Implementation of the Application Layer in the Business Process Management System

Josip Lorincz, Anamarija Talaja, Dinko Begusic
FESB, University of Split, Croatia, e-mail: josip.lerinc@fesb.hr , atalaj00@fesb.hr, dinko.begusic@fesb.hr

***Abstract*— The paper deals with implementation of the application layer in the business process management (BPM) system of the telecom operator. Using the concept of service-oriented architecture (SOA), it is possible to upgrade a process management system from the two-layer architecture to the three-layer architecture. In the three-layer architecture, presentation and data layers are separated by a business logic layer. The business logic layer contains application programming interfaces (APIs) that communicates according to representational state transfer (REST) protocol rules. In this paper, developed process management tool uses APIs which are programmed by means of Spring Boot technology in Java programming language. Implementation of SOA based on APIs is shown on a client that is programmed in Angular technology using TypeScript language. The layer between the client and the database contain APIs whose basic function is to fetch required data from an Oracle database of a telecom operator. The approach proposed in this paper allows upgrading of the existing two-layer to the three-layer SOA of the software solution dedicated to the management of telecom operator business processes.**

Keywords – application layer, SOA, REST protocol, API, Spring Boot, Angular

I. INTRODUCTION

Business process management (BPM) is a fundamental monitoring approach for business processes. Supervision of business processes provides insight into unfinished business processes, which is essential for analysing the performance of the system. Before the existence of operations support systems (OSS), most of telecommunications business support jobs were handled manually. Soon there was a need for a computer usage, which enables business optimization of telecom operators, and thus the concept of OSS systems is created. In addition to the OSS system, there is a system called business support system (BSS), which refers to the business systems that deal with users and related sale processes. In the past, OSS and BSS systems of the telecom operator were clearly separated, but the complexity of the network led to their mutual integration into the BSS/OSS or B/OSS form. The OSS and BSS systems must work together to provide services for the customer. Offering service to the customer today is based on negotiation between demands of commercial products controlled by the BSS system and the ability of the OSS system to deliver requested service.

In this paper, a three-layer architecture which allows monitoring of a telecom operator business processes is developed. Using the concept of service-oriented architecture (SOA), business logic layer is implemented between the presentation and data layer as a middle layer. Through application programming interfaces (APIs), which are implemented on the business logic layer, the user of the system can have better control of business processes. For the development of APIs, the REST protocol is used and resources are made in the JavaScript Object Notation (JSON) format. The business logic layer is programmed in the Spring Boot technology and client is programmed in the Angular technology. Data resources are used from the Oracle database system of a telecom operator. Data format is based on a Representational State Transfer Protocol – JavaScript Object Notation (REST-JSON). Hence, in this paper, an approach allowing the upgrade of the two-layer to the three-layer SOA for the software solution dedicated to the management of telecom operator business processes is proposed.

The structure of this paper is as follows: Section II provides a description of the related works. In Section III, the description of the basic SOA concept with its main entities and layered architecture are presented. In the next Section IV, REST protocol whose rules are used for regulating APIs communication are described. Program structure of the middle layer and functions of implemented APIs, including program structure of client and application model are introduced in Section V. Finally, some concluding remarks are given in Section VI.

II. RELATED WORK

In traditional systems, business functions are hardcoded in the applications. Using the SOA concept, a function is coded only once and then it can be reused many times. Authors in [1] combined BPM and SOA, which generates greater efficiency in the management of information systems. They proposed BPM-SOA workflow model which consist of the six main steps:

1. Setting the BPM
2. Setting the hypertext transfer protocol (HTTP) API(s) for business process (BP) actions
3. Setting RESTFull resources
4. Setting RESTFull services
5. Implementing RESTFull services
6. Service unit testing

Setting the BPM and setting the HTTP API from BP actions are the main steps that enable SOA and BPM integration. Authors in [2] also suggest the unification of BPM and SOA. They propose a layered architecture of common SOA and

