24th International Scientific Symposium
Strategic Management and Decision Support Systems in Strategic Management

17th May, 2019, Subotica, Republic of Serbia

Zoran Dragičević

PhD student
Faculty of Economics in Subotica
Subotica, Republic of Serbia

Saša Bošnjak

Faculty of Economics in Subotica
Subotica, Republic of Serbia

HARMONIZING BUSINESS AND DIGITAL ENTERPRISE STRATEGIES USING SOA MIDDLE-OUT AND SERVICE-BASED APPROACH

Abstract: The organization's agility represents its ability to respond fast to changes by the transformation and/or optimization of business processes, therefore it is a key factor in the competitiveness and growth of the organization. Bearing in mind the importance and expansion of the use of digital technologies, the organization's agility can be compromised if the digital strategy and IT resources are not flexible enough to adequately respond to changed business conditions. Service-Oriented Architecture (SOA) is an approach to the development of business software systems that promote better alignment of business and digital strategy, i.e. business goals and IT resources, enabling the organization to faster adapt and respond to changes in the business environment. However, the success of the SOA initiative depends largely on the choice of delivery strategy and the way services are identified, as well as the choice of an approach in the implementation of SOA, given the way of integration and communication between services. The middle-out delivery strategy is a compromise between top-down and bottom-up approach. This approach, at the same time, supports SOA implementation guided by a strategic vision, business strategy and strategical goals, and use of existing IT resources with focus on urgent requirements and tactical goals. Realization of middle-out delivery strategy is based on several small, iterative SOA projects, where each individual SOA project was implemented to meet specific business goals and requirements. On the other hand, in order to respond to new challenges and requirements of the digital era regarding the distribution, scaling and increased complexity, SOA evolves towards increasing agility and a simpler, service-based approach at business and technical level. This avoids the complexity of the Enterprise Service Bus (ESB) as an integrator in communication between the services. Avoiding the ESB as a universal mediator and orchestrator in communication between the services enables a serious shift in the delivery speed and scalability of software solutions. This paper, in general, discusses the ways of more efficient alignment of business and digital enterprise strategies based on the implementation of SOA initiatives for the development of business software systems. Within the framework of the research of this paper, the very significant challenges and best practices are identified and clearly distinguished, as well as the advantages and disadvantages, related to the architectural and methodological aspects of the implementation of the SOA middle-out delivery strategy and service-based approach. The complete research effort, as well as the results obtained, is dedicated to the effective implementation of SOA in the context of the development of business software systems. The results of the work include consideration of the evolution of SOA-based approaches in the digital era, leading to increased agility and a reduction of complexity.
Keywords: SOA, business and digital strategy, middle-out, service-based approach, software development

INTRODUCTION

The ability to respond quickly to changes by business process transformation and/or optimization is a key factor in the competitiveness and growth of organizations in an increasingly competitive environment and market conditions dictated by globalization and the expansion of the use of digital technologies. However, this ability can be compromised if the digital strategy and IT resources are not flexible enough to adequately respond to changed business conditions. Service-Oriented Architecture (SOA) is an approach to the design of corporate software solutions that affirm better compliance of business and digital strategy, i.e. business goals and IT resources, enabling the organization and its business partners to adapt more quickly and respond to changes in the business environment. On the other hand, SOA can be viewed as a result of the application of service orientation, where service orientation is a paradigm that unambiguously establishes a framework composed of a specific set of software development design principles (Erl, 2005; SOA Manifesto, 2013).

SOA aims to improve the efficiency and productivity of an organization using services as a basic tool for implementing enterprise business logic in software solutions and supporting the implementation of strategic goals related to service-oriented computing, where three strategic goals are the most important: increasing organizational agility, increasing ROI and reducing IT costs. Well-implemented SOA projects directly link IT resources to the business goals of the organization. This directly improves key aspects of the organization, such as, among others, 1) to build stronger relationships with customers and suppliers, 2) facilitate obtaining more accurate, more complete and more updated information of business intelligence that are critical to making better decisions. The direct implication of these two improvements reflects better support of key business processes that are achieved through increased availability of information with a significant impact on increasing employees' productivity. On the other hand, SOA as an advanced integration technology (Šereš & Tumbas, 2014) provides a platform for easy development and maintenance of integrated systems and applications, and easier harmonization of IT resources with the business model and changing business requirements. This approach enables faster and cheaper application development, with a clear focus on increased productivity, flexibility and ease of maintenance (Erl, 2007).

