciej Hamiga
Marcin Jarząb

AN ANALYSIS OF METHODS FOR SHARING AN ELECTRONIC PLATFORM OF PUBLIC ADMINISTRATION SERVICES USING CLOUD COMPUTING AND SERVICE ORIENTED ARCHITECTURE

Abstract *This paper presents a case study on how to design and implement a public administration services platform, using the SOA paradigm and cloud model for sharing among citizens belonging to particular districts and provinces, providing tight integration with an existing ePUAP system. The basic requirements, architecture and implementation of the platform are all discussed. Practical evaluation of the solution is elaborated using real-case scenario of the Business Process Management related activities.*

Keywords ePUAP, public administration, SOA, cloud, architecture

1. Introduction

As the Internet becomes one of the most important means of communication, public administration tries to follow this trend, by offering its services through the World Wide Web. In many countries, IT systems are being created, which not only provide web-based access to public administration for citizens but also try to digitalize the whole business process workflow of these institutions.

The purpose of this paper is to analyze and propose an effective solution which aims at moving public administration services into a cloud environment [4] using a Service Oriented Architecture (SOA) paradigm. Changing the old, on-premises deployment model to cloud-based hosting can improve cost-efficiency, provide better reliability, scalability and unify the IT infrastructure country-wide whereas SOA provides capabilities for the development of applications by combining loosely coupled and interoperable services.

Such an approach, however, also brings new challenges that need to be faced, such as quality of services assurance, adequate tenant isolation level and security, system scaling and failover capabilities. All these factors need to be taken into account during project and implementation phases.

1.1. What is ePUAP

The Electronic Platform of Public Administration Services (ePUAP—Polish acronym) is a computer system that aims at facilitating and accelerating the process of computerization of Polish public administration services, country-wide. A centralized point of access to all services for the citizens is an additional and important benefit.

The system itself is accessible through the web, both for administrators and regular users, and offers a set of services, from which any subset can be chosen by the public administration unit to implement. These services are as follows:

- Communication services — including sending and receiving electronic documents, with various options of validation / receipt acknowledgment.
- Security services — such as Single Sign-On (SSO), citizen identification number verification.
- Coordination services — simplified environment enabling the use of business process modeling software.
- Catalog services — centralized repository of document templates, procedure descriptions etc.

All services are published using Web Services standards, which makes integration with external systems possible. After registering certificates in the ePUAP system, external applications can pull or push documents to the public administration units in an automated way. Public administration unit themselves can also make use of the integration mechanisms, to hook up their own electronic document systems into ePUAP.

1.2. Problem statement and research contribution, related work

While ePUAP is a step forward in the process of digitalization of public administration, it still does not completely address the problem of on-premises software systems that are functioning in administration units all around the country. ePUAP is merely a front-end, acting as a bus to which proprietary local governments computer systems can plug-in and does not take into account the heterogeneity of business processes among various institutions. While input data formats (i.e. documents that citizens file in) are often standardized, the workflow of business processing can significantly differ between organizations.

There is already some research about models of how to utilize Cloud computing and SOA in the context of eGovernment public services [7]. In [5] there are elaborated techniques and patterns that should be used to transform electronic government solutions based on traditional architectures to these new paradigms. They state that there is a strong requirement for clear legal regulations, strong leadership of the national government and environment which is able to sustain technology innovations. An interesting initiative is also supervised by the United States Government and General Services Administrator to launch the cloud computing platform Apps.gov [1] hosting large portions of federal government infrastructure. The platform provides access to storage and compute resources and also exposes business, productivity and social apps for customers.

As a solution to the problem described above, this paper aims at proposing a technological solution that would move on-premises public administration software installation into a centrally hosted, IT cloud based on SOA paradigm. The proposed technological stack is used to implement a platform which integrates with ePUAP to realize real-life use cases and promotes standardization of the IT infrastructure and software while enhancing cost effectiveness, reliability and accessibility of electronic public administration services.

This paper is organized as follows. Section 2 presents a vision of ePUAP in a cloud using the SOA approach. The architecture and implementation are presented in sections 3 and 4 respectively. In section 5 a case study is presented. The paper ends with conclusions and a list of possible improvements which could be considered as a part of future work.

2. Vision of ePUAP in the cloud using SOA

The vision of public administration IT services hosted in a cloud brings multiple advantages and possibilities for organizations that host their own, local computer systems or do not own such systems at all (Fig. 1). All hardware resources would be moved to centralized data centers — dependently on financial possibilities: multiple across the country, or single with a disaster recovery backup location. All services would be made available to service consumers by the data center using virtualization techniques for storage, compute and network resources. Hardware needed on the or-

ganization side (local government agency) would consist only of personal computers. Citizens would use an ePUAP portal which is integrated with particular cloud instances, serving a given group of citizens assigned to country region i.e. province and district.