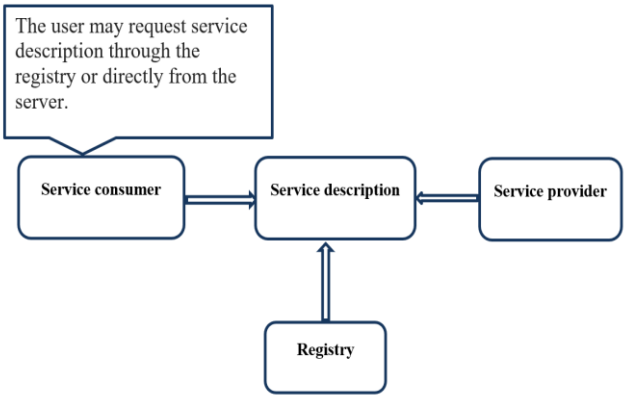

Fig. 1. SOA interaction model

BPM layers. Transformation of the traditional OSS by using SOA and BPM is presented in [3]. The proposed architecture is based on web service which wraps components to service, such as authentication, authorization, accounting, service activation and user management. It uses Enterprise Service Bus to call inner services and outside services that belongs to other applications. The most important thing is BPM, dedicated to the development and management of the business tasks by combining services with processes.

Authors in [4] introduce an architecture which combines SOA and BPM. By combining SOA and BPM, tasks in the BPM corresponds to the services in SOA. Authors in [5] provide an approach for the interaction steps between process and service modelling by means of the concept of meta-model written in Meta Object Facility (MOF) language. They also implement a software prototype that is supported by the proposed meta-model. Authors in [6] present a proposal for integrating meta-model with BPM and a meta-model for components defined with Service Component Architecture. Evaluation and categorization of five Java-based BPM systems for integration with web services is done in [7]. It is shown that the use of the domain-specific business activities is the most sophisticated technique for integrating services.

The work in [8] suggests a public e-procurement (PeP) system which is organized in the three-layer architecture: data layer, business logic layer and presentation layer. However, according to our knowledge, this is the first work which presents the SOA concept based on the three-layer architecture (data layer, middle layer and client) implemented for BPM of a telecom operator. Novelty is that a developed and proposed solution in this paper is based on the RESTfull services which are using data from a telecom operator database. Additionally, the REST API is using data of process engine that controls the execution of telecom operator business processes.

III. SERVICE ORIENTED ARCHITECTURE

The SOA is an architectural style that uses methods and technologies that enable dynamic connectivity and communications of software from various business partners and platforms by offering services in a reliable way. This