However, although well-planned and implemented SOA projects can help the organization to improve competitiveness, they do not provide a guarantee of a success. The success of SOA in increasing of competitiveness, as a very complex process, largely depends on the choice of delivery strategy and the way services are identified, on the one hand, as well as the choice of approach in the implementation of SOA, especially from the aspect of choosing the mode of integration and communication between services, on the other.

In the relevant literature, the following SOA delivery strategies have been identified: top-down, bottom-up, meet-in-the-middle (also known as outside-in) and middle-out (Terlouw et al., 2009; Slimani et al., 2013). There is a broad consensus among researchers on the necessity of reaching a compromise between top-down and bottom-up approach. The first approach starts from a wide perspective of the enterprise and its strategic goals, trying to define specific solutions that fit into the strategic framework and tactical requirements of individual projects. The second approach starts from existing systems, technologies or common services, giving preference to urgent requirements and tactical goals.

Middle-out is a balanced hybrid approach, i.e. compromise between top-down and bottom-up approach, which at the same time takes the best of both. The application of such a delivery strategy produces both, the compliance of business and information infrastructure with strategic goals, and services suitable for reuse. In the context of such hybrid approach, the effective implementation of SOA can be viewed as a careful balancing and alignment of strategic objectives on the highest level, and immediate, urgent requirements on the tactical and operational level of business, in a way that supports the reuse of services. Therefore, one of the clearly visible direct advantages of such software solutions is the possibility of reuse of services. In addition, the positive aspects of this hybrid approach include achieving relatively fast results, primarily based on reducing potential conflicts between business analysts, software architects, and developers. Avoiding and overcoming conflicts between people involved in the process of business software systems development contributes, above all, to a better understanding of business requirements by software architects and developers, and thus avoiding the problems in translating key strategic goals into concrete software solutions that need to be delivered fast. It also prevents the occurrence of problems that can lead work in the bounded context, with a focus on the implementation of specific software solutions and requirements for this software solution. Due to the lack of consideration of the wider context and the lack of a clear link with the strategic vision and goals, work in the bounded context for a long-term result has mainly higher costs of business software solutions (Arsanjani, 2004; Marks & Michael, 2006; Microsoft, 2006; Erl, 2007; Rosen et al., 2008; Kohlborn et al., 2009; Valipour et al., 2009; Mirarab et al., 2014). However, this approach is difficult for the implementation of SOA (Slimani et al., 2013).

On the other hand, the digital era has brought new challenges and demands when it comes to distribution, scaling and increased the complexity of software solutions. This has led to the evolution of the SOA and the emergence of several approaches in the implementation of SOA, which, among other things, differ in terms of service granularity, resource sharing, integration and service communication, such as a traditional ESB approach, service-based approach, microservices, and serverless approach (Ford et al., 2017). Traditional ESB, as a complex integrator and a service orchestrator, usually implemented in the form of a monolithic application, was not designed for cloud (Villamizar et al., 2015; Taibi et al., 2017). For this reason, in the digital era the focus has shifted towards increasing agility and simpler service-based approach at business and technical level, avoiding the complexity of the ESB to allow faster delivery and

scaling. In this context, the particular challenge is to achieve a balance between the development of business functionality (business services) and integration with existing systems (integration services), which will deliver an agile, decentralized, and at the same time resistant architecture that supports the reuse of the service in each iteration.

Bearing in mind the new challenges and demands of the digital era, like fast delivery, scaling, increased complexity and agility, the question arises as of how to align the business and digital strategy in the context of development of business software systems using the SOA middle-out delivery strategy and service-based approach in the implementation of SOA. In this regard, this paper identifies and addresses the challenges and best practices, as well as significant advantages and disadvantages, related to architectural and methodological aspects in the application of the SOA middle-out and service-based approach in the context of the development of the business software systems.

The paper is organized as follows: Section 2 contains a critical overview of existing knowledge related to service identification and delivery strategies, with a special focus on the middle-out approach, as well as different approaches to SOA implementation, with a special emphasis on service-based approach. Section 3 defines the research questions and describes the applied research method. Section 4 presents the results of a descriptive case study and an exploratory case study of the development of a business software system using the SOA middle-out and service-based approach. Section 5 gives answers to research questions, with a reference to the transformation of the applied traditional SOA methodology, identifies potential follow-up research and highlights research limitations. Finally, conclusions are drawn and future works delineated.