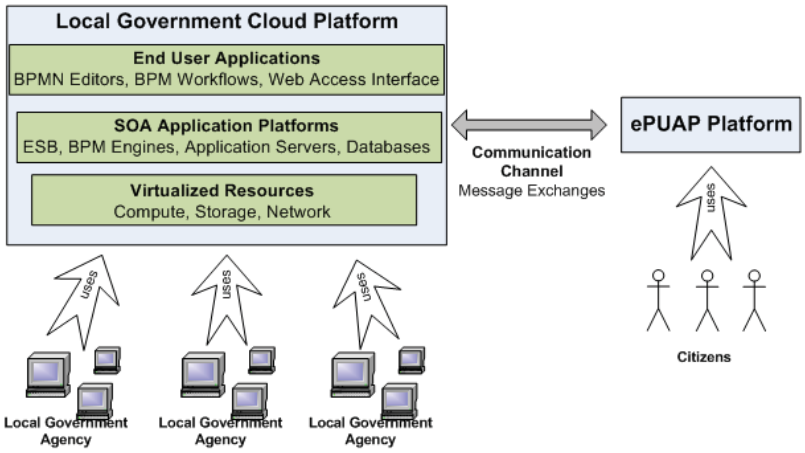

Figure 1. Vision of ePUAP platform in the cloud based on SOA paradigm.

Service Oriented Architecture would be introduced, encouraging software collaboration though the use of standards-based communication protocols and constructs such as Enterprise Service Bus (ESB) that enable message mediation between various message formats. Existing proprietary, legacy applications can be either rewritten as SOA components and plugged into the ESB or left in place, with software communication adapters installed. That would ensure message routing between legacy applications and the rest of the system.

To simplify the process of porting internal business processes to the new system, utilization of high level design tools, such as Business Process Modeling Notation (BPMN) diagram editors, business analysts cooperating with developers will be able to model organization business workflows which will make use of all of the services available in the cloud environment.

Access to all services migrated to the cloud will be web-based, with web application providing access to all components of the system. Proprietary web applications, internal portals etc. can also be hosted in the cloud, making use of virtualized infrastructure and high availability clusters of application servers.

Additionally, integration with the current ePUAP solution should be provided. In the beginning, ePUAP could serve as a centralized front-end for public administration services country-wide. During the next stages of movement to cloud infrastructure, such integration could be tightened, alongside with moving the ePUAP itself into the common public administration cloud.

3. Architecture

The SOA Solution Stack (S3) [3, 10] proposed by IBM elaborates the process of SOA applications development and deployment. The S3 model presented in Figure 2 provides a detailed description of architectural elements divided into nine layers. Each layer has a physical and logical aspect and lets the organization define the degree of consumer-provider integration. The existence of both functional and non-functional service requirements is also assumed, which establish the SOAs objective. S3s nine layers are operational systems, service component, services, business process, consumer, integration, QoS, information architecture, and governance and policies. The five horizontal layers relate to the overall functionality of the SOA solution, which are cut-crossed by the vertical layers that are non-functional in nature.

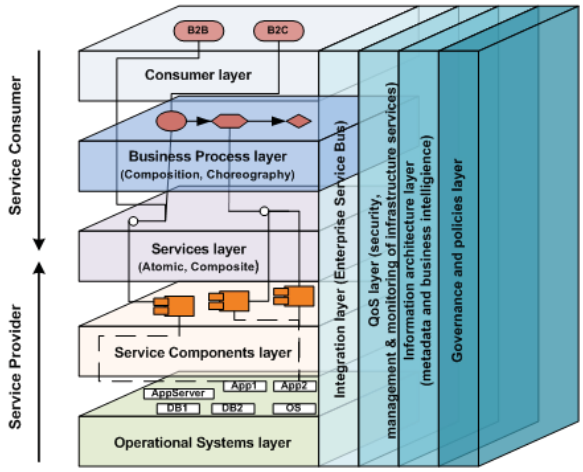

Figure 2. S3 SOA reference architecture.

Modern computing infrastructures for applications should be created according to architectures that provide flexible platform and operational capabilities for the provisioning of required computational resources by applications. This can be attained by introducing service orientation to the concepts of orchestration and management of computational infrastructure through a Service-Oriented Infrastructure (SOI) [8] paradigm.

The proposed architecture (Figure 3) delivers virtualized and physical services that can be described in categories such as Hardware as a Service (HaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), Workflow as a Service (WaaS) which further can be aligned with the S3 model. The presented architecture is enough general and can be orthogonally analyzed not only in the ePUAP context.

The framework enabling realization of such architecture must be modular with components that are configurable to fulfill the constantly changing requirements and evolution of IT technologies. They require solutions that not only correspond to current requirements, but also allow to evolution and adaptation, as technology advances and new requirements emerge.