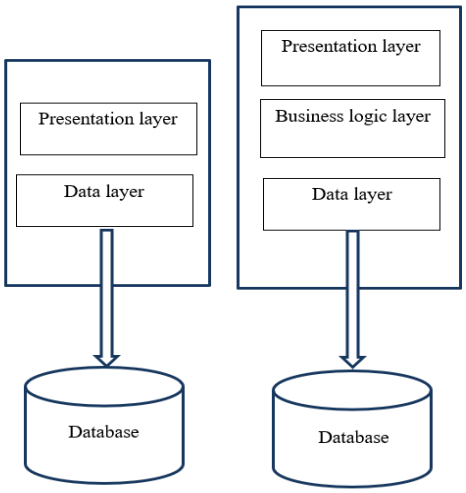

Fig. 2. Comparison of the two layer and three layer architecture

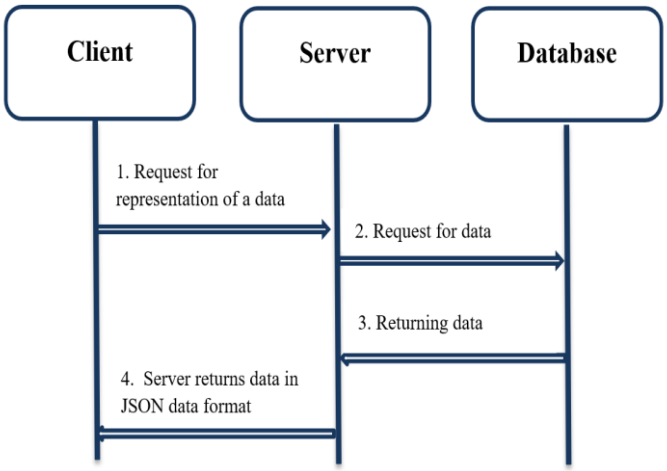

Fig. 3. Example of REST concept

approach allows advanced applications and information systems implementation [9]. It defines the interaction model based on service description between the main functional units: service provider, service consumer and registry (Fig. 1). As shown in Fig.1, functionalities of the main units of the proposed SOA architecture are:

- **Service description**

An entity containing detail description of the services which must be offered by the developed software solution based on SOA.

- **Service consumer**

An entity searching for a service to perform the required function. It can be an application or some form of the software module. The location of the service can be found through the registry, or if the location is already known, direct communication between user and provider is possible.

- **Service provider**

An entity that accepts and executes requests from the user. Provides description and implementation of offered services.

- **Registry**

The main function is to store description and location of provider's services.

The SOA concept provides an implementation of the middle layer between the presentation and data layer. Through a two-layer architecture, the user has a direct connection with a database. The two-layer architecture is used in a less complex software systems, but it is also applicable in the advanced systems to provide certain functionalities. The main drawbacks of the two-layer architecture is a short system life cycle, lack of advanced APIs and poor isolation of programming code from the user. For that reason, a commonly used application development model is based on the three-layer architecture that supports an additional layer between users and the database. This layer is known as the business logic layer (Fig. 2), which allows isolation between users, code and shared logic between different user's deployments.

IV. REST PROTOCOL

The REST protocol is an architectural style for developing web services. It is popular due to its simplicity and the fact that is built upon existing systems and features of the Internet's HTTP protocol [10]. Development based on the REST protocol approach is opposed to creating new standards, frameworks and technologies, in order to achieve objectives of software solutions based on the SOA concept.

Example of the REST concept is visualized in Fig. 3. As shown in Fig. 3, the server contains data where the client may request to display the status of that data. For example, if a special source of data is stored in the database, displaying the status of these data can represent a simple list of values from the database. The client does not know where the server has stored the data. The server could store data in the Oracle database or in some file, which is not essential to the client. The client is interested only in displaying the information provided by the server. For displaying data, JSON data format is often used. When a client wants to change some of the resource content, server gives a resource view. After the client changes resource, it sends resource back to the server which adjusts changed content with a new view.

V. PROGRAM SOLUTION

Program solution proposed in this paper is based on the software implementation of the middle layer between the client and the database. The three APIs were implemented on the middle layer. Resources of the APIs are visible on the client side.

Functional specification of these three APIs are:

1. **API for the retrieval of all process definitions**
   Program logic for getting all the process definitions from the database. Based on a defined uniform resource locator (URL), resources from the database can be accessed.
2. **API for retrieval of all process instances**
   The function of fetching all process instances by a query that receives three parameters: identification (ID) of the process definition, the start date and the completion date of the process instance execution. Resources can be accessed by URL.
3. **API for getting the execution steps of a particular process instance**
   It has the function of fetching the entire hierarchical structure of a particular process instance from the database. Query receives ID of exact process instance.