2. BACKGROUND AND RELATED WORK

The digital strategy connects people, processes and technology (Nahrkhalaji et al., 2018), with one key difference between the digital strategy and the traditional IT strategy. The traditional IT strategy aims to support the implementation of a business strategy while the digital strategy can be seen as an IT strategy that tends to become the highest-level business strategy (Sebastian et al., 2017). Digital strategy radically re-examining the use of digital technologies to enhance the user experience, operational processes, business models and business strategies, focusing on one of the above areas and applying specific initiatives to successfully respond to market changes (Westerman et al., 2011, 2014; Fitzgerald & Kruschwitz, 2013). In this regard, the hierarchy is gradually eliminated and the boundary is erased between business and IT strategy, leading to their gradual fusion into a digital strategy (Bharadwaj et al., 2013). Therefore, the need to align business and IT strategy is increasingly becoming a need to align business and digital strategy. However, aligning business and digital strategy is a bigger challenge because enterprises have a problem when they try to up-front define a digital strategy, due to a dynamic environment that requires simultaneous changes and reconfiguration of the various components, not only at the strategic level but also at the level of business and IT resources (Yeow et al., 2018). Therefore, the importance of IT resources in increasing the ability of enterprises to faster deliver software systems to the production environment is particularly emphasized (Grover, 2013; Mithas et al., 2013).

In such a context, to understand better importance and potential of SOA to align business and digital strategy, it is necessary to look at SOA origins, motivational factors, and SOA implementation challenges. Although the term SOA skewed in 1996., according to the largest number of sources, forward-looking companies, such as banking, telecommunications and finance companies, were previously able to implement service layers using a variety of distributed technologies (CORBA, DCOM) to better plan and implement their business strategies (Rosen et al., 2008). Main drivers and motivation factors for SOA implementation are: increasing agility and flexibility, reuse, rationalizing data, integration and reducing costs, while key challenges in SOA implementation are: business and IT compliance, reusable (agnostic) services, efficient development (faster delivery, lower costs), integration of applications and data, agility and flexibility (Rosen et al., 2008). There are significant challenges in maintaining SOA applications to make sure compliance with business goals when there is a change in business strategy and goals, and especially if applications are designed primarily to meet urgent requests or requirements at the tactical level. Therefore, the increase in business-to-technology compliance is seen as one of the key objectives in the implementation of SOA (Erl, 2007). Accordingly, Markes & Bell (2006, p. 1) define SOA „*as a conceptual business architecture in which the business functionality or application logic is made available to SOA users or consumers as a shared reusable service available on an IT network. Services in an SOA are modules of business application functionality, with exposed interfaces and are invoked by messages*“.

The success of SOA implementation depends to a large extent on the choice of delivery strategy and the way services are identified, using top-down, bottom-up, meet-in-the-middle (also known as outside-in) or middle-out approach (Terlouw et al., 2009; Slimani et al., 2013). Top-down is an approach driven by a business strategy where services are identified, designed, and implemented on the basis of a detailed analysis of business requirements (Arsanjani, 2004; Marks & Michael, 2006; Erl, 2007; Kohlborn et al., 2009). However, the application of this approach often requires too much time, so, when and even if the project is completed, the developed software solution does not meet the new business requirements in an altered business environment (Microsoft, 2006). Therefore, it proved to be impractical, i.e. did not give the desired results (Rosen et al., 2008). In addition, this approach is not suitable if integration with existing systems is necessary, even in the development of trivial software solutions (Zimmermann et al., 2005). The bottom-up approach is based on existing IT resources and is run by IT departments in order to develop reusable services based on existing resources of the organization (Arsanjani, 2004; Marks & Michael, 2006; Erl, 2007; Kohlborn et al., 2009; Valipour et al., 2009; Mirarab et al., 2014). However, this approach has limited success, because

the development of SOA solution without a direct link to the business context and business goals results in a confusing implementation with little relevance to the organization (Microsoft, 2006). In addition, this approach increases the dependence of the service in relation to the existing technological environment (Terlouw et al., 2009) and creates isolated services that are not suitable for reuse, i.e. do not provide benefit from SOA. The meet-in-the-middle approach involves combining and iterative application of top-down and bottom-up approach (Marks & Michael, 2006; Erl, 2007), but existing software systems limit the available options in modeling (Zimmermann et al., 2005), while problems arise when aligning service candidates from the initial top-down phase with created bottom-up services (Terlouw et al., 2009).

