Computer Science and Information Systems 16(1):45–74 https://doi.org/10.2298/CSIS171005010M

Building a BPM Application in an SOA-Based Legacy Environment

Mladen Matejaš[1] and Krešimir Fertalj[2]

[1] Privredna banka Zagreb,
Radnička cesta 50, 10000 Zagreb, Croatia
mmatejas@gmail.com; mladen.matejas@pbz.hr
[2] Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia,
e-mail: kresimir.fertalj@fer.hr

Abstract. Modern organizations need to understand and constantly improve their business processes (BPs) in order to make successful business decisions. This paper describes an integration model for building a Business Process Management Application (BPMA) and connecting the BPMA with legacy systems based on Service-Oriented Architecture (SOA). A BPMA is an application developed to support a BP performed by legacy application/s. A combination of multiple BPMAs provides support for multiple BPs and forms a BPM solution. The presented model is characterized by a simple co-dependence of the BPMA and the existing systems, minimal changes to the legacy applications and a maximal utilization of the existing functionalities. It enables the existing applications to function independently from the BPMA and simplifies the business data used in the BPMA. An extensive evaluation of the model was undertaken by experts from the BPM area. Its feasibility is demonstrated on a real-life business use case scenario.

Keywords: BPM, SOA, business process, integration model, business layers.

1. Introduction

In the volatile modern economy, the relevance of information technology and business process modeling is constantly growing [1]. By closely observing the business environment, one can identify BPs as its core elements. A BP can be defined as a sequential flow of business activities (BA) over a certain period [2]. The performance of an organization depends upon BPs which provide the means for achieving the organization’s fundamental objectives [3]. Furthermore, the process view of BPs provides a better understanding of the organization and facilitates the development of information systems that would successfully support these BPs.

Business Process Management (BPM) is a set of management disciplines whose goal is to continuously optimize the efficiency of processes and activities through automation [4]. BPM provides support for the efficient management of a business environment with the purpose of increasing its flexibility and productivity. BPM implementation is a process of building a BPM solution in an existing business environment. The main objective of such a project is to create an environment, with the help of BPM tools, where the flow of processes is determined in real time by events and the results of the actual process execution. A set of BPM tools (sometimes called a BPM suite) is a software product designed to develop, maintain and optimize a composite process application [5].

Although a BPM solution has its benefits, several questions arise: How many resources need to be spent for its implementation? What changes need to be made in the existing legacy environment? Is it necessary to develop new applications or services to support the BPM solution? To properly answer these questions, it is necessary to analyze the amount of user activities the BPM solution needs to support. Functionalities that support user activities can be determined by reviewing the needs of the business owners as they have a clear understanding of how the entire business system should function and can therefore roughly define the business requirements. Business owners allocate resources, approve the funding, decide on the scope of the projects and that way create additional business value. However, extensive redesigns and upgrades of existing information systems are tasks that business owners do not approve lightly. While such improvements can provide stability to the existing information systems, from a business owner’s perspective, they do not create a new product or service, nor do they attract additional customers.

Integrating BPM in legacy environments, where BPs are managed by existing applications, is a challenging task. Some BPM implementation options include rewriting legacy applications using a BPM suite and building a custom BPM solution within the existing environment. However, redesigning and totally rewriting the existing processes with BPM tools, or building new applications, would be an enormous effort and would require a large amount of resources and time. Also, the organization’s ability to grow, respond to market demands and cope with ongoing challenges would be limited during the redesign period. Consequently, business owners hesitate in approving such a task.

In case of SOA-based legacy environments, i.e. environments comprising legacy applications based on an SOA architectural style, a solid service background is present. Solid service backgrounds enable the flexibility of legacy applications and constitute a good foundation for the endeavor of BPM integration.

The goal of this research is to develop an integration model which would enable BPM implementation by identifying business activities in existing SOA-based applications and reusing them in a newly created BPMA. The usage of existing business activities, supported by the functionalities of the legacy applications, removes the need for extensive redesigns of the existing environment. By applying minimal changes to the existing legacy environment, organizations can, to the satisfaction of the business owners, minimize the time and resources needed for BPM implementation while enjoying the benefits provided by BPM.

The rest of the paper is organized as follows. Section 2 is an overview of relevant research in the area concerning the subject of the article. The challenges, similarities, differences and co-dependence of SOA and BPM, along with the general idea on how to connect the two approaches are presented in Section 3. Additionally, Section 3 discusses the difficulties arising from a conventional approach to BPM-SOA integration, outlines the foundations for the new and improved integration model, and lists the needed characteristics of the new model. Section 4 defines the model in detail and explains its components and structure. A proof of concept implementation on a plausible business use case is presented in Section 5. The similarities and differences of the integration model with Enterprise Service Bus (ESB) architectures, Enterprise Application Integration (EAI) principles and the role of SOA are discussed in Section 6. Further on, Section 7 describes the evaluation of the model while Section 8 presents some conclusions.

2. Relevant Research on the Topic

SOA, BPM, optimizations of BPs and the construction of BPM solutions are often discussed topics. Relevant research ranges from emphasizing the process-oriented view [1], [6], [7], [8] to providing methods for analyzing and building business information systems [1], [8], [7]. The literature includes theoretical models for identifying the success factors of a BPM project [7], and analyses of the factors that complicate BPM implementation [3]. Articles [2] and [9] demonstrate a connection between business optimization and service-oriented architecture, [10] explores how the flexibility of SOA affects the core BPs of an organization, while [11] presents a detailed meta-model of the interactions between the activities of a BP and the functionality provided by software services. The benefits that BPM and SOA architecture bring to a business environment are further discussed in [5], [6], [12] [13], whereas in [14] one can find an interesting comparison of the structure of companies based solely on BPM or solely on SOA principles.

The process of building, optimizing and using a BPM solution, based on the specific requirements of a use case scenario, is explored in [4], [8], [15] and [16]. Their research aims range from emphasizing the importance of monitoring key process indicators (KPIs) [8], [15] and honoring service level agreements (SLA) [15], to applying the basic elements that support agility in managing BPs by utilizing Capability Maturity Model (CMM) [16].

Prototypes of BPM solutions are also described. Research in [4] presents a solution, specific for Third Party Logistics or 3PL companies, that integrates the processes within the organization with those of customers and external partners, while the idea of a business process model based on a set of services offered by distributed service providers (Software-as-a-Service or SaaS) is explored in [12]. BPM solutions are also used to detect misalignments between a BP and the supporting software systems and the effect of the misalignments on the performance of the BP [17].

Several attempts to integrate BPM in existing systems were made, but the articles describing them often do not provide a sufficiently detailed discussion of the conceptual model presented, the principles it is based on, the role of its components or the technologies, tools, frameworks and guidelines for its implementation. Implementation on a business use case, or a real-life problem that inspired the creation of the model, along with some kind of evaluation are also lacking. The existing articles, however, outline some interesting ideas, as will be seen in the following review.

An example of incorporating existing SOA services into BPM by using the modular principles of object-oriented approach is presented in [13], but the idea presented is to create a BPM solution in an SOA environment from scratch and without using a BPM tool. Unlike in [13], the authors in [5] discuss Business Process Management Systems (BPMS) and emphasize the need to apply a BPM tool while constructing BPM solutions. However, they define the roles for development teams, business users and project managers, which means that they mainly analyze the human aspect of the BPM integration effort.

Matei in [6] considers optimizing BPs by managing the existing services in an SOA architecture. He concludes that existing legacy applications should be re-engineered into a set of reusable modules or services. Still, the re-engineering of large and complex legacy applications can be difficult, if not even impossible.

The question of redesigning legacy systems is also discussed in [18] where the authors describe a three-layer SOA-BPM architecture which is based on a central service repository specially adapted for building an effective apparel quotation system. The authors also state the need for further research to determine the best way of integrating BPM into legacy systems. The dilemma is whether to build new services or to transform, adapt and reuse legacy systems.

The approach studied in [19] presents the integration of a BPM system in a healthcare environment. The focus lies on the dynamic allocation of long-lasting tasks to the currently available, or most suitable, practitioners. The authors define a middleware layer to connect the semantic layer (ontologies and rules used to define healthcare concepts and management principles of interdisciplinary healthcare teams or IHTs) and the execution layer (BPs defined in the BPM tool and existing legacy applications, or Hospital Information Systems, HIS). However, since healthcare is a branch where most work is done in practice, the role of the existing HISs is limited and not clearly defined, mostly used to start a BPM process on the arrival of the patient. With the help of the semantic layer, BPM is primarily used as a tool for assigning tasks and for further task and process management. The concept of reusing legacy application functionalities, or interfaces, is not studied as process actions are not executed in legacy applications and usually do not span through multiple HISs. The study presented in [20] addresses the need for business agility in the domain of insurance companies by creating a SOA architecture comprised of a Workflow Management System (WfMS), ESB, a Business Activity Monitoring (BAM) and a Business Rules Management (BRM) system. Although the architecture uses the ESB to invoke different services (via Remote Method Invocation or RMI), the option of replacing the exposed services with applications, interfaces and parts of applications' functionalities is not considered.

The usage of existing legacy applications as a part of the process monitoring system is presented in [14]. A service interlayer is constructed to connect systems across the organization and create an overview of the BPs. However, the additional service interlayer is built on top of legacy applications, and thereby uses the existing applications as a basic resource instead of incorporating the functionalities of these applications directly into the BPs.

A quite interesting idea is presented in [4] where the authors discuss the creation of software components that would interface legacy systems and use them as building blocks for the BPM solution. The reuse is limited to the fact that the existing legacy services based on Apache Tuscany implementation of a Service Component Architecture (SCA) are used to model a BP workflow in an OSWorkflow open source engine. Although the authors mention the reuse of legacy applications, the concept is not elaborated or studied further.

In [10] a complex and detailed view of the SOA-BPM integration is presented from the technical perspective. The model requires that parts of BPs (business rules and content specific routines) be extracted into separate components which would then be used to build a complex but effective BPM solution. The article does not clearly define the role of existing applications, and specific BPs contained within those applications, in a final BPM solution. The focus of the presented model lies on redesigning legacy application architecture, rather than building upon and using existing structures as its foundation. Also, the usage of a BPM suite was not considered.