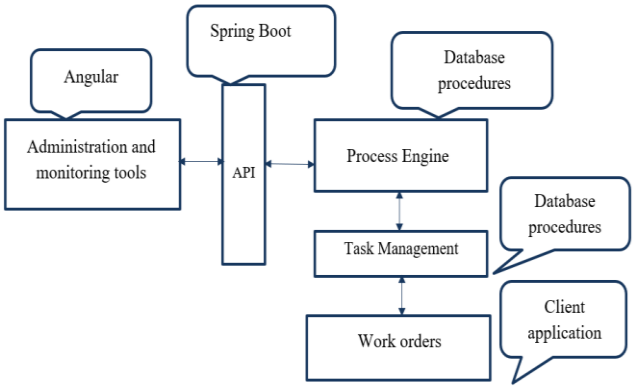

Fig. 4. Implementation of the middle layer for a real process management system

A. Implementation of a Middle Layer in Existing Architecture

Upgrading the existing two-layer architecture to the three layer architecture was applied to the real process management software system. In Fig. 4, implementation of the middle layer for the real software application enabling telecom operator process management is presented. Upgrading is achieved with the implementation of the middle layer between the client and database (Fig. 4). To achieve this, APIs are implemented in the middle layer. By means of APIs, the efficiency of the software system and its data is significantly improved. An overview of all business processes allow BPM system users to prioritize the performance of the individual business processes. Through APIs, system user has insight into unexecuted business processes that could endanger the completion of the business process in the given time frame. As shown in Fig. 4, the new architecture is based on the implemented layer between *Administration and monitoring tools* and database of the *Process Engine*. The basic elements of architecture are:

- **Administration and monitoring tools**

Administration and monitoring tools enable monitoring of business processes. Existing administration and monitoring tool is replaced by the client who provides better insight into the execution of business processes. The client is programmed in Angular technology.

- **Middle layer**

Middle layer implementation is based on APIs. APIs have a direct connection to the *Process Engine* database which enables full separation of data and presentation layer (Fig.4.).

- **Process Engine**

Represents the core of the BPM system. It is developed in Oracle's Procedural Language/Structured Query Language (PL/SQL). The basic function of the *Process Engine* is to control the execution of the process from the start to the end point. In order to enable operation of the *Process Engine*, descriptive tables, functions and procedures are required. The basic solution that is implemented in practice was provided by information system based on a *process model*. The process model allows mapping of the operational business processes, their management and execution.

- **Task Management**

Main function of the *Task management* is control of task execution. Control is performed by assigning a task to the particular execution group and then tracking the execution time of a particular task.

- **Work orders**

The *Work order* is made according to the description of the technical details from the catalogue containing different working activities of telecom operator staff. The *Work order* is made in the form of a document containing all the essential information required by the staff which will operationally execute work order in the field.

Monitoring statistic of the process execution provides the basis for determining key performance indicators (KPIs). A KPI is a measurable value that demonstrates how effectively a company is achieving key business objectives [11]. Through the developed system, monitoring of the most important KPIs is enabled. An example of a KPI, implemented in existing system, is an average total duration of a business process, where shorter time indicates the more desirable value of the KPI. Second KPI indicator, implemented in existing system, belongs to volume metric value, and it is measuring the total number of realized requests for the specific services. The higher number of realized requests, the more desirable the KPI value is. Last implemented KPI indicator is measurement of working time on specific tasks and work orders. Shorter average working time means the more desirable KPI value.

B. Middle Layer Program Structure

The APIs methods have been developed in the IntelliJ IDEA programming environment. APIs are programmed in Spring Boot technology that is based on the Java oriented framework Spring. Spring Boot is selected since it enables creating stand-alone, production-grade Spring based applications and it is suitable for connection with third-party libraries. Following the REST concept and corresponding rules, each API is a separated web method that has its own functionality and independence.