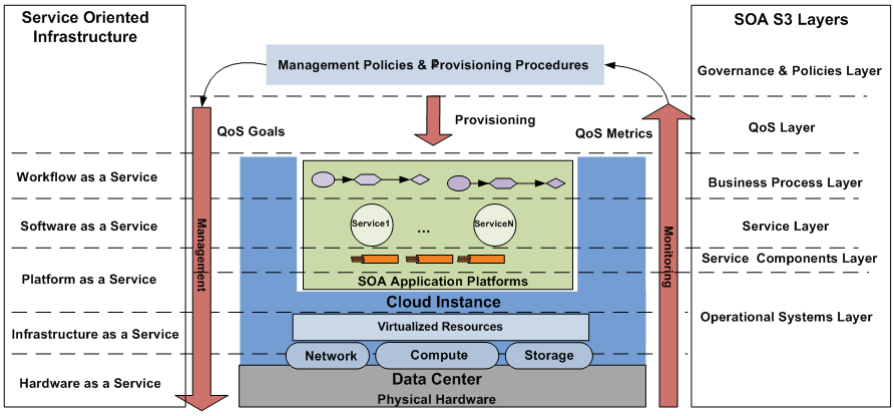

Figure 3. Architecture of the cloud and S3 model combined.

Exposed services must be fully discoverable, interoperable and easy to integrate with external provisioning frameworks required for instance to build hybrid clouds. The solution must support aspects such as:

- Cloud environment creation through the provisioning of virtualized infrastructure with allocated compute, storage resources distributed over physical servers connected with virtual network.
- The provisioned virtualized infrastructure elements can be modified during runtime so the running application services can obtain or release access to physical resources during execution.
- Particular elements of the cloud need to be constantly governed to ensure the required Quality of Services (QoS) using tools to monitor and manage the configuration of particular elements within the S3s Operational Systems layer through the specification of QoS goals (QoS Layer) for provisioned elements, definition of provisioning procedures and adaptation policies (Governance and Policies Layer) stored in a central repository.

3.1. Infrastructure as a service

The purpose of the virtualized infrastructure layer is to supply other layers with computing power, storage and basic software components that will enable them to run higher-level user applications. The main mean used to achieve this goal is the use of virtualization techniques, which will create a feeling of unlimited resources for the upper layers.

Two of the main challenges that arise when adapting the cloud computing model for hosting multiple organizations within one shared cloud are tenant isolation and quality of service assurance. Heavy virtualization, either 'bare-metal' or 'hosted' can guarantee that both of these requirements are fulfilled. However, for smaller public administration units, lightweight virtualization can be applied, in a form of operating system containers, such as OpenVZ or Solaris containers [9]. This method of virtualization can provide sufficient isolation and QoS parameters, with more fine-grained resource allocation for a given group of organizations.

The choice of hardware infrastructure that will be used for virtualization is left for the implementer to choose — virtual machines (VM) can be hosted on literally every modern hardware platform. Such platforms can be either cluster installations, with hardware virtualization support, or multiple lower end nodes connected using a fast and reliable network. When considering the second option, VM load balancing and migration can be crucial for ensuring the efficient use of hardware resources. Luckily, most commercial and open source cloud management software provides VM migration support.

To fully leverage the advantages of virtualization and to ease and the speed up deployment and configuration of applications, a sub-layer of application servers will be introduced. Such servers would be distributed in the form of Virtual Appliance — bundles containing Just Enough Operating System (JEOS) to run them using the VM hypervisor. After the deployment of a virtual appliance, a sserver would join the pool of available resources and be ready for configuration and deployment (either manual or automatic) of the proper software for the needs of given organization.

3.2. Platform as a service

PaaS exposes services on top of a given software platform (application server, ESB, Message Oriented Middleware, database) that is provided as a "service" to build high-level service. PaaS might produce a platform that contains a properly configured OS, database and application server with already-deployed application components, often serving tenant application[1], and enabling software delivery through an on-demand business model. Many application platforms like J2EE or .Net provides such support using so-called domain notion where particular application server instances can be assigned to a group of infrastructure users. Database-level multi-tenancy can be achieved by both using separate database schemas and appropriate user permissions, or — in the case of higher database throughput requirement — by physical separation of databases on two or more physical nodes.

[1]There is a distinction between multi-tenancy, isolated tenancy and mega-tenancy. Multi-tenancy uses the same instances of database or other environments with the same version of an application. Isolated tenancy devotes a dedicated and unique database to each application installation offering flexible software customization with many applications' version. Mega-tenancy is a sort of combination of the multi and isolated tenancies where execution environments are dedicated with a shared database.

SOA is a key concept of the proposed solution. Every subsystem, operation or functionality will be published as a standardized service and integrated with other services using Enterprise Service Bus software available in the PaaS layer. ESBs provide advanced message routing functionality, data exchange format mediation, and monitoring and high availability capabilities. What is also important in an environment that is opened for extension, ESB through the use of technologies such as Java Business Integration (JBI) or Open Services Gateway initiative (OSGi) supports an easy process of plugging-in new components/services into the system.