The need for further research arises from the fact that detailed models describing the integration of BPM into a service-oriented legacy environment are still lacking. Also, only a small number of studies analyze tools and implementation methodologies which would make such integration possible.

3. Building BPM Solutions, Overview, Case Study and the Characteristics of a New Model

An overview of SOA and BPM is given in the following subsections, along with the explanation of difficulties that organizations face while building BPM solutions in existing legacy environments. A live real-world example of a legacy business application is studied in detail. Based on the conclusions, a set of characteristics needed in the new integration model for building a BPMA in an existing SOA business application architecture is outlined.

3.1. Differences, Similarities and Connections of BPM and SOA

A general discussion about the challenges of SOA and BPM is given in this section. Similarities, differences and the importance of a proper co-dependence of the two approaches are illustrated.

SOA and BPM both have a common goal: to achieve success in a business environment by accomplishing the desired business goals, a set of stable states that are valid after the BP is executed [17]. Generally, while BPM deals with the execution of well-defined processes in a specific order, SOA aims to expose the functionality of a system using well-defined services [5]. BPM and SOA concentrate on the management and coordination of processes, or services, and organize them in different repositories where they can be monitored and controlled [10]. By their nature, both SOA and BPM are iterative approaches where a set of predefined phases, or steps, is used to build, test and analyze the desired functionality, service or process [14]. Although SOA and BPM complement each other, they exist on different levels.

Operating on the lower level, SOA represents a set of technical principles and gives a dose of flexibility to the architecture of information systems [9] [20]. It has the task of combining simple services, self-contained units of functionalities [8], into complex ones while taking into consideration a set of technological limitations [10, 14]. As such, SOA serves as a network for filling the gaps between the infrastructure, data, and a BPM solution [4]. It ensures the technological means that provide resources for the implementation of goals defined by BPM.

As a set of management principles, BPM is on a higher level. It is more about the decomposition of processes into sub processes and activities and it ensures that the processes are integrated, automated, optimized, monitored and documented, so they can effectively support the decision-making process [7], [20]. BPM groups, sorts and presents the fetched data in a user friendly and logical way. The presented data is then subjected to different user input and user actions which, finally, create a quality service for the customer. Therefore, BPM refers to management of resources; it is a roadmap that integrates human-driven processes (processes in which human interaction takes place) with the use of technology to create the desired service or product.

Despite the availability of several BPM and SOA software solutions on the market, their full integration is still a tremendous practical challenge, mainly due to extensive organizational and IT changes that an organization needs to undertake [12]. However, there is evidence that a partnership between BPM and SOA produces an effective and indispensable solution that meets the challenges of the modern business environment [13].

3.2. Motivation, Difficulties and the Traditional Way of Building BPM Solutions

Let us explain the difficulties involved in the conventional approach to integrating BPM with an existing SOA-based legacy environment and analyze in detail the ensuing disadvantages using a real-world example of a legacy business application.

An inspection of the complex business environment in the banking sector reveals a multitude of different interconnected and intertwined business activities and processes. Changing a single process can affect multiple processes and cause unintended, negative consequences. The primary motivation for introducing BPM in that kind of environment is a systematic overview and control of this vast business architecture.

Satisfactory control can only be achieved if related business activities are first identified and grouped, as this step would lead to the use of individual applications for each set of related activities. That way, each separate business area, for example the entire business of corporate clients, would be controlled through only one complex module. A centralized management of a business area would allow a single point of entry, from where one could, using only electronic channels, create a business offer, arrange a meeting with a client or initiate processing and contracting of desired services. In addition, one would have total control over the situation and be able to see the status of issued requests at any point in time.

Most BPM solutions which are integrated into a SOA model are based on the fact that the presentation of business data and interaction with them are closely related to the BPM tool and, in most cases, presented by it [13]. That means that users can access, change or view the presented data using various predefined user interfaces and according to the given roles [10]. A business arranged in that manner is closely dependent on the BPM tool. Consequently, it is very difficult to sell a BPM-dependent software solution to a third party who do not use BPM-related software without extensive redesigns.

In addition, the implemented BPM tool must be able to handle large volumes of business data and massive objects, which complicates the implementation process. A comprehensive presentation of data hinges on the creation of complex forms with many data fields, menus, labels and presentation-related business logic, which is often not fully supported by the tool itself. Such a cumbersome architecture can produce quite a vulnerable application, which is then difficult to maintain or upgrade efficiently. The differences in data presentation between an architecture that contains legacy applications and an environment with a BPM solution are outlined in **Fig 1**.

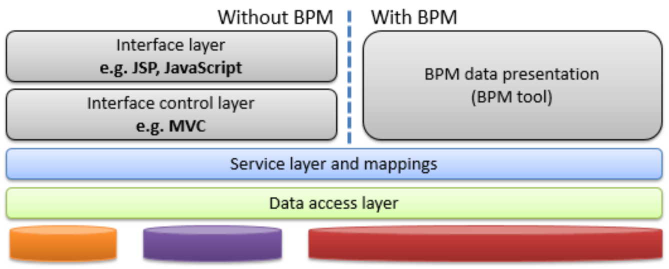

Fig 1. Architecture of a system with and without a BPM tool

Finally, a successful BP execution depends on the communication between different software applications supporting different parts of BPs. Much of the changes or reshaping of processes in the real world depends on how difficult it is to change the communication between the mentioned parties. If no standardized means of communication exist, communication is often ingrained in different, separate and monolithic software solutions. Processes built in that manner are strictly related to the technology used in the background and are, therefore, inert to changes which become very difficult and expensive to implement [18].

Legacy Business Application Case Study

To illustrate the problem accurately, let us look at a real-life candidate for a BPM implementation: a banking application managing the business of corporate clients. The application is based on the Java Spring framework and SOA architecture. It consists of four layers where each layer is implemented in a separate project. The data access layer is used for creating, updating, removing, and retrieving data from different databases. The service layer is like a wrapper around the data access layer that integrates the fetched data into reusable services using the basic business rules. The web presentation layer consists of interfaces and interface control components that manage the user's interaction with the data. Services from the service layer and additional business rules are also used to arrange, present and manipulate the acquired data. For example, in an application based on the Spring framework, the web presentation layer would consist of a Model–View–Controller or MVC pattern combined with JavaServer Pages (JSP) and JavaScript. The domain model, or layer, is common for object-oriented systems and describes the business area one is dealing with, for example banking, insurance, sales, etc. The domain model reflects the business model and consists of object representations of real-life artifacts specific to the domain in question. In the banking domain, one would have objects representing accounts, customers, orders and so on. Objects from the domain model are used in all other layers of the application.

Since its first operation 10 years ago, the application has constantly been developed and upgraded as the business environment grew and regulations changed. Over the 10 years, a team of 8 people on average continuously worked on the application. The application's core data is stored in two Oracle database instances, each containing a database data model comprised of more than 400 data tables. The application is connected with 18 different internal systems and applications which provide a variety of banking services, e.g. credit rating management along with on demand credit score calculation, service and product catalog functionalities, client exposure calculation, depositing management, access to a variety of central records, liquidity management, document digital signature support, extensive reporting in conjunction with a data warehouse system, document management features, etc. Additionally, external service providers are consulted regarding matters like the Foreign Account Tax Compliance (FATCA) or the verification of black-listed clients.

To clearly see the challenges of redesigning and implementing the described banking application in a BPM suite, one must grasp the size of the application and the resulting amount of work invested in its development. Even more importantly, one must compare the sizes of the application layers. To achieve the above goal, let us analyze the application in terms of the number of lines of code (LOC), not counting reports, JavaScript, CSS and JSP pages which are all part of the web layer. An analysis using SonarQube, an open source platform for analysis and inspection of code quality, revealed that there are 396.560 lines of Java code in the entire application. The size of the application layers compared to the total size of the application is shown in **Fig. 2**.

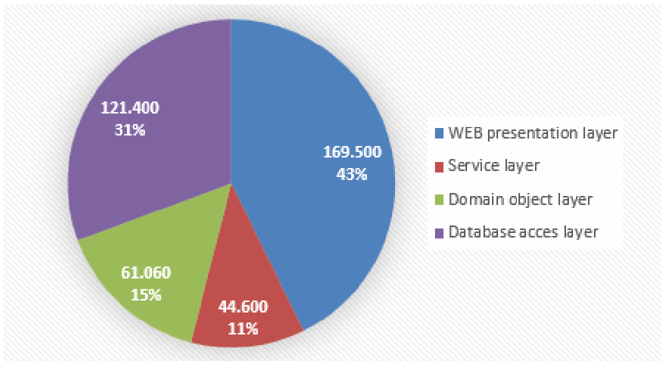

Fig. 2. Comparison of layer sizes in a banking business application based on the number of LOC

If we want to convert our banking application into a BPMA, a typical method would be to separate the service and database access layers and expose them as standalone services. Those services would then be used by the new BPMA. To create the BPMA, however, the entire Web presentation layer of the legacy application should be rewritten in a BPM tool. As shown in **Fig. 2**, the Web presentation layer makes almost a half of the original application (Java code + JSP + JavaScript + Jasper reports). Rewriting it from scratch would be a daunting and long-lasting task. To avoid that problem, an alternative BPM integration model is proposed in this article.

3.3. Characteristics Needed in a New BPMA Integration Model

To further clarify the contribution of the paper, let us look into the foundations of the new model for building a BPMA within an existing SOA business application architecture. Based on the conclusions made in the previous section, the following characteristics of the new model were identified.

Redesigning and totally rewriting existing processes in custom BPM tools, new frameworks or building new applications must be avoided. According to the case study presented in the previous subsection, rewriting everything from scratch would require a large amount of resources, time, and would also limit the organization's ability to deal with the ongoing market challenges during the redesign period. Therefore, a new model should, before all, maximize the utilization of existing functionalities, existing applications and architecture while requiring as little adaptation and change as possible. Maximum utilization of the existing architecture reduces the cost of BPM implementation as the use of resources is kept to the minimum. Also, since less work is needed, the implementation time is reduced.