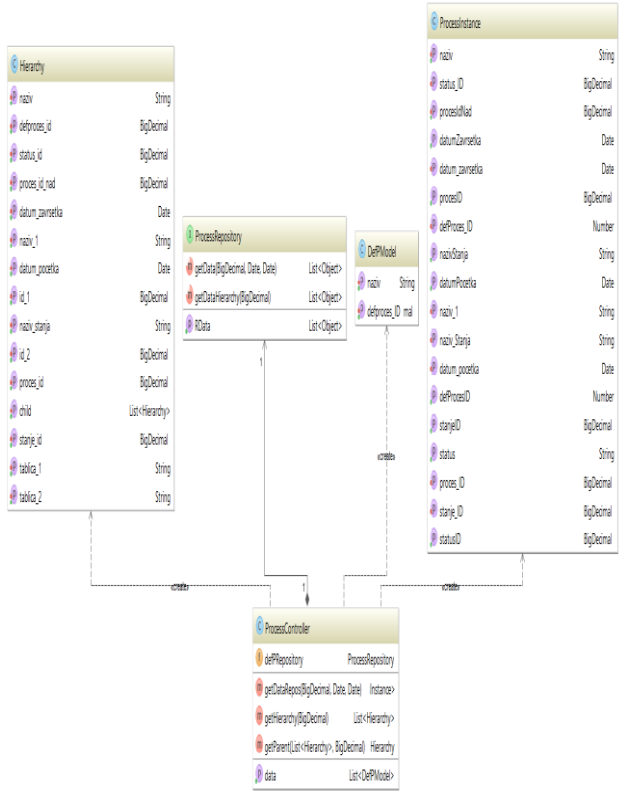

Fig. 5. Program structure of the middle layer

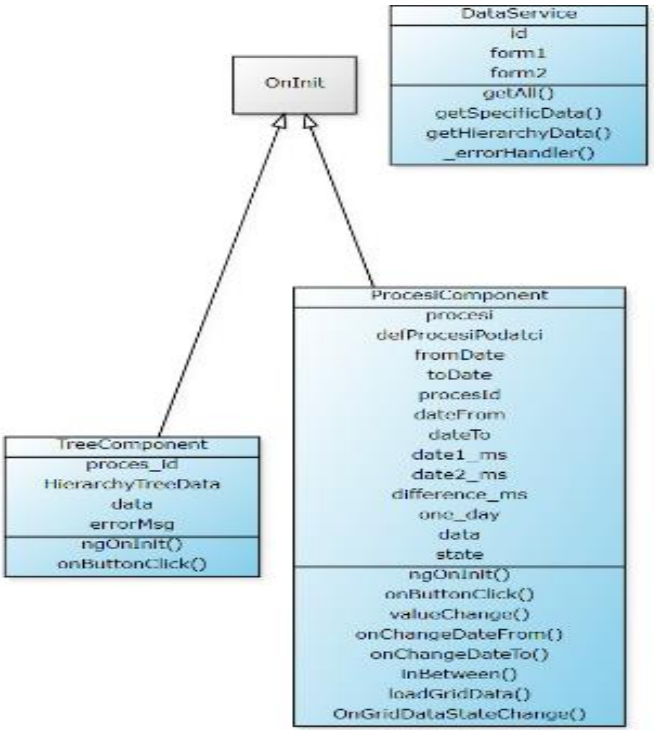

Fig. 6. Program structure of the client application

Program structure of developed middle layer is presented in Fig. 5. The programming structure consists of three basic units:

Fig. 7. Table view of all process instances (in Croatian)