The middle-out approach is guided by a strategic vision and business goals, and implemented with several smaller iterative SOA projects, where each individual SOA project is planned and implemented to meet specific business goals and business requirements (Microsoft, 2006). This approach represents a compromise between top-down and bottom-up, as it simultaneously produces both business and information infrastructure aligned with strategic goals and reusable services. The key to simultaneous realizing of these two seemingly irreconcilable goals is the SOA reference architecture, also known as the initial or minimal architecture (Rosen et al., 2008). However, the need for defining common semantics for services of different types (business, application, domain, utility, integration, basic, external...) makes this approach difficult for the implementation of SOA (Slimani et al., 2013). It is interesting that many researchers do not recognize both, the meet-in-the-middle and middle-out approach, or consider them same using them as synonyms (Gimnich & Winter, 2007; Wang et al., 2007; Rosen et al., 2008; Nikravesh et al., 2011; Mircea, 2012; Svanidzaitė, 2012; Mirarab et al., 2014; Pulparambil et al., 2017; Al-Hamed et al., 2018). Some authors additionally use the term hybrid approach (Bani-Ismail & Baghdadi, 2016). Terlouw et al. (2009) and Slimani et al. (2013) have dealt with a detailed analysis of these approaches, identifying significant differences in the number and order of activities, which supports the thesis that these two approaches should be considered separately.

Regardless of the chosen approach to service identification, SOA implementation, as a complex venture, can be difficult to perform well without the use of an adequate SOA methodology. Ramollari et al. (2007) provide a comparative overview of the SOA methodologies. They note that the seven identified methodologies support a middle-out approach: *IBM Service-Oriented Analysis and Design* (SOAD), *IBM Service-Oriented Modeling and Design* (SOMA), *SOA Repeatable Quality* (RQ), *CBDI-SAE Process*, *Service Oriented Architecture Framework* (SOAF), *Service Oriented Unified Process* (SOUP) и *Papazoglou* methodology. Three identified methodologies support a top-down approach: *Mainstream SOA Methodology* (MSOAM), *BPMN to BPEL* и *Steve Jones' Service Architectures*, while none supports bottom-up approach. Rosen et al. (2008) propose their SOA methodology based on a middle-out approach, reference architecture, common semantics, modeling of business processes, Model-Driven Development (MDD), service discovery in design time and service governance. Al-hamed (2018), with the aforementioned, identifies two new methodologies: the *Method Engineering* paradigm proposed by Garo et al. (2011), which allows the definition of new methodologies based on the use of parts of existing methodologies in a given context, and *Improved Methodology* by Emadi et al. (2012), which supports a meet-in-the-middle approach.

On the other side, the way of integration and communication between services has a major impact on the success of the SOA initiative. Due to effective governance is one of the key success factors in the implementation of SOA (Cerny et al., 2017), traditional SOA implementations typically included a use of the ESB. ESB is a universal mediator and an orchestrator in the communication between services of various types, which enables the integration of different applications and technologies with built-in mechanisms for the transformation of messages, registration, monitoring and service governance (Papazoglou & Van Den Heuvel, 2007; Rosen et al., 2008; Rademacher et al., 2017). ESB is a powerful tool that can significantly simplify the SOA implementation if there is enough service available, even to the extent that the notion of SOA is often misidentified with the term ESB (Ouertani, 2015). However, the application of the ESB is not a precondition or guarantee for successful implementation of SOA, because the price to be paid is increasing complexity and poor performance (Rosen et al., 2008; Cerny et al., 2017).