In order to produce a loose coupling between a BPMA and the existing architecture, emphasis should be on adaptability. The existing legacy applications need to be operational without a BPMA as the system may become unavailable due to failures or upgrades of the BPM tool. The organization can decide to switch BPM tools and use a different, better, tool from a different manufacturer. In extreme cases, the organization can decide to stop using its BPM tool in order to cut costs or if it aims to develop a custom solution that better fits their needs.

A BPMA should manipulate with only a reduced set of business data required to achieve the desired functionality.

A major requirement is to allow changes in the business process flow by modifying the business logic contained in the BPMA. This needs to be done with minimal changes to the background applications and systems, or ideally, without modifications.

The ideal solution should be independent of different hardware, operating systems or technologies used in the organization. The integration is more successful when communication is independent of the back-end systems.

Every business process flow must reflect the real-time status of the underlying data in the background system. Changes in BPs must be visible in the underlying applications automatically, and the state of the data in those applications needs to be promptly displayed in the process application.

To assure flexibility, the integration model must be able to operate with different BPM tools simply by adapting the implementation of its Application Programming Interfaces or APIs for the interaction with the new tool.

Finally, it is important that the integration model is scalable. Scalability would facilitate the integration and use of different legacy applications, developed in different technologies and from different parts of the organization in a BPM solution.

4. The Proposed Integration Model

In order to achieve the characteristics described in the previous section, create a fluid data flow, clearly define the business environment and enable the flexibility of the system, the following architecture is proposed (**Fig. 3**).

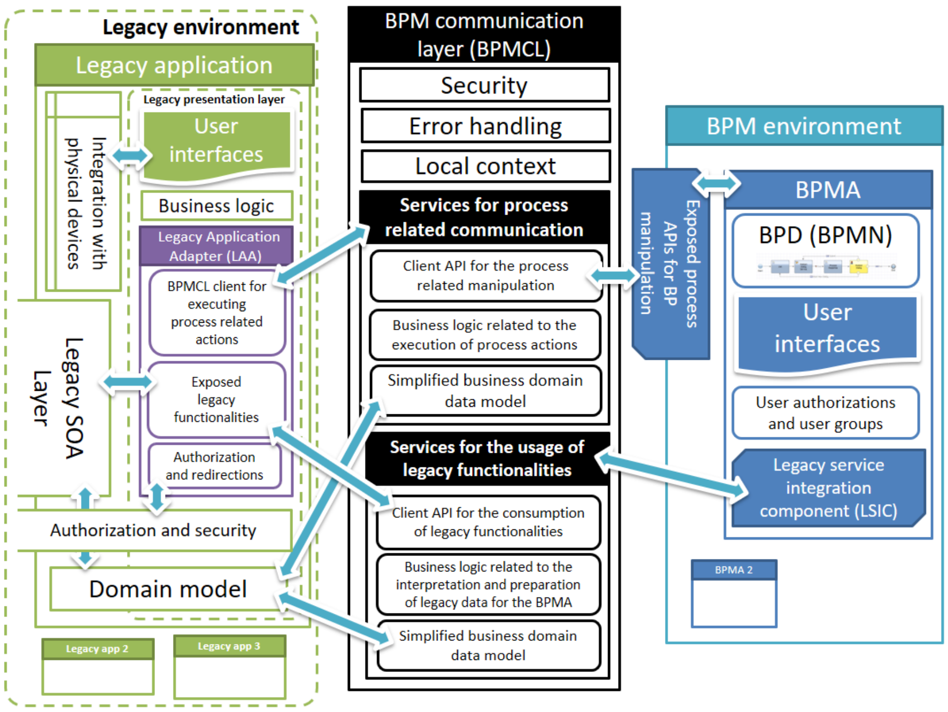
Legacy environment
Legacy application
Legacy presentation layer
User interfaces
Integration with physical devices
Business logic
Legacy Application Adapter (LAA)
BPMCL client for executing process related actions
Exposed legacy functionalities
Authorization and redirections
Legacy SOA Layer
Authorization and security
Domain model
Legacy app 2
Legacy app 3
BPM communication layer (BPMCL)
Security
Error handling
Local context
Services for process related communication
Client API for the process related manipulation
Business logic related to the execution of process actions
Simplified business domain data model
Services for the usage of legacy functionalities
Client API for the consumption of legacy functionalities
Business logic related to the interpretation and preparation of legacy data for the BPMA
Simplified business domain data model
BPM environment
Exposed process APIs for BP manipulation
BPMA
BPD (BPMN)
User interfaces
User authorizations and user groups
Legacy service integration component (LSIC)
BPMA 2

Fig. 3. Conceptual diagram of the proposed integration model

The integration model is based on three key layers or parts: the legacy application layer, BPM communication layer (BPMCL) and a BPMA built in the BPM tool. Legacy applications contain parts of the functionalities of existing business processes. BPMA describes the existing business process with a BPD modeled in congruence with the Process Model and Notation (BPMN) standard. BPMCL provides the methods for a universal way of communication between the legacy applications and the BPMA. The role, functionality and capabilities of each layer are shown and explained in detail in the following subsections.

The original contribution of this article resides in the definition of the BPMCL layer and the principles of integrating the BPMA with different legacy applications via the BPMCL in a way that exploits the existing functionalities and user interfaces while avoiding extensive redesigns of existing environments. Additional contributions lie in the conclusions of the presented case study, proof-of-concept implementation study and the extensive evaluation of the model by relevant experts from the BPM area.

4.1. Legacy SOA Layer

This layer represents the SOA implementation in the organization and enables the usage of existing functionalities. It contains the data access and service layer.

The data access layer consists of a variety of data servers, media for storage of all types of data, different databases on different machines and in different technologies.

All business data used in the system is initially retrieved through this layer. Some basic logic, required for different lookups, is also implemented in this layer.

The service layer consists of various services with specific purposes which can be divided into several groups or categories, based on the purpose. The division improves performance, enables better understanding and eases the maintenance of the overall system. Based on our analysis of the legacy environment (section 3.2) basic service categories were identified. If required, additional service groups can be added to the presented list. An overview of the proposed service categories (with examples) is shown in **Fig. 4**.

The first service group consists of services for a direct access to databases that manage the data required by the key functionalities of the applications. For example, a set of services that manages payment orders in an internet banking application. The basic characteristics of this category would be speed and high availability of the underlying databases.

The second group should consist of services accessing resources or databases that do not have a key role in the functioning of the system. A typical example is fetching data that are rarely used or accessing data archives that have no impact on the core functionalities. Another example of a non-essential resource would be a document management system (DMS). Such a system can be used to store a variety of contracts and confirmations that the clients use, but which are not crucial for performing core functions, for example execution of payment transactions.

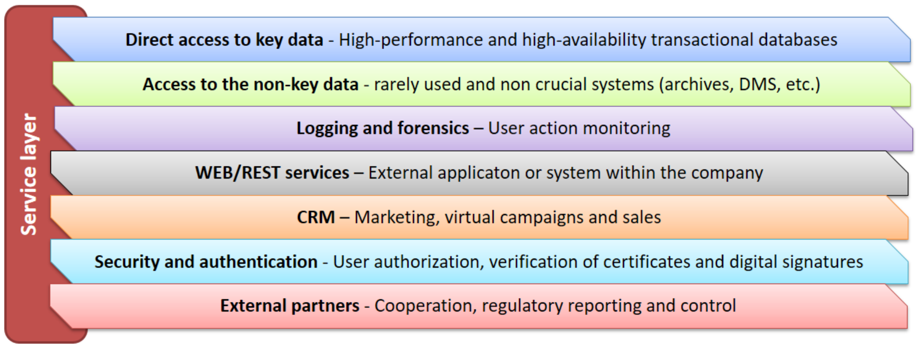
Service layer
Direct access to key data - High-performance and high-availability transactional databases
Access to the non-key data - rarely used and non crucial systems (archives, DMS, etc.)
Logging and forensics – User action monitoring
WEB/REST services – External applicaton or system within the company
CRM – Marketing, virtual campaigns and sales
Security and authentication - User authorization, verification of certificates and digital signatures
External partners - Cooperation, regulatory reporting and control

Fig. 4. Basic service layer groups

The next group encompasses monitoring and forensic services that enable tracking and logging of key actions that the user does while working with the application. A good practice here is to use NoSQL databases as they can store large quantities of data relatively fast.

Another part of the service layer would contain remote calls to the exposed web services of some external application, or system, to fetch additional data when needed. For example, in an online banking application, the data set to be retrieved from the external system can consist of detailed information about the clients from the central register, or verification of retail payment accounts. Services that would make such data exchange possible, broadly speaking, represent the communication between two applications over the Internet and can be implemented using Representational State Transfer (REST) or Simple Object Access Protocol (SOAP) based services.

An important part of the service layer is a connection with a specific Customer Relationship Management or CRM system which, as a marketing and sales tool, is used to inform the client about the current benefits, additional offers and new services.

The next addition to the service layer is the group of security and authorization verification services. A typical example is the use of a Validation Authority (VA) server to verify the certificates of users during the execution of key actions in the applications. Those actions include authorization during initial login, execution of orders, change of user authorizations, etc.

A final group of the service layer are services for communication with external companies, business partners or regulatory bodies. A classic example could be the delivery of mandatory reports to the tax administration offices or collaboration with a financial mediation organization.

4.2. Presentation Layer

The presentation layer integrates two types of user interfaces, parts of the presentation layers of legacy applications and simple BPMA user interfaces constructed in the BPM tool.

The presentation layer of the legacy application enables the interaction with data retrieved by the legacy SOA layer. It enables the usage of complete business functionalities and provides a comprehensive view of all business-relevant data. Interaction with the processes implemented in the legacy application is done entirely through this layer, along with the execution of all process-related actions.

The BPMA user interfaces deal with the initial user interaction as the user goes on to solve the tasks that were assigned to him/her. The primary role of the BPMA user interfaces is to show the crucial data regarding the process, and to redirect the user to the predefined forms in the presentation layer of the legacy application, where the rest of the work is done. To redirect the user, the BPMA utilizes the legacy service integration component (LSIC) which fetches the data needed for the redirection, and the authorization and redirection component of the legacy application adapter (LAA) which authorizes the user and executes the redirections trough the legacy application. The details of the redirection process are described in detail in the following subsection.