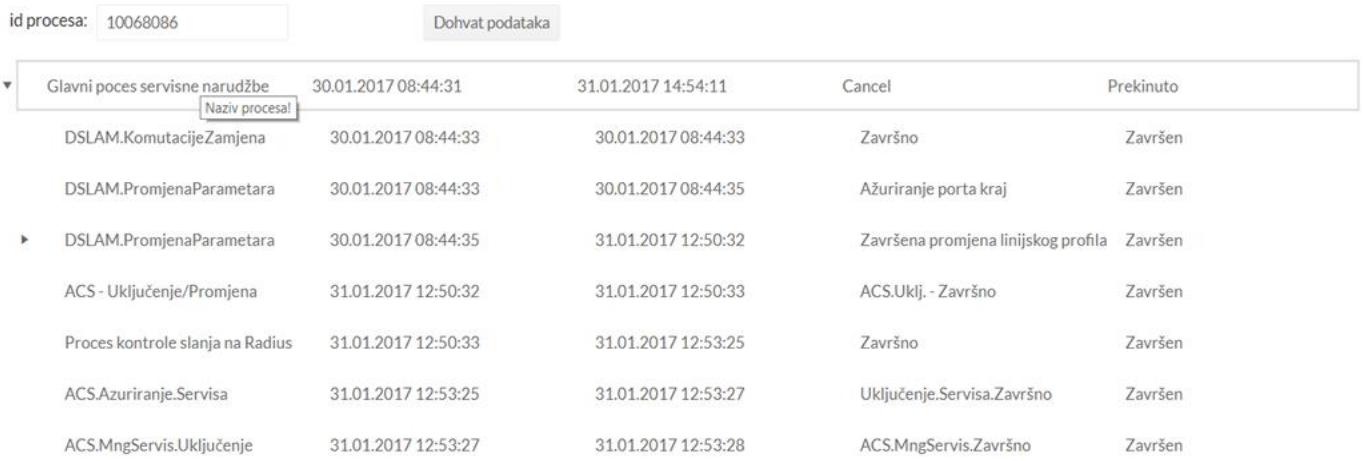

Fig. 8. Steps to execute a certain process instance (in Croatian)

- **Controller**

Contains API's program logic. The APIs class is flagged by the @RestController annotation, which means that API behaves according to the REST protocol rules

- **Model**

It contains the attributes in which the columns from the database tables are mapped. Each row of the database tables represents an object that is the type of model class.

- **Repository**

Represents database interface. Includes database queries with related methods. Each query is written in the Structured Query Language (SQL). Repository methods are invoked in the controller.

As shown in Fig. 5, the controller (*ProcessController*) represents the focus of the entire programming architecture. This class controller generates and uses object instances. There are three model classes, one repository and controller. Model classes and repository extends all controller methods. Within the controller, there are methods that call for the repository methods (*defPRepository*). A method that is defined in the controller class contains the URL through which API resources will be available. Repository (*ProcessRepository*) contains methods that retrieve all process definitions, process instances and steps of executing particular process instance based on a defined query.

A. Client Program Structure

The client application is developed in the Visual Studio Code programming environment using Angular technology. Angular enables component based architecture that provides a higher quality of code and it is easy for use. The program structure of the client application is presented in Fig. 6. It is structured into the two components (*TreeComponent* and *ProcessComponent*) and one service (*DataService*). Within the service, there are API's URL. Data requirements are realized through the HTTP protocol. Both components can access the service. Program code is written in a TypeScript program language. The TypeScript is a strict syntactical superset of JavaScript language. As an open source programming language it is designed for development of large applications and transcompiles to JavaScript.

As shown in Fig. 6, *OnInit()* method allows loading two components when launching an application. Both components are separated and contain attributes and methods that are

mutually independent. Each component can be used as a separate block unit in any client environment.

The client contains Kendo user interface (UI) controls which allow user better review of API resources. Kendo UI controls can be taken from Telerik Progress web site. Telerik is a company offering software tools for web, mobile and desktop application development and subscription services for cross-platform application development [12]. Kendo UI controls are implemented in Hypertext Markup Language (HTML) component's file and edited in Cascading Style Sheets (CSS) file. All the process definitions are implemented in a drop-down list which represents Kendo UI control. Process instances are shown in the table view presented in Fig. 7. Execution steps of the particular process instance are shown in the tree view which also represents Kendo UI control. The tree view control presented in Fig. 8 is organized in the hierarchical data structure. Each component represents one application view.

DataService presented in Fig. 6 is a separate entity. It is independent of any component. Within its class, there are three methods: *getAll(), getSpecificData()* and *getHierarchyData()*. Each of these methods belongs to one API whose program logic is implemented on the middle layer calls of the business logic layer. Within the method, an HTTP request is executed where unique URLs of each API are implemented. The instance of the *DataService* class is implemented within the component's constructor and can thus access the resources of every API by its URL.