In the described use case of a system for public administration units, services hosted in the ESB could consist of:

- A business process execution service — an entry point to the business process execution engine.
- Legacy systems ESB adapters — providing access to proprietary applications in a standardized fashion.
- ePUAP services — ESB "proxied" services for pulling and push data from/to existing electronic administration system.
- Data services, catalog services — such as document storage, LDAP lookups etc.

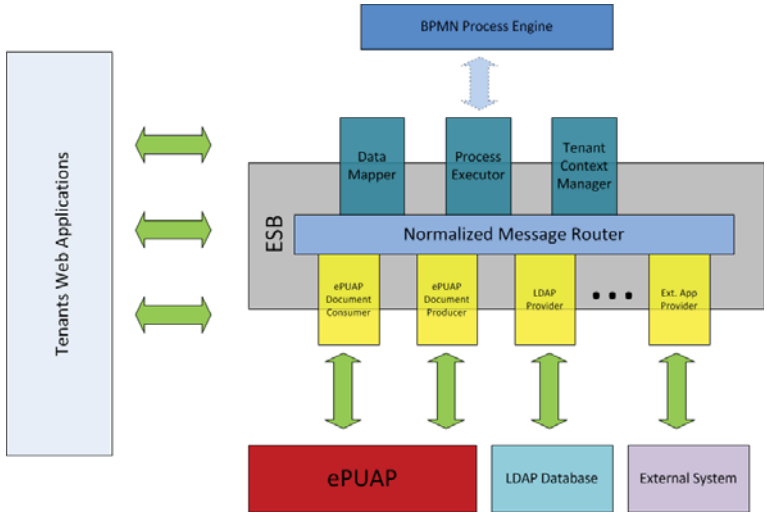

Figure 4. Message communication exchange between ESB and ePUAP portal.

3.3. Software as a service

Multiple instances of web-based applications will be the backbone of the systems front-end. Clustered (or single — for non-demanding organizations) applications will be configured to fulfill each tenant specific needs. Through the use of web interface, all system capabilities and services will be exposed to end users — i.e. public admini-

stration unit employees. ePUAP will remain the main entry point for country citizens, however, specific applications can always be made accessible to the broader public.

Some web applications will be crucial to the functioning of the system, thus they will be 'prepackaged' and deployed for every tenant. An example of such application is a BPM task console, which enables users to access tasks assigned to them by the business process execution engine. Other applications can be specifically requested by the organization and deployed 'on-demand' — legacy applications being moved to the cloud also fall under this category.

3.4. Workflow as a service

One of the key aspects of the designed system is the WaaS functionality. Business Process Modeling (BPM) tools aim at bridging the gap between business analysts and technical teams providing additional support during the phases of design and deployment of complex business processes.

A standard chosen for BPM implementation in this project is BPMN — this format, maintained by OMG Consortium, is capable of modelling workflows and storing information about the presentation of the model. Such an approach enables the use of standardized editors, workflow visualizations etc. Up to version 1.2, BPMN was responsible only for the model and presentation of the process. To execute the BPMN described process in an execution engine, an intermediate step of BPMN2BPEL mapping was needed. Resulting Business Process Execution Language files were ready to be deployed and run in existing engines coming from various vendors. With the introduction of BMPN 2.0, a standard BPEL mapping was described and made obsolete the same time — BPMN 2.0 supports native process execution, without the need for any kind of translation. Products that leverage BPMN 2.0 features already exist on the market — including a powerful, open-source business process modeling framework jBPM, maintained by JBoss.

BPM tools provide end users with the ability to setup their own workflows which will interact with other services plugged into the ESB. Web application will be able to pull information from the business process execution engine to, for example, determine tasks that are pending user approval, display them and then submit the approval information back to the engine so the process can resume.

Modeled processes will also have access to the existing ePUAP system — data submitted by the citizen using the ePUAP frontend can be transferred to the BPM engine, processed according to the workflow and then pushed back to ePUAP.

4. Implementation

As a part of the research, a simple implementation was prepared, to ensure that the vision proposed can be realized in reality with the use of currently available technologies, frameworks and software. Proof of concept (PoC) implementation covers each layer of the architecture described earlier, to assemble a basic yet working solution.

4.1. Infrastructure layer

For the purpose of PoC implementation, all resources were allocated on one physical node. Oracle VirtualBox[2] was used as VM hypervisor. Vagrant[3] software took the responsibility of managing virtual machines instances. Choice of Vagrant was dictated by the usage of only one node — actual VM migration capabilities were not required in such an environment and Vagrant is designed to create multi-VM development environments on a single physical computer. It is not suited for production use — if the designed system were to be deployed in reality, an enterprise-class solution like OpenNebula, Eucalyptus, OpenStack [6] should be preferred. Such solutions offer more sophisticated VM management, VM live migration, physical nodes load monitoring etc.

Java application servers were used as a basis to run the remaining software components of the system. JBoss[4] was chosen as AS vendor, with JBoss AS 7.1 product — this choice was dictated by the compatibility of the selected BPM solution, good clustering capabilities and community support.