As an optional part of the presentation layer, one can show a BP diagram at each step of the process. This would allow users to clearly see the current and all remaining actions.

Legacy Application Adapter (LAA)

Legacy applications execute the actions of the existing business processes. To enable their communication with the BPMA, a Legacy Application Adapter (LAA) component is created. Integrated into the legacy application, the LAA has the task of sending instructions to the BPMA and responding to the requests of the BPMA with the help of the BPM communication layer, which will be described in the following section.

To help determine the intensity and the nature of the communication between the LAA and the BPMA, the term *process point* was defined. *Process points* mark locations where the LAA and the BPMA exchange information. Process points can be *standard* and *background*. *Standard process points* are events in the process flow where some process-related data in the legacy application changes. At that point, the legacy application communicates the change to the BPMA by updating the state of the BP instance. Updates can consist of minor changes of the BP instance-related data, or they can indicate a major change in the state of the process; e.g. mark a completion of a task or a point where the process flow moves from one legacy application to another. *Background process points* are important for the BPMA and indicate a call of the BPMA to the LAA using LSIC in order to fetch business data, get direct links to the tasks in the legacy applications, or to perform simple background checks. *Process points* have proven to be important for the discovery of the processes in legacy applications and their integration with the BPMA.

As the user is redirected from the BPMA to the legacy application to complete his/her task, the first point of interaction with the legacy application is the authorization and redirection component of the LAA. The authorization and redirection component completes the user login process with the help of the mechanism that already exists within the legacy application. Next, it prepares the necessary data regarding the user task and redirects the user to the screen where he/she can perform the assigned task. Upon the completion of the task, the LAA updates the status of the process in the BPMA and returns control over the process to the BPM environment.

To further elaborate the user interaction with the legacy application interfaces, *interaction points* were defined. *Interaction points* identify parts of the process flow where the user begins/ends the interaction with the process in the legacy application. Interaction points usually include *standard and background process points*. *Interaction points* have proven to be important concepts during the design of the authorization and redirection component of the LAA.

Each LAA consists of two parts. Application specific components contain the information concerning the legacy application so that the adapter can interface the BP in the legacy application. According to **Fig. 3**, specific components of the LAA include the authorization and redirection component and the component for exposing legacy functionalities. Existing authentication mechanisms, available in the legacy environment, are used by the components of the legacy application, while the authorization and redirection component of the LAA uses those legacy components to perform the authorization of the BPMA users (BPM users access the legacy application via specific links, which is further explained in sections 5.1 and 5.2). The component for exposing legacy functionalities can expose the functionalities of the legacy SOA layer in a needed form (e.g. via Representational State Transfer or REST endpoints). More importantly, however, it can expose parts of the functionalities that exist only in the legacy application and are needed for the BP execution. The specific components of the LAA are different for each of the legacy applications due to the differences in technologies, frameworks or methodologies used in the legacy applications.

The second part of the adapter is a general component that knows how to use the BPMCL to communicate with the BPMA. The general component, BPMCL client, utilizes the services for process-related manipulation provided by the BPMCL and is basically the same for each of the LAA in different legacy applications. It is possible to apply a method for automatic generation of the general LAA component in different programing languages, as described in section 5.1.

To illustrate the scalability of the model, let us examine the process of adding an existing BP, or a part of a BP implemented in a legacy application, under the control of the BPM solution. The described task would require the following actions:

- Creating the BPMA by building a BPMN model of the process in a BPM tool.
- Integrating and adjusting the adapter in the legacy application.
- Modifying the specific part of the adapter and connecting it with the targeted process, or parts of the process, by identifying *process* and *interaction points*.
- Integrating the communication with the BPMA based on the process workflow of the created BPMN process model.

An illustration of integrating the adapter in a legacy application, to support a part of the BP, is shown in **Fig. 5**. The existing legacy application functionalities are reused to execute the activities of the BP.

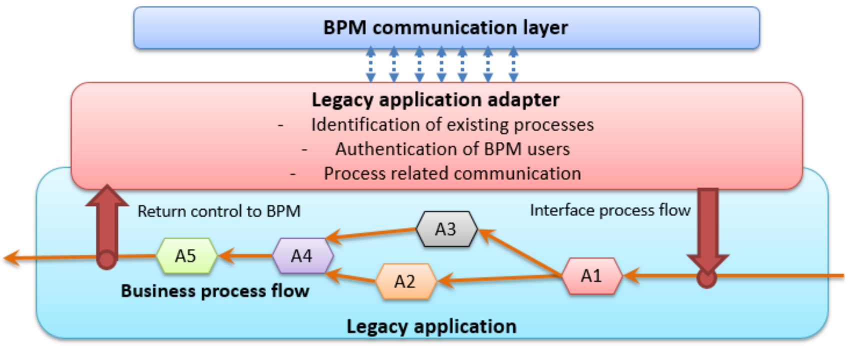
BPM communication layer
Legacy application adapter
- Identification of existing processes
- Authentication of BPM users
- Process related communication
Return control to BPM
Interface process flow
A1
A2
A3
A4
A5
Business process flow
Legacy application

Fig. 5. Adapter interfacing a process in a legacy application

The adapter also has the task of separating the legacy application from the BPMA. If the need arises, it can be used to completely exclude the BPMA from the legacy application. The adapter allows the interaction for only specific user roles and in line with their authorizations. Legacy application users, with roles that do not require BPM access, can continue to execute their daily tasks in legacy applications.

4.3. BPM Communication Layer (BPMCL)

BPMCL is the component, a middleware layer, which enables the communications between the legacy systems that perform specific business actions and the BPMA. It contains the methods for manipulating process data by calling process APIs exposed in the BPM environment. BPMCL gives flexibility to the integration model by providing means for the orchestration of BPs built in a legacy environment using a generic BPM suite.

Communication, which takes place in the key parts of the business process flow, consists of precisely defined services performing specific actions. Actions include the creation of processes, change of status, retrieval of the allowed user tasks, etc. Additionally, methods for the retrieval and manipulation of data from SOA-based legacy systems are also available on the BPMCL. The BPMCL can fetch legacy data from multiple legacy applications, perform the needed operations over the data, and present it to the BPMA in a desired way.

The BPMCL has its own domain data model that mirrors the BP data structures of the BPMA and is used in the communication with the BPMA and the legacy applications. This greatly simplified model is created by reducing the domain model of legacy applications so that it contains only a basic set of data needed by the BPMA.

Special components within the BPMCL are responsible for security and error handling.

A component for managing the local context of service calls is also defined. The local context is a devised principle where a specific set of data is transferred with a service call to identify the caller and define the transaction specific parameters of the service call. The local context is valid for the duration of the service call and is mandatory for each of the service calls. The following exemplify local context-related data: an identifier of the legacy application initiating the call, process instance id, or data related to the user who is working in the legacy application.

All presented BPMCL components are further explained in section 5.

4.4. BPMA with its User Interfaces

A BPMA is built in a BPM environment by using BPM tools. The process of BPMA creation requires that flow charts of business processes and sub-processes be defined. A BPMA created in this manner contains a process model of an existing BP that is executed in a legacy environment with the help of several existing legacy applications.

Based on user input, business process manipulation (starting a BP, changing BP status, assigning user tasks, etc.) is performed using process APIs (programmable interfaces related to process-bound communication) which are exposed in the BPM environment. The exposed process APIs enable access to BPM artifacts, business processes, tasks and process-related data. Process APIs, which are part of the BPM tool used by the organization, depend on the specific implementation of the tool's manufacturer.

In order to retrieve business-relevant data needed in certain steps of BPs, the BPMA contains a legacy service integration component (LSIC). The LSIC, with the help of the BPMCL and the business logic implemented in the BPMCL, calls the legacy SOA services exposed by the adapters in legacy applications. This solution enables direct communication with the organization's SOA implementation while the business logic implemented in the BPMCL allows one to interpret, manage or group the returned data as needed.

User authorizations and user groups defined within the BPM environment are used in BPMAs when building BPDs.

A set of BPMA user interfaces, input and presentation forms, together with the legacy presentation layer, are used to handle user interaction. User interfaces constructed in the BPMA provide the central points for user interactions. They receive and redirect users to a legacy application where the functionalities of BP actions are performed.

Different complex data types used by legacy applications and further simplified by the BPMCL are mapped to simple data types that a BPMA can handle.

5. Utilization of the Integration Model, Proof of Concept

In this section the proposed integration model is further explained from a business point of view. In addition, the section discusses the technical implementation of the model in a real-life legacy environment. A simplified version of a business process from the banking environment, the Loan Utilization Request Approval Process (LURAP), is used as an illustrative proof-of-concept (PoC) scenario to test the implementation of the integration model and confirm the feasibility of the overall approach.

5.1. Technical Concepts and Implementation Principles

This section describes the implementation principles of the proposed integration model and presents the selection of tools, frameworks and methods used in a PoC implementation effort in a real-life business use case scenario. Due to the limited scope of this work, the used software solutions, tools and technology standards are not explained in detail.

A standardized method of communication between the components of the model needs to be defined and the issue of security must be addressed. In the PoC implementation effort, communication is achieved via REST API due to its simplicity (JavaScript Object Notation or JSON data format), ease of implementation, testing (e.g. Postman REST client) and compatibility with legacy systems written in different technologies. HTTP Basic authentication method, in combination with SSL (Secure Sockets Layer) protocol and additional custom HTTP headers, was used to secure the communication.

As part of the architecture proposed in this article, a general BPM tool for building the BPM process model is needed. Process modeling, the creation of BPDs and their elements in the BPM tool, needs to be consistent with the BPMN standard as it enables the construction of a BPMA with all the elements described in section 4.4. Interaction with the BPMA must be enabled using predefined APIs exposed on the BPM tool. APIs must provide control over process instances, BPDs, services, activities, users, user groups and user tasks. The BPM tool needs to have the ability to call external services so that the LSIC can be created as shown in **Fig. 3**. Additionally, the selected BPM tool needs to support the construction of basic user interfaces.

To ensure the validity and flexibility of the integration model as shown in **Fig. 3**, and to validate the listed requirements of the BPM tool, a study of the APIs and features of four BPM suites was conducted (Red Hat JBoss BPM Suite, Bonitasoft BPM, Alfresco Activiti BPM and IBM BPM).