A. Application Model

As shown in Fig. 7, the user gets details about all process instances for selected process definition. Through table view, user gets process instance ID, the name of process definition, start and end date of its execution, name of state and status which provides information on whether the process instance is finished.

Fig. 8 shows the hierarchical structure of each process instance. The user has insight into the steps of execution of the process instances as well as the ability to see the steps preceding its execution. According to Fig. 8, the execution status of each process is clearly indicated for every business process scheduled for realization.

VI. CONCLUSION

By using the SOA concept, software products from various business partners and platforms can be dynamically connected. Systems that fully implement the SOA concept are characterized by the rapid advancement of the business process and the quick adaptation of the information system to new changes and requirements. This approach makes the system agile.

In this paper, the two-layer software architecture of a telecom operator workflow process management has been upgraded to the three-layer SOA. The main advantage of the proposed solution is separation of the client side and database side of BPM system and the use of the APIs as the logic holder.

By means of APIs that are programmed according to the REST protocol rules, the user of the BPM software system can access any required resource. All the logic required for fetching data from the database is placed in the APIs. Client's function is only to show resources in the required form. Through the client, the user of the process management solution has insight into all process instances for the selected process definition and can follow the steps of executing an individual process instance.

REFERENCES

[1] Octavian Dospinescu, Catalin Strimbei, Roxana Strainu and Alexandra Nistor. “REST SOA orchestration and BPM platforms”, In *Informatica Economică* vol.21, no. 1/2017, pp.: 30-42., https://bit.ly/2Klkpuz

[2] Imran Sarwar Bajwa, Rafaqut Kazmi, Shahzad Mumtaz, M. Abbas Choudhary and M. Shahid Naweed, “SOA and BPM partnership: A paradigm for Dynamic and Flexible Process and I. T. Management”, In *World Academy of Science, Engineering and Technology, International Journal of Economics and Management Engineering* vol:2, No:9, 2008, pp.: 990-996., https://bit.ly/2KmFzbM

[3] Jian Tang,Haitao Qu,Qian Wang,Xiaoxiang Luo,Renjie Pi, “Transform traditional OSS by using SOA and BPM”, In *2008 Third International Conference on Pervasive Computing and Applications*, 2008,Vol.2, pp.: 999-1002

[4] Fuhua Ge,Shaowen Yao, “Architecture combining SOA and BPM”, In *2011 International Conference on Computer Science and Service System (CSSS)*, 2011, pp.:2124-2127

[5] Bazán Patricia,Gabriela Perez,Roxana Giandini,Javier Diaz,” Process-Service Interactions Using a SOA-BPM-Based Methodology”, In *2011 30th International Conference of the Chilean Computer Science Society*, 2011,pp.: 100-107

[6] Patricia Bazán,Gabriela Perez,Roxana Giandini,Elsa Estevez ,Javier Diaz “Services conceptualization within SOA/BPM methodology”,In *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, 2012, pp.: 1-10

[7] Markus Doedt,Bernhard Steffen,” An Evaluation of Service Integration Approaches of Business Process Management Systems”,In *2012 35th Annual IEEE Software Engineering Workshop*, 2012, pp.: 158-167

[8] Vjekoslav Bobar, Ksenija Mandic, “Design and implementation of Software Architecture for Public e-procurement System in Serbia”, In *International Conference on Information Society and Technology (ICIST)*, 2014, Vol.2, Poster papers, pp.: 338-343, https://bit.ly/2Md955w

[9] G. Selda, “Service Oriented Architecture“, Internet, 4 August 2005 https://bit.ly/2OVrp4Y

[10] L. Keneth, “The little book on REST services “, Internet, 2016, https://bit.ly/2vvuWih

[11] Internet, August 2018, https://bit.ly/29tAwDO

[12] Wikipedia (https://en.wikipedia.org/wiki/Telerik), July 2018.