With Vagrant supporting the concept of "box" — VM image with preconfigured JBoss AS and Java technology stack preinstalled was used. This made spinning up a new application server node as easy as adding a new Vagrant VM to the list and selecting an appropriate prepackaged box.

4.2. Platform as a service layer

After VM startup, a set of maintenance tasks needed to be performed in order to prepare the virtual node for actual usage. This problem was addressed by setting up Chef[5] provisioning framework to perform network and OS setup, application server configuration and startup, application deployment etc. A chef introduces the concept of "cookbooks" and "recipes" to perform administrative tasks. Cookbooks are sets of recipes — ruby scripts that perform actual operations on the provisioned nodes. Recipes are configurable with the use of attributes, which can be overridden automatically (e.g. dependent on the operating system), or manually — specified by the administrator before the VM provision takes place.

The Chef repository contains a number of recipes that deal with common tasks — such as installing desired JDK. The Chef also provides so-called "resources" — helpers that enable users to easily write their own, specialized recipes. Using attributes, resources and the Chef template system (which is able to produce configuration files of any type and structure), recipes were prepared to assign application servers to the accurate domain or to point the application code to the correct database node, assigned to the tenant.

[2] `https://www.virtualbox.org`
[3] `http://vagrantup.com`
[4] `http://www.jboss.org/jbossas`
[5] `http://wiki.opscode.com/display/chef/Home`

With Vagrant explicit support for Chef, provisioning configuration was merged with the Vagrant VM configuration. In a production environment, Chef encourages users to make use of the client-server approach — with central Chef Server storing recipes and node-specific configuration files, and Chef Client — executing configured sets of recipes on the nodes.

Listing 1: Chef recipe configuring and running JBoss as part of a server domain

```
chef.add_recipe("jboss::domain-conf")
chef.add_recipe("jboss::domain-run")
chef.json = {
  :jboss => {
    :domain_name => "mpl.umk.jboss.slave1",
      :domain_controller => {
        :address => "192.160.6.11", :port => 9999
      },
    :bind_address => "192.160.6.10"
  }
}
```

The ESB concept is the central point of SOA and the entire concept of the system. The Fuse ESB[6] product was chosen as the ESB implementation. It is based on the Apache ServiceMix[7] project, but provides additional documentation, maintenance and support to deliver an enterprise-ready, open source service bus.

Expansibility of the designed system is required — if one cloud needs to fulfill the needs of multiple organizations, tenants need to have the ability to plug their own extensions into the common architecture. Fuse ESB enables users to achieve that goal by providing support of two technologies — JBI and OSGi.

Java Business Integration is a specification developed under the Java Community Process, for an approach to implement SOA architecture based on Java technology stack. Specification is based on the Web Services and Normalized Message Router (NMR) concepts, and provides portability between any JBI implementation. Currently, however, JBI is slowly being replaced with OSGi-based platforms. OSGi, a module system and service platform for Java, offers an impressive, standardized and dynamic component model. Fuse ESB 4.x series bring OSGi support using Apache Karaf/Felix[8] as an implementation.

To compete with the JBI specification, an alternative to NMR is also required. In Fuse ESB, Apache Camel[9] integration framework is such a solution. Apache Camel is an implementation of the Enterprise Integration Patterns — a set of rules and patterns for integrating heterogeneous data sources and message formats. With powerful DSL available in multiple programming languages, multiple components capable of transforming data from most of endpoint types used in the enterprise and the con-

[6] http://fusesource.com/products/enterprise-servicemix
[7] http://servicemix.apache.org
[8] http://karaf.apache.org
[9] http://camel.apache.org

cept of "Exchange" — a normalized message container — Apache Camel can serve as a modern NMR replacement.

Components realized as part of PoC implementation leverage OSGi and Camel to create OSGi bundles that can be easily and dynamically plugged into the ESB. An OSGi bundle serving as ePUAP integration layer was created, that allowed the platform to receive and send documents to/from ePUAP, appropriately map the document data to business process data and finally start a process instance. Web service consumers and producers were created using Apache CXF[10] framework — which is also integrated into Camel. This made creating a simple WS component as simple as writing one short XML configuration file. ePUAP integration layer takes care of the appropriate certificate signing of the outgoing requests, and the verification of requests incoming from ePUAP — all other system components do not have to take security mechanisms required by ePUAP into account.

4.3. Software as a service layer

The SaaS is the least standardized layer of the system architecture stack. Any kind of Java-based web application can be deployed on a standalone or clustered JBoss server. For the purpose of PoC implementation, two instances of a user task web console were deployed, serving two different organizations. The application itself is provided alongside with jBPM[11] solution stack — it a simple example of software accessing the business process execution engine and the business process repository to execute previously defined processed. Console is also capable of handling human tasks assigned to the users in the system. They can be displayed and completed by the user, thanks to the Freemarker Template Language (FTL)[12] templates generated by the jBPM software.