For the purpose of this PoC case study, a BPM tool from the IBM organization (IBM Business Process Manager or IBPM), a leading commercial suite in the BPM space [19], was selected. All details about the IBPM tool can be found in [21]. No in-depth analysis of the tool's structure, functions, or the philosophy of an iterative BPM implementation process will be undertaken in this article. IBPM provides control over all process-related artifacts through a set of predefined REST APIs, while a combination of dashboard and task completion coaches is used to enable human interaction with the BPMA [21]. IBPM Integration Services are used in the LSIC, while General System Services manage the data within the BPMA. Decision Services are used to apply business rules during the process execution with the help of BAL (Business Action Language) rules [21].

The role of BPMCL needs to be implemented in a standalone service application that is able to communicate with other legacy applications and the previously selected BPM tool. A variety of frameworks and tools are available for the task. For the PoC implementation Spring Boot framework along with Maven as a dependency management and build automation tool was used. Spring Boot is a favorable candidate for the construction of BPMCL as it features a minimal, non-XML based configuration, has an embedded servlet container, offers extensive support for nonfunctional requirements and enables a relatively simple installation in test and production environments. Detailed technical description of the basic BPMCL components used in PoC implementation is given in the following chapter along with a simplified Unified Modeling Language (UML) class diagram shown in the **Fig. 6**.

RestInterceptor implements the HTTP Basic authentication and has the task of processing *bpm-access-request-header* custom HTTP header. The data from *bpm-access-request-header*, along with *RequestLocalContext* and *RequestLocalContextHolder* classes, is used to implement the local context principle described in section 4.3. *CustomResponseEntityExceptionHandler* provides general exception handling while *ExceptionResponse* class represents a universal exception description. Services for process related communication are implemented via *LoanUtilizationBpmRestControler* REST endpoints, which contain the need business logic, use the simplified domain data model and execute the client APIs for the process related communication (*BpmRestManager*). One part of the *BpmRestManager* is a *BpmApiEndpoint* enum containing URI identifiers of a predefined format for the execution of the IBPM REST APIs. For example, to start a task `"/bpm/wle/v1/task/{taskId}?action=start&x-http-method-override=PUT"` URI is defined. Clearly, a verb tunneling principle [21] is used when the number of characters in a GET request exceeds the practical limitations of the HTTP protocol. Following the principles of services for the process-related communication, services for the usage of legacy functionalities are implemented via *LegacyRestController*, *LegacyManager* classes.

Springfox implementation of the Swagger 2 specification (*SwaggerConfig*) is used in BPMCL to document the functionalities of BPMCL based on an OpenAPI specification (OAS). The generated OAS document, in JSON or YAML (YAML Ain't Markup Language) format, enables computers to detect and people to understand the REST service endpoints without accessing the source code. This implementation exploits the primary advantage of the OAS: code generation tools can use it for the purpose of automatically creating REST clients, or REST endpoints, in different programming languages. To achieve the task, a Swagger Codegen tool is combined with AnthillPro continuous integration system to create BPMCL REST client artifacts and store the artifacts on a central repository of the organization (Sonatype Nexus).

The generated BPMCL REST client (general component of the LAA) needs to be adapted to the technologies and programming languages used in the legacy application. Therefore, to further ease the implementation, Swagger Codegen tool is used to generate REST clients in different programming languages.

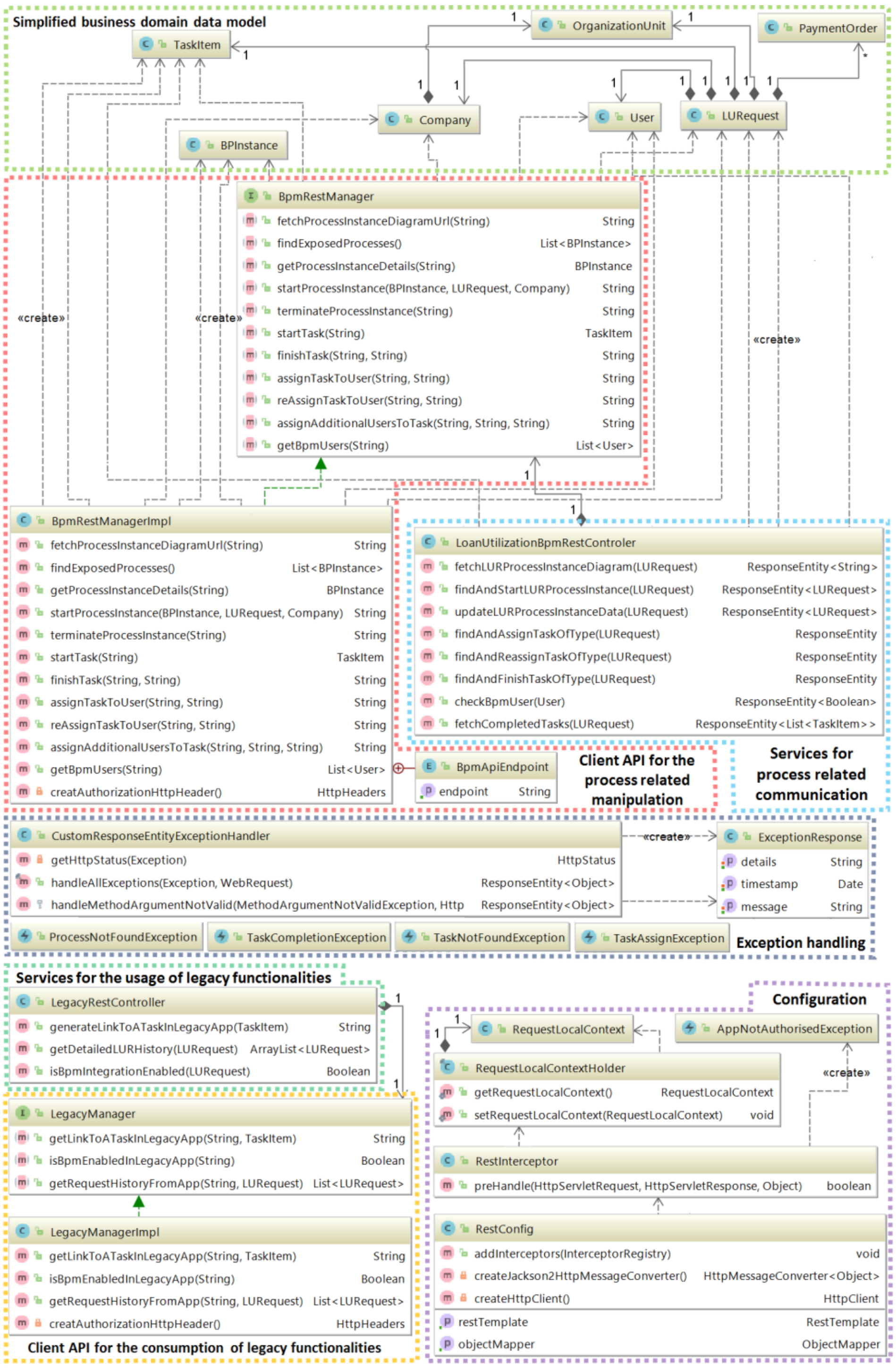
Simplified business domain data model
TaskItem
OrganizationUnit
PaymentOrder
Company
User
LURequest
BPInstance
«create»
BpmRestManager
fetchProcessInstanceDiagramUrl(String) String
findExposedProcesses() List<BPInstance>
getProcessInstanceDetails(String) BPInstance
startProcessInstance(BPInstance, LURequest, Company) String
terminateProcessInstance(String) String
startTask(String) TaskItem
finishTask(String, String) String
assignTaskToUser(String, String) String
reAssignTaskToUser(String, String) String
assignAdditionalUsersToTask(String, String, String) String
getBpmUsers(String) List<User>
BpmRestManagerImpl
createAuthorizationHttpHeader() HttpHeaders
LoanUtilizationBpmRestControler
fetchLURProcessInstanceDiagram(LURequest) ResponseEntity<String>
findAndStartLURProcessInstance(LURequest) ResponseEntity<LURequest>
updateLURProcessInstanceData(LURequest) ResponseEntity<LURequest>
findAndAssignTaskOfType(LURequest) ResponseEntity
findAndReassignTaskOfType(LURequest) ResponseEntity
findAndFinishTaskOfType(LURequest) ResponseEntity
checkBpmUser(User) ResponseEntity<Boolean>
fetchCompletedTasks(LURequest) ResponseEntity<List<TaskItem>>
BpmApiEndpoint
endpoint String
Client API for the process related manipulation
Services for process related communication
CustomResponseEntityExceptionHandler
getHttpStatus(Exception) HttpStatus
handleAllExceptions(Exception, WebRequest) ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException, Http ResponseEntity<Object>
ExceptionResponse
details String
timestamp Date
message String
ProcessNotFoundException
TaskCompletionException
TaskNotFoundException
TaskAssignException
Exception handling
Services for the usage of legacy functionalities
LegacyRestController
generateLinkToTaskInLegacyApp(TaskItem) String
getDetailedLURHistory(LURequest) ArrayList<LURequest>
isBpmIntegrationEnabled(LURequest) Boolean
LegacyManager
getLinkToTaskInLegacyApp(String, TaskItem) String
isBpmEnabledInLegacyApp(String) Boolean
getRequestHistoryFromApp(String, LURequest) List<LURequest>
LegacyManagerImpl
createAuthorizationHttpHeader() HttpHeaders
Client API for the consumption of legacy functionalities
Configuration
RequestLocalContext
AppNotAuthorisedException
RequestLocalContextHolder
getRequestLocalContext() RequestLocalContext
setRequestLocalContext(RequestLocalContext) void
RestInterceptor
preHandle(HttpServletRequest, HttpServletResponse, Object) boolean
RestConfig
addInterceptors(InterceptorRegistry) void
createJackson2HttpMessageConverter() HttpMessageConverter<Object>
createHttpClient() HttpClient
restTemplate RestTemplate
objectMapper ObjectMapper

Fig. 6. Basic class diagram of BPMCL components

Once the generated BPMCL REST client artifacts are uploaded to the central repository of the organization, they can be included in a legacy application (LAA component) with the help of a dependency management system.

Additionally, based on OAS, IBPM can automatically discover implemented REST services. Integration services of the BPMA LSIC are created by importing the OAS of the BPMCL's services for the usage of legacy functionalities (*LegacyRestControler*). Due to the limited space in **Fig. 6**. the interdependence between the services for the usage of legacy functionalities and the simplified business domain data model of the BPMCL is not shown.

LAA components largely depend on the structure and the technologies used in the legacy applications. However, some technical implementation guidelines can be applied in general. In terms of the Spring MVC model, the role of an adapter can be implemented into a special set of controller and helper classes that will work as a central entry point to the application. If we have a single page application, an LAA can be created as a separate set of components with a dedicated routing module.

Let us discuss a Spring MVC legacy application with an LAA. A BPMCL Java REST client artifact, auto-generated by Swagger Codegen, is used by the legacy application to execute a process-related action in *standard* process points. Simple REST controllers expose the existing SOA functionalities for the usage by the BPMA in *background* process points.

Authorization part of the redirection component is implemented in a central controller that uses the existing login methods of legacy applications: certificate-based authentication via Public Key Infrastructure (PKI) and a simple username and password authentication. Certificate-based authentication requires integration with physical devices (smart card readers or USB based tokens). The code for integration was reused from the legacy application. The controller for the redirection part of the authorization and redirection component, based on the process and type of action, redirects the user to the *interaction point* of the legacy application. After the user executes the action, the legacy application is closed.

5.2. Business Use Case Scenario

The LURAP (**Fig. 7**) is chosen due to its complexity. This business process both requires multiple user authorizations on different levels and involves multiple legacy applications in the process flow. It is, therefore, suitable for the demonstration of the integration model. When a loan is approved, the funds need to be transferred to the desired accounts. To achieve the transfer, the client needs to issue a Loan Utilization Request (LUR).

The client creates the LUR in the internet banking application, fills in the necessary data, issues payment instructions and attaches the documentation if needed. Further on, the LUR is validated, signed and sent to the bank for a series of approvals. First *standard process point* is located at this step. The adapter integrated in the internet banking application contacts the BPMA, creates the LURAP instance (*findAndStartLURProcessInstance*) and starts the first task of the process (*findAndAssignTaskOfType*).

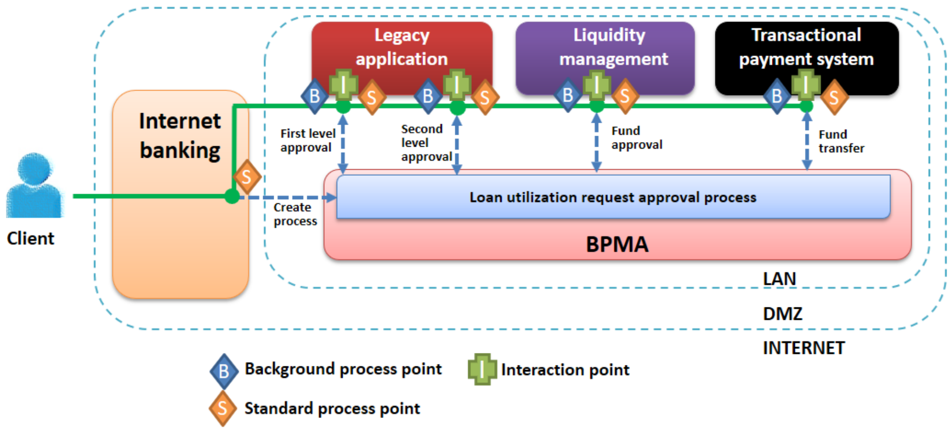

Fig. 7. LURAP BPM integration example

In the bank, working in a BPMA, a business information system user notices that a new first level approval task in the LURAP has just been assigned to him/her. Being a responsible user as s/he is, s/he immediately claims the task and goes on to solve it.

Based on the data available in the LURAP (status and type of the LUR), the BPMA asks the BPMCL to determine and create a link to the legacy application in the bank (to the LAA's redirection and authorization component) that is responsible for the necessary action (*generateLinkToATaskInLegacyApp*). The described action represents a *background process point*. Data contained within the generated link allows the redirection and authorization component to authorize the user (initial login data), prepare the data for the user task (LUR identifier, action type and business user information) and redirect the user to the exact form in the legacy application, where the required action can be executed (*interaction process point*).

Using this principle, the execution of actions directly from the BPMA is avoided. There is no need to implement complex forms and actions as a part of the BPMA because they already exist in the legacy applications. It is also not necessary to transfer and recreate the entire existing domain model (massive objects with many properties) in the BPMA, because the BPMA can function with a reduced set of data.

Let us look back to the business user and the approval process. Located on the approval form in the legacy application, the user selects the desired action from the list of the allowed actions. The application executes the selected action, verifies the LUR, and updates the status of the request in the back-end system or database. Here one can find the next *standard process point*, the legacy application notifies the BPMA about the changes (*findAndFinishTaskOfType* and *updateLURProcessInstanceData*) and the first level approval task is completed.

After the completion of the first level approval, the second level approval task is started and completed in the same manner, thereby fully approving the request. The following step in the process flow is to ensure the funds for the payments to the client. At this stage, the BPMA directs the user to the liquidity management application where s/he completes the task by reserving the necessary funds. Next, the BPMA initiates the task of creating payment orders with the help of the transactional payment system. Finally, after the payments are completed, the client is notified and the loan utilization request process ends successfully. Each of the described task executions include *background* and *standard process points* along with the *interaction points* as shown in **Fig. 7**.

5.3. Remarks, Conclusions and Additional Comments

Besides the *process and interaction points* that are required for the execution of BP, one can also define optional *process and interaction points*. Optional *background process points* can be added by allowing the BPMA user to, on demand, fetch data from the legacy applications. For example, the user executes the action which fetches the history of the request from the legacy applications (*getDetailedLURHistory()*). With the introduction of additional *standard process points* users can fetch a BPD diagram of the process instance (*fetchLURProcessInstanceDiagram()*) to view the current state of the process execution while performing tasks in legacy applications. Optional *interaction points* allow the user to access different legacy applications in case some additional verification of process-related data is needed during the BP execution.

While considering *interaction* and *process points* the question of access control mechanisms arises. In order to ensure that the tasks of a business process are executed by authorized users, proper authorization mechanisms must be applied [22]. In the proposed model, user access is largely defined by the existing access control mechanisms of the legacy applications as they execute the actions of the process. Studied legacy applications are based on the standard RBAC (Role-based Access Control) model (explained in [22] and [23]), with a dynamic SoD (Separation of Duty) concept (described in [22]). User roles defined in the legacy applications contain the authorizations to execute tasks and access resources. They are defined at the process level and do not depend on the current task or process instance being executed, involved resources (documents or some sensitive data) or the elements outside the standard access control entities.

Therefore, the proposed integration model features a double-layer RBAC model. The first layer is implemented in the BPMA, where, during the construction of BPD, user groups were defined and assigned to the lanes containing tasks which the group is allowed to execute. The second layer includes the existing RBAC mechanisms of legacy applications.

Further improvements of the access control mechanisms in the proposed model could include a more advanced access control model. In that case, extensions of the RBAC, such as the TRBAC (Task-Role-Based Access Control) model where permissions are assigned to tasks and tasks are assigned to roles, could be used.

In case of a BP where context aware access control privileges (privileges depending on current context conditions) are needed, the model would require a Context-sensitive access control model for business processes (COBAC) [22]. The advantage of COBAC is that it extends the RBAC model with business processes, activities, context and resource categories [22,23]. It enables the definition of access control policies on process instance and process context level, assignment of activities to roles and definition of resource access permissions for the activities [22,23].

However, for a successful integration of COBAC concepts in the proposed integration model, further research and adaptations of the proposed model are required.

PoC implementation effort has shown that it is not always necessary to implement all LAA components in the legacy applications. For example, if a certain legacy application has a single task of starting a process instance, only the auto-generated BPMCL client can be implemented. In a case where only certain legacy functionalities need to be used by the BPMA, only the component that exposes them can be created.

During the creation of a simplified BPMCL domain data model, several design options came into consideration. A BPMCL domain model can consist of parts of the domain models of legacy applications. In case of a shared code repository, during the AnthillPro automated BPMCL build process, classes of legacy applications are copied and packed with the BPMCL. The BPMCL can then use those data objects for the communication with the legacy applications and the BPMA.

If the domain classes of the legacy applications contain a large number of members not needed in the BPMA, one can use serialization to and deserialization from JSON to filter the not needed members. In the BPMCL, classes with a reduced number of identically named members needed for the BPMA are created. During REST calls to the legacy applications, JSON values of members that are not present in the BPMCL classes are automatically discarded in the deserialization process.

Another option is to create a separate domain data model in the BPMCL according to the needs of the BPMA. The created domain data model is included in the auto generated BPMCL client component of the LAA. During the execution of calls to the BPMCL, the data domain model from the legacy applications is remapped to the BPMCL domain data model.

The usage of non-SOA based legacy applications in the integration model, e.g. monolithic application architectures, was also considered. Due to many dependencies, intertwined functionalities and services, monolithic architectures are much harder to understand, which would cause problems in identifying and isolating parts of existing processes and difficulties in determining *process* and *interaction points*. Reusability is the main concept of the proposed integration model and successful reuse requires some sort of modularity in the existing systems. In a monolithic application architecture, the usage of separate, distinguishable components, along with parts of business functionalities is questionable. Services tend to get tightly coupled and entangled as the application evolves, which makes it difficult to isolate and use them for individual purposes. Additional modifications of the LAA component would be required, resulting in a tighter coupling with the monolithic application, which is the easiest way to complicate the code and increase interdependence.

Further detailed studies and adaptations of the integration model for use in non-SOA based environments are required, along with a proof of concept implementation effort.

6. EAI, ESB, SOA and the Proposed Integration Model

Let us discuss the similarities and differences of the proposed integration model with standard ESB architectures, EAI principles and the role of SOA.

ESB is a set of principles, an architecture, that allows multiple applications to communicate with one another via a bus-like infrastructure. Fundamentally, ESB architecture is a “communication agent”. It receives a message, translates it into the appropriate format and sends it to wherever the message is needed.