4.4. Workflow as a service layer

jBPM was chosen as the provider of the BPMN modeling suite. It does not only provide the process execution engine, but also the complete set of tools accompanying it. These additional components include BPMN graphical editor (distributed as Eclipse IDE plugin or a standalone web application), Guvnor process repository, jBPM task console and Human Task Service for managing user tasks.

The business process execution engine itself is a lightweight, pure Java component that can be either embedded into the application that needs to start a process execution, or for example run as a central service, exposing the jBPM engine functionality. The bundled jBPM console makes use of the embedded engine to execute processes locally. While such an approach can work in a small scale deployment scenario, the 'as a service' approach fits better into the SOA-based environment and provides more

[10] `http://cxf.apache.org`
[11] `http://www.jboss.org/jbpm`
[12] `http://freemarker.sourceforge.net/docs/dgui_quickstart_template.html`

control over state persistence, load balancing and availability. To leverage these advantages, an OSGi bundle was created, which after plugging into the ESB served as a service oriented business process execution engine.

Second most the important component of the jBPM suite is the Guvnor repository (Figure 5), a web application that brings rich collaboration capabilities into the world of business process modeling. Guvnor serves as a versioned storage of business processes and related assets.

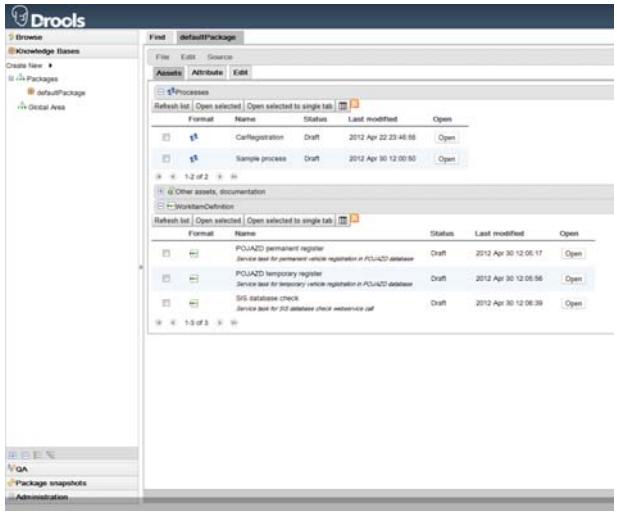

Figure 5. Guvnor process repository console.

jBPM introduces a concept of "Service Task" — a business specific task that can be used when creating BMPN processes. Creation and sharing of Service Tasks through Guvnor makes it easy for different organizations and vendors to plug in their proprietary services into the cloud. All that is required is to create an appropriate Service Task and OSGi bundle pair. The first can be deployed to Guvnor and the latter can be plugged into the ESB and provide connectivity with the system. Technical teams and business analysts working on the processes or service tasks are able to check in them to Guvnor and then momentarily use them in all of the applications, with the use of Guvnor REST API. Guvnor does also provide a web BPMN editor, user roles management and documentation generator functionality. Human Task Service is a sample jBPM implementation of a service that manages the state of human activities, according to the WS-HumanTask [2] specification. jBPM supports modeling of human tasks with the use of special User Task nodes. User Task nodes can receive or send process data to the user using FTL form templates — a simplified HTML form language. Human Task Service is used to manage and persist assignment and state information.

In a multi-tenant environment, multiple instances of Human Task services can be used to manage tasks inside each organization separately, while ensuring adequate isolation and security.

5. Case study

For the PoC implementation purpose, the Municipality of Cracow provided examples of actual business processes concerning new vehicle registration. A single variant of the process was chosen as the basis for a brief case study.

5.1. Deployment model

To demonstrate multi-tenant support, described vehicle registration process was setup for two distinct organizations, the Municipalities of Cracow and Tarnow. the Cracow Municipality is considered to generate more load than the Tarnow office and will be used to present clustering capabilities on the running virtual machines. Figure 6 presents the cloud structure setup for these two tenants.

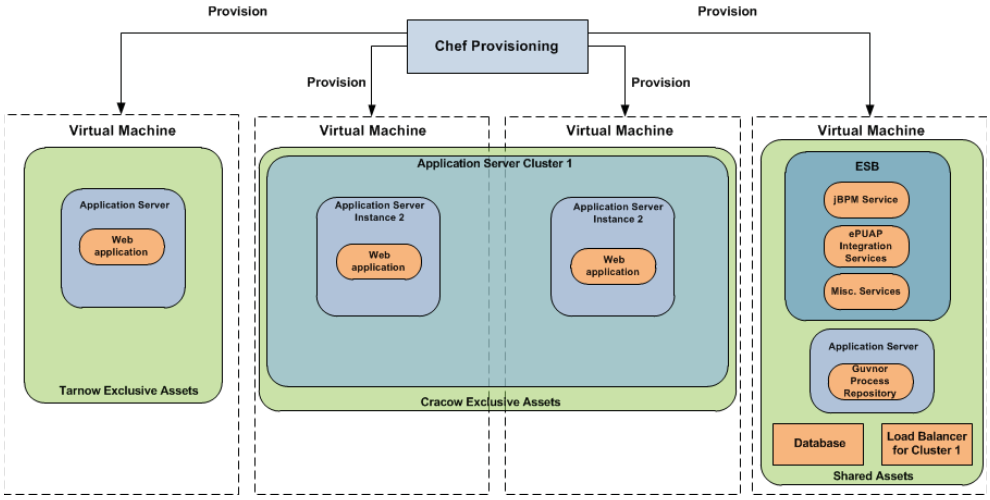

Figure 6. PoC cloud deployment structure.