The proposed integration model serves to control the BPs and redistribute the work that certain applications must perform within a BP by utilizing existing functionalities.

Besides enabling the communication of different systems, the proposed model not only utilizes the existing user interfaces and the business logic but also coordinates human interaction in the execution of the BPs. ESB orchestrates services [20], while the proposed integration model orchestrates processes by identifying, extracting and integrating activities from different legacy applications with the goal of achieving a centralized point of interaction for the successful execution of BPs.

The integration of data across applications, with the goal of simplifying BPs extending through the connected applications, brings the proposed integration model closer to the EAI domain. To be precise, the undertaken presentation level EAI includes elements of the application level EAI and the user interface level EAI. The application level EAI is visible in the access to the BPs and to the data of legacy applications through legacy application interfaces. It allows the invocation of the existing business logic, is transparent to the integrated applications and also preserves the data integrity of the integrated applications. Elements of user interface EAI can be identified in the reuse of user interfaces of the legacy systems (along with any business logic embedded in those interfaces) which are combined with the user interfaces of the BPMA to create a central point for user interaction.

A BPMCL can be described as a custom extension to the ESB integration, specially adapted to the principles and needs of the proposed integration model. It is a middleware layer designed to provide the means for controlling BPs and legacy applications. While typical ESBs transport and translate data, a BPMCL also has the role of simplifying business data using its own data domain model (sections 4.3 and 5.3). Principles like the auto generation of BPMCL REST client artifacts (section 5.1) enable the compatibility of a BPMCL with legacy applications built in different technologies. The compatibility of the ESB implementation with the existing systems leads to a low-effort integration into the existing architecture [20]. A BPMCL not only enables the interaction between different systems but includes a business logic for isolating parts of existing applications as sets of reusable components, complete with functionalities and user interfaces, for the use by the BPMA. It also enables the orchestration of actions on BPs and helps in managing user interaction. Custom-defined exception handling, security and local context concepts further extend the ESB implementation.

In case that an existing ESB is successfully implemented in an organization, the principles of the BPMCL could be implemented with the ESB suite. Based on the possibilities and type of the implemented ESB suite, adaptations of the LAA, the BPMCL and the proposed principles of the integration could be made. However, further research into the topic, along with a concrete survey of the ESB suite used in the organization is required to provide an exact assessment the described scenario.

So, what is the role of SOA? ESB is one of the ways to implement SOA, a model of SOA implementation where one service is used for the management of communication between other services. Additionally, SOA can also provide the means for achieving EAI by facilitating the flow of information between applications. One can say that SOA makes the building blocks for ESB and EAI. It is the modularity of SOA, as discussed in the last two chapters of section 5.3, that enables the operation of the proposed integration model.

7. Evaluation of the Proposed Integration Model

The proposed integration model was presented to the group of 40 experts from five large commercial organizations. The group consisted of business information system users, project managers and business owners, and they were asked to give their opinions and identify possible use cases for the integration model. All interviewed experts were familiar with the BPM but differed in the amount of experience with different BPM initiatives.

The respondents with limited experience and limited practical knowledge in BPM were aware of the basic BPM concepts and principles. For the most part, they had studied examples of BPM initiatives, attended basic BPM software trainings or limited practical tests designed to demonstrate concepts and determine the abilities of BPM tools. The respondents with extensive practical experience had regularly designed and maintained business process management systems, were familiar with the role and operation of BPM tools and had insight into the practical advantages and disadvantages of BPM solutions.

The evaluation was conducted in two phases and was based on the principles of the Delphi method. The Delphi method is a structured communication technique used to achieve consensus forecasts from a group of experts in a structured and iterative manner. In the first phase, each of the experts individually attended a presentation on the integration model. They were then asked to state their opinion of the model and present a possible use case scenario, a conclusion about, or description of the situation where the proposed model would prove useful. Following the completion of the interviews, the obtained scenarios were systematized and summarized. In the second phase, the collected scenarios were verified. Each of the interviewed experts received a summary and was asked to rate each of the scenarios. In this way, based on the opinions of different experts, the most relevant use case scenarios related to the proposed integration model were obtained, which is the goal of the Delphi method. Detailed results of the evaluation are presented in the following subsections.

7.1. Research Results

Interviews with the experts produced the following use case scenarios in which the proposed integration model would be a valuable resource.

- ***Implementation of the BPM pilot project in the organization***
  People with less BPM experience viewed the proposed integration model as a possible way of doing a BPM pilot project. This was owing to the fact that only minimal adaptations of the existing systems are necessary, which makes it a faster and less costly implementation effort. Organizations could, utilizing the existing systems, build a BPMA to support one of the existing processes. In line with the needs of the organizations, the implementation effort would provide an insight into good and bad aspects of BPM and help them decide whether to proceed with a full BPM implementation. In case of a negative decision, BPM is simply decoupled from the existing systems.
- ***Practical assessment and comparison of different BPM tools***

The flexibility of the proposed integration model is suitable for examining the features and characteristics of different BPM tools. Since BPMCL separates legacy applications from the BPM tool and provides a universal way of communication with the BPM tool, the implementation of the same process can be relatively easy linked to BPMAs built into different BPM tools.

- ***Implementation in a stable, reliable, well-built legacy application environment***
  One of the experts with extensive BPM experience stated that an integration model which maximally utilizes the existing environment is great if the organization has a solid business system. A stable and solid business system implies well-built legacy applications which successfully support the business of the organization and are expected to meet the needs of the organization in the forthcoming future. Furthermore, the employees that already work on existing processes would not need an extensive BPM-related training as they would mostly use the existing applications.
- ***Further development and improvement of existing systems***
  The proposed integration model would enable further development of existing, or new, BPs in existing applications. If the organization had a complete picture of the process, i.e. if the performance data were gathered by a BPM tool, any problematic activities could be detected and the corresponding background application responsible for the activity could be modified accordingly. The experience of developers with existing technologies and their knowledge about existing applications would enable a faster implementation. Once a new BP is implemented in a legacy application and a BPMA is created, the two would be integrated.
- ***Potential way of building an iBPM system***
  Defined by Gartner, Inc. as an extension of a conventional BPM tool, iBPM can glue together the capabilities of pre-built BPs with business planning approaches by means of capabilities like systems cross linking, cloud computing, real-time decision-making, etc. iBPM sounds like a new concept, but it was already happening when enterprises were using BPMS systems with other tools to optimize their processes. The presented integration model could be used to achieve the described connection.
- ***Environments with limited resources***
  Organizations where limited resources prevent the re-engineering of existing applications, but there are initiatives for analysis and improvement of BPs.

In the second phase, the collected scenarios were verified. Experts needed to evaluate the quality of use case scenarios for the integration model by rating it on a scale from 1 to 5 (1 being *Very poor*, 5 being *Excellent*). The results of the evaluation are shown in **Fig. 8**.

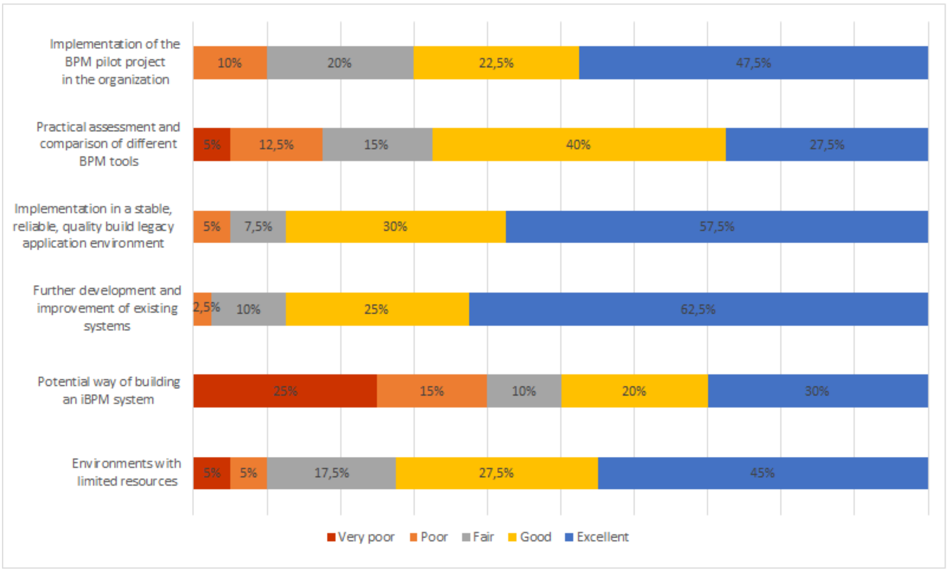

Fig. 8. Evaluation of the use case scenarios for the proposed integration model

To conclude the evaluation of the use case scenarios let us consider what one of the interviewed project managers said: ”Often, the best way to use a tool is to use only the best of its features.” A key part of any BPM tool is a feature for the modeling and managing of BPs. Moreover, a BPM tool incorporates predefined components that enable interaction with data flows in the modeled processes. These often include standard interfaces, predefined actions, or resources for solving some general problems. While BPM tools involve components for solving general problems, specific organizations that implement BPM have specific problems and specific needs that are already met in the existing systems and the information structure of the organizations. Consequently, one could get optimal results by combining the core management of business process flow from the BPM tool with the interfaces and specific solutions in the existing applications.

7.2. Additional Comments, Conclusions and Concerns

An interesting conclusion was made by one of the experts. He observed that the proposed integration model would not be the best choice for an environment with a lot of small, independent, legacy applications, each managing a limited number of processes. In such an environment one would need to create many separate adapters (one per application) and connect all the small legacy applications with the BPM solution. Since the process would increase the cost and the complexity of the BPM implementation, a better course of action would be to redesign the smaller applications into a set of services. Those services would then be used by the created BPMAs to support the required processes. Therefore, the proposed integration model is better suited for a business environment with a few large applications which control most of the BPs in the organization.

Next, businesses occasionally implement new processes that are not strictly related to the existing functionalities and existing applications. What should be done in such situations? The implementation of those processes into existing applications is not a good choice due to the disparity between the new processes and legacy applications. Also, it is not profitable to build separate applications to support the new processes, and then interface them with the BPM solution. The reasonable course of action in the described situation would be to develop a BPMA and the underlying services to support the created process.

Using existing applications as a part of the BPM solution is a bad idea if the implemented processes require redesigns. A better course of action would be to redesign and re-implement the processes using a BPM tool.

By processing the generated links which enable the users to access legacy applications, the authorization and redirection component of the LAA utilizes the existing authentication techniques. An organization’s existing information security policies can thus be reused [16]. In case no valid security policies exist, a simple Single Sign On method (SSO) with the help of Lightweight Directory Access Protocol (LDAP) service, a central place to store authentication data, can be implemented. Different applications and services could connect to the LDAP server and validate users.

8. Conclusion

This research explores how important it is for organizations to control their BPs. Furthermore, it shows how control of BPs can be achieved by integrating BPM into existing SOA based legacy environments.

There are several examples of BPM integration in the literature. They, however, do not build upon the existing structures or utilize existing structures. Instead, they focus mostly on redesigning the legacy application architecture. As the process of redesigning legacy environments requires large amounts of time and resources, it is a luxury that many organizations cannot afford. Furthermore, a review of the literature showed that the usage of modern BPM tools which simplify the creation and management of complex processes is rarely considered when creating BPM solutions.

Both the case study presented in this article and the existing literature support the premise that a new integration model is needed to ease BPM implementation in the existing environments. This paper presents new considerations regarding the question on how to adapt the existing systems for BPM implementation. The undertaken research identifies the difficulties and problems relating to the integration of BPM in SOA legacy environments. Based on the results of the study, the requirements for a new integration model are defined.

A generic, scalable multi-layered integration model is proposed, the role of each layer in the overall solution is closely examined and the provided benefits are discussed. In order to save resources and time, which are usually spent on redesigns and refactoring during BPM implementation, modifications of existing legacy applications are kept to the minimum. The model enables companies to utilize business actions that already exist in legacy applications and connect them to form a BP. A loose interdependence between the BPM solution and the legacy environment is achieved. Consequently, legacy applications are operational without a BPMA. The model provides a central place where business users can interact with a business environment instead of coping with multiple applications and different systems to perform daily tasks. A better understanding of business reality is provided and the model can be helpful when detecting flaws and suggesting improvements in an organization's business activities. The paper provides concrete illustrations of ideas and steps, and it also presents a specific business case of integration.

The evaluation section presents several possible applications of the model in various organizations and in different situations. The most suitable environments for the model's application are discussed in details. Technical aspects of the integration model are reviewed in a proof of concept implementation effort. Additional remarks and conclusions of the implementation effort are outlined. The paper also presents a partial answer to the eternal question whether and when one needs to use existing systems, and when some redesigns are necessary.

Several limitations of the proposed model are also outlined. The integration is reviewed in a single case study and evaluated by experts based on the principles of the Delphi method. Full scale application of the model in different organizational contexts would contribute to its consolidation, offer proof of its benefits, and suggest possible modifications or improvements. Additionally, the question of versioning and a more advanced exception handling will be covered in detail in future research. Next, the success of BPM implementation was discussed rather broadly and additional research into the criteria for measuring success is needed. This is particularly challenging since success is not something one can measure easily and since it can vary both in magnitude and over time.

The research presented in this article may provide organizations with the opportunity to use the proposed integration model as a platform for achieving greater business value and thereby help them to become more flexible and more focused on their core business objectives.

References

1. Elaine A. Carvalho, Tatiana Escovedo, Rubens N. Melo: Using Business Processes in System Requirements Definition, 33rd Annual IEEE Software Engineering Workshop, 125-130, 2009.
2. Matej Hertiš, Matjaž B. Jurič: Ideas on Improving the Business-IT Alignment in BPM Enabled by SOA, 2013 International Conference on Information and Communication Technology (ICoICT), 55-60, 20-22 March 2013.
3. Imanipour Narges, Talebi Kambiz, Rezazadeh Siavash: Obstacles in BPM implementation and adoption in SMEs; University of Tehran, 2012. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1990609 [17 February 2016]
4. Wen ZhenHua, Huang Yousen, Deng ZiYun, Zhang Wei: SOA – BPM Based Information System for Promoting Agility of Third Party Logistic, International Conference on Automation and Logistics, 248 – 252, Shenyang, China 5-7 Aug. 2009.
5. Fuhua Ge, Shaowen Yao, Architecture combining SOA and BPM, Institute of Electrical and Electronics Engineers, Jan 27, 2011.
6. Gheorghe Matei: SOA and BPM, a Partnership for Successful Organizations, Informatica Economică vol. 15, April 2011.
7. P. Trkman: The Critical Success Factors of Business Process Management, International Journal of Information Management, No. 30(2):125-134, April 2010;

8. Razvan Daniel Zota, Liviu Ciovica: Designing Software Solutions Using Business Processes, Procedia Economics and Finance Volume 20, 2015, Pages 695-699
9. Marinela Mircea: "Adapt Business Processes to Service Oriented Environment to Achieve Business Agility", Journal of Applied Quantitative Methods; Vol. 5 Issue 4, p679, 2010.
10. Nan Wang, Vincent Lee: An Integrated BPM-SOA Framework for Agile Enterprise, ACIIDS'11 Proceedings of the Third international conference on Intelligent information and database systems - Volume Part I, Pages 557-566, Springer-Verlag Berlin, Heidelberg 2011.
11. Bazán Patricia, Roxana Giandini, Gabriela Perez, Javier Diaz; Process-Service Interactions using a SOA-BPM-based Methodology, Chilean Computer Science Society (SCCC), 2011 30th International Conference, 9-11 Nov. 2011, Page(s): 100 – 107;
12. Alexandre Perin de Souza, Ricardo J. Rabelo: An Approach for a more Agile BPM-SOA Integration supported by Dynamic Services Discovery, 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, page 186-195,
13. Imran Sarwar Bajwa: SOA Embedded in BPM: A High Level View of Object Oriented Paradigm, WASET 2011 Spring International Conference, 304-308. Tokyo, Japan, (2011)
14. 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, World Academy of Science, Engineering and Technology 45, 2008.
15. S. Al Aloussi: SLA Business Management Based on Key Performance Indicators, Proceedings of the World Congress on Engineering 2012 Vol III July 4-6, London, U.K.
16. Nivedita P Deshmukh: Leveraging BPM Discipline To Deliver Agile Business Processes In Emerging Markets, 2013 IEEE International Conference on Business Informatics, 338-345;
17. Lerina Aversano, Carmine Grasso, Maria Tortorella: Managing the alignment between business processes and software systems, Information and Software Technology, Volume 72, Pages 171–188, April 2016.
18. Li, Qian; Chen, Yu; and Zhang, Lanfang: An Apparel Trade Quotation Architecture Based on BPM and SOA; PACIS 2009 Proceedings. Paper 115.
19. Nihan Çatal, Daniel Amyot, Wojtek Michalowski, Mounira Kezadri-Hamiaz, Malak Baslyman, Szymon Wilk, Randy Giffen: Supporting process execution by interdisciplinary healthcare teams: Middleware design for IBM BPM; Procedia Computer Science, Volume 113, 2017, Pages 376-383, https://doi.org/10.1016/j.procs.2017.08.350, September 2017.
20. Hausotter, A., Koschel, A., Zuch, M., Busch, J., Hödicke, A., Pump, R., Seewald, J., Varonina, L.: Applied SOA with ESB, BPM, and BRM - Architecture Enhancement by Using a Decision Framework, in: Westphall, C.M., de Barros, M. (Hrsg.), Proc. SERVICE COMPUTATION 2016: The Eighth International Conferences on Advanced Service Computing, IARIA: Rome, Italy, S. 20-27. March 2016.
21. IBM Company: Overview of IBM Business Process Manager, ftp://ftp.software.ibm.com/software/integration/business-process-manager/library/ibpm_overview_pdf.pdf, April 2017.
22. Milosavljević, G., Sladić, G., Milosavljević, B., Zarić, M., Gostojić, S., Slivka, J.: Context-sensitive Constraints for Access Control of Business Processes. Computer Science and Information Systems, Vol. 15, No. 1., DOI: 10.2298/CSIS160628037M, 2018.
23. Stevan Gostojić, Goran Sladić, Branko Milosavljević & Zora Konjović: Context-Sensitive Access Control Model for Government Services, Journal of Organizational Computing and Electronic Commerce, 22:2, 184-213, DOI: 10.1080/10919392.2012.667717, 2012.

Mladen Matejaš was born on May 26, 1985 in Zabok, Croatia. After completing the mathematical high school in Krapina, in 2003 he enrolled at the Faculty of Electrical Engineering and Computing at the University of Zagreb. He received his B.Sc. degree (Dipl. Ing.) in computing in October 2008, by defending his graduate thesis entitled: "*Component Approach to Application Development in Heterogeneous Systems*". Since February 2009. he has been employed at Privredna Banka Zagreb in the Application Development Department, where he participated on different software development and system integration projects. With an interest in the Business Process Management research field he enrolled in doctoral studies of Computer Science at the Faculty of Electrical Engineering and Computing at University of Zagreb, Croatia. Since then, he participated in several domestic and international conferences, published an international reviewed article, participated in the EU funded project "HEUREKA-knowledge to success" and successfully completed several education courses and workshops in various fields. He uses English very well in language and written communication and possesses basic communication skills in German language.

Krešimir Fertalj is professor of computer science at the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia, currently serving as Head of the Department of Applied Computing and Head of the Laboratory for Special Purpose Information Systems. He achieved his B.Sc. (Dipl. Ing.), M.Sc. and Ph.D. degrees in computing at the same university. Since graduation in 1988, he has been working at the Department of Applied Computing, where he currently lectures a couple of computing courses on undergraduate, graduate and doctoral studies. His professional and scientific interest is in automated software engineering, complex information systems, project management and in software security. He was the leader of several scientific, and research and development projects. He was the mentor of more than 200 graduate, 9 MS and 8 PhD theses. He is a senior member of IEEE and member of Croatian Academy of Engineering. He was one of the founders and member of management board of PMI chapter in Croatia.

Received: October 5, 2017; Accepted: August 10, 2018