Central point of the system, Fuse ESB is deployed on a separate virtual machine; the bus itself however is shared amongst both organizations — providing access to business process execution engine amongst other services. Another shared component, Guvnor process repository, is deployed alongside the service bus and enables both organizations to collaborate and reuse resources in their workflows. Data access layer is realized based on single MySQL database with tenant schema separation.

Municipality of Cracow serves as the example of an organization that needs to handle excessive traffic — hence load balancing and clustering is required. BPM

task console is clustered using two virtual machines and JBoss application server clustering capabilities. Municipality of Tarnow task console is deployed in a standard, standalone fashion. Both organizations have access to and communicate with their respective accounts in the ePUAP test environment.

5.2. Process model

The process description delivered in the form of a process diagram (prepared in a proprietary format) showed clearly that the workflow is completely manual and is not taking advantage of the digitalization possibilities. Citizens were obliged to fill in a standardized paper form and deliver it to the office. During workflow next phases, administration employees performed manual validation of the documents, prepared additional documents to be stored in the Municipality office and contacted with external IT systems to manually register the vehicle or the check if the vehicle has not been stolen in the past.

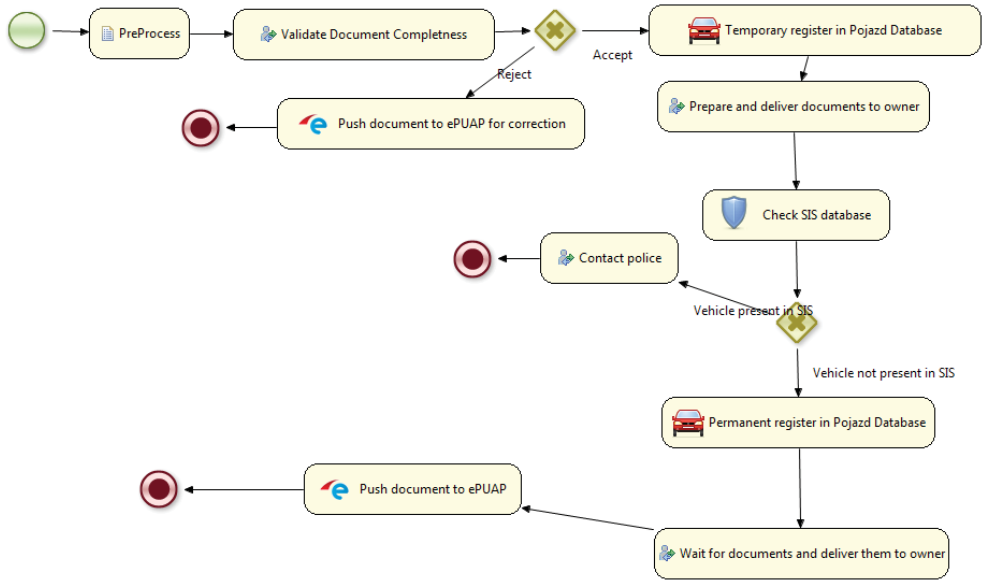

Figure 7. Vehicle registration BPMN process diagram, created in jBPM suite.

The move of the process to cloud environment started with moving the process definition to the standard BPMN format, using Eclipse jBPM designer plugin (Figure 7). As ePUAP is acting as a front-end to the Municipality of Cracow system (Figure 8), the data is pushed from ePUAP to the BPM engine, initiating the process execution (Figure 9). ePUAP communication service tasks were already prepared, so that they could be used by process designers system-wide. Data received from ePUAP is mapped to process variables and used to create a user task that aims at data validation. FTL

templates are used to present the data to the Municipality of Cracow employee. After employee action is taken, in the case of positive validation result, the vehicle is automatically registered in the country-wide POJAZD database — the automation of this step is one of the advantages of process digitalization. If document validation fails, data is pushed back to ePUAP with an appropriate note that will help the citizen to correct the document.

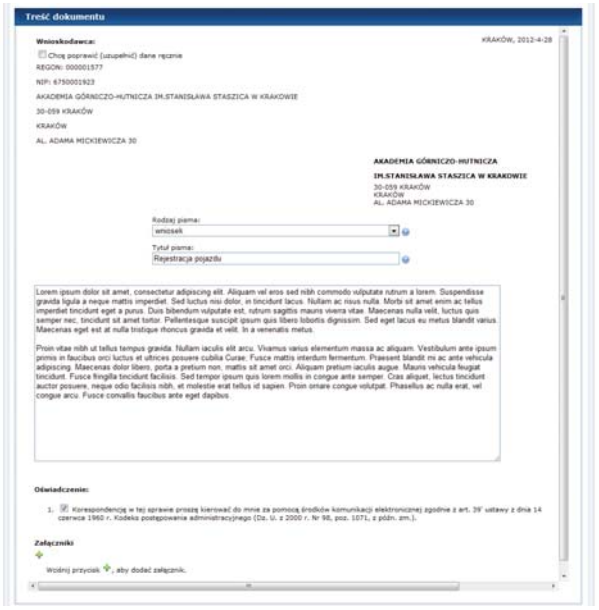

Figure 8. An example of ePUAP user form.

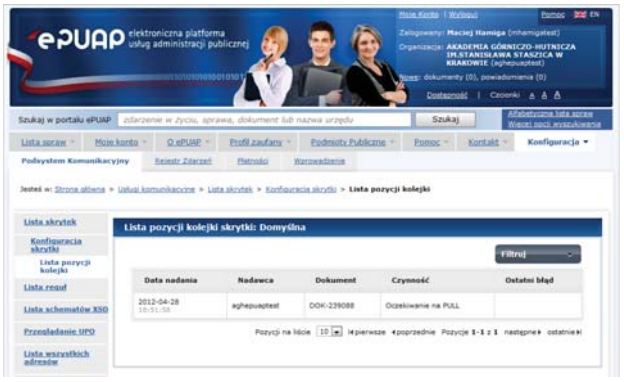

Figure 9. Administrative view of documents queue, with document waiting to be pulled by the external system.

Interaction with POJAZD database is achieved through dedicated Service Tasks and OSGi bundles — which again, can be reused by other processes. Communication with Police SIS database of stolen vehicles is realized in an identical manner and can be partially automated as well. After successful completion of the process execution, state of the request is updated in the ePUAP system.

6. Summary

Concept presented in this paper is an attempt to free public administration units country-wide of the burden of maintaining their own IT systems. Through the use of modern architecture paradigms, such as SOA and Business Process Modeling tools, a complex system was designed; open for extension and tenant collaboration.

Leveraging private cloud computing concept and building the system on top of virtualized infrastructure provided scalability and availability almost 'out-of-the-box'. Introduction of standardized packaging in the form of virtual appliances and additional provisioning software made the infrastructure adaptive to change without effort. Realization of PoC implementation based on an actual business process provided by the Municipality of Cracow proved that creation of production-ready system according to the proposed guidelines is feasible and can bring significant advantages — such as automation of certain tasks that today are being executed by public administration employees.

Further work on the discussed topic might include deployment of a sample implementation on production-grade hardware in a dynamically changing environment — in order to execute tests of performance, scaling and reconfiguration (including virtual machines migration between physical nodes). The concept of mixing two levels of virtualization (VM-based and container-based) can also be researched in greater detail to verify whether such approach would bring real value into the system architecture.

Acknowledgements

The presented research was supported by funding from the European Regional Development Fund no. POKL.04.01.01-00-367/08-00.

References

[1] Us government cloud platform for federal agencies. http://apps.gov.

[2] WS-HumanTask. `http://incubator.apache.org/hise/WS-HumanTask\_v1.pdf`, 2007.

[3] Arsanjani A., Liang-Jie Z., Ellis M., Allam A., Channabasavaiah K.: *S3: A Service-Oriented Reference Architecture.* IT Professional, 2007.

[4] Buyya R., Broberg J., Goscinski A.: *Cloud Computing — Principles and Paradigms.* Wiley, 2011.

[5] Cellary W., Strykowski S.: *E-Government Based on Cloud Computing and Service-Oriented Architecture*. ICEGOV, 2009.

[6] Delgado V.: *Exploring the limits of cloud computing*. Master's thesis, Universitat Politecnica de Catalunya, 2010.

[7] Elbadawi I.: *Cloud Computing for E-Government in UAE: Opportunities, Challenges and Service Models*. Proc. of ICEGOV 2011, Albany, NY, USA, ACM, 2011.

[8] M.Chang., He J., Castro-Leon E.: *Service-Orientation in the Computing Infrastructure*. Proc. of SOSE 2006, Los Angeles, CA, USA, 2006.

[9] Vaughan-Nichols S. J.: *New approach to virtualization is a lightweight*. Computer, vol. 39, IEEE, 2006.

[10] Zieliński K., Szydło T., Szymacha R., Kosiński J., Kosińska J., Jarzab M.: *Adaptive SOA Solution Stack*, vol. 5. IEEE Transactions on Services Computing, 2011.

Affiliations

Maciej Hamiga
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, maciej.hamiga@gmail.com

Marcin Jarząb
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, mj@agh.edu.pl

Received: 30.04.2012
Revised: 26.06.2012
Accepted: 3.09.2012