In the digital era, expansion of digital technologies follows the complexity, distribution, and scalability, as well as the increase in the speed of software delivery (Erder & Pureur, 2016). ESB is not designed for cloud (Villamizar et al., 2015; Taibi et al., 2017), therefore it is increasingly criticized as fat, inflexible and difficult to manage (Zimmermann, 2016). A particular problem is the scaling of monolithic applications, where the ESB becomes a bottleneck (Posadas, 2017), which, along with demands for drastic increase in delivery speed, has greatly influenced the evolution of SOA to new implementation approaches, such as service-based approach and microservices (Ford et al., 2017). They replace the ESB as orchestrator by direct communication between the services. Bearing in mind that these two approaches are based on direct communication between the services, some researchers mistakenly view them as the same. However, there is a key difference between them; a service-based approach emphasizes the reuse of the services and resources ("*share-as-much-as-possible*"), while microservices are orientated to the concept of a bounded context ("*share-nothing*" or "*share-as-little-as-possible*") (Richards, 2016; Cerny et al., 2017; Ford et al., 2017; Bogner et al., 2018). This is one of the primary reasons why some authors, who come from the agile community, see microservice architecture (MSA) as a new architectural style (Fowler, 2015; Pahl & Jamshidi, 2016).

3. RESEARCH METHODOLOGY

The first objective of the research is a better understanding of how SOA middle-out strategy delivery and service-based approach can contribute to the harmonization of business and digital strategy in the context of the development of business software systems. The second objective is to identify and address the challenges and best practices, as well as significant advantages and disadvantages related to the architectural and methodological aspects of the software development and delivery process in such a context. In this regard, the following research questions (RQ) were set up:

RQ1. How SOA middle-out strategy delivery and service-based approach can contribute to the harmonization of business and digital strategy?
RQ2. What are the challenges and best practices associated with the application of SOA middle-out and service-based approach?
RQ3. What are the key advantages and disadvantages of implementing SOA middle-out and service-based approach?

To understand better a combination of SOA middle-out delivery strategy and service-based approach, in the context of the development of the business software system using digital technologies, the case study method was applied. In doing so, a descriptive case study is used to better understand the context, the vision of the system and the architectural-methodological aspects of the process of software development and delivery. In addition, an exploratory case study is used to identify the challenges, best practices, as well as advantages and disadvantages of such an approach in a given context. The combination of a descriptive and exploratory case study aims to better understand the problem in its natural context and define the framework for further research (Yin, 2003; Runeson & Höst, 2009).

For the needs of the research, the case of the development of the software system for performance management of business processes (hereinafter System) in the Kompanija Boksit was selected. After the implementation of the ERP system and quality management system for standardization needs (ISO, FSC, HACCP), Kompanija Boksit has identified a need for a more advanced analytical software tool for managing the performance of the business processes. The System is based on the SOA middle-out delivery strategy and service-based approach, and digital technologies, as analytics, cloud and mobile, in the combined ecosystem. A small, dedicated team consisting of 1 to 3 members, with different levels of competence and experience, has implemented the System. The development of the System has implied the more active role of existing IT resources and new digital technologies, such as cloud, mobile, and analytics in redefining and operationalizing of business strategies, especially in the direction of improving business operations, i.e. the way the value is delivered to the customers (Berman, 2012). In this connection, the digital strategy of Kompanija Boksit was the result of the gradual fusion of business and IT strategy and can be seen as increasing the capacity of existing IT resources and potential application of new digital technologies. Therefore, digital strategy of Kompanija Boksit is not hierarchically subordinated and passive in relation to the business strategy, but more proactive, as an integral part of business strategy.

Various data sources, including documents, source code and semi-structured interviews with one of the authors were used in the data collection process. One of the authors, at different positions in Kompanija Boksit, was engaged deeply in defining and operationalization of the business and digital strategy, implementation of ERP system and quality management system. In addition, as business analyst, software architect and full-stack developer, he played a decisive role in the development of the System, especially in the initial phase, where he developed a functional prototype (Dragičević, 2010), and configured the basic elements of the combined DevSecOps (Development, Security & Operations) ecosystem, with secured development, testing and production environment. Key research questions were related to the context, motivation, vision of the system, architectural and methodological aspects of the development process and the delivery of the System. Qualitative data were collected in order to better understand the process of development and delivery, and to identify the challenges, best practices, advantages and disadvantages of SOA application middle-out delivery strategy and service-based approach in aligning the business and digital strategy of Kompanija Boksit. A qualitative analysis has been used to analyze data, because it supports a more detailed description of the observed phenomenon, while qualitative data allow for better insight into the complex processes (Eisenhardt & Graebner, 2007). In order to reduce the risk of bias, two authors have carried out research.

4. EMPIRICAL RESEARCH RESULTS

In accordance with the proposed research method, the results of the descriptive case study were presented first. Results describe the implementation of the SOA middle-out delivery strategy and service-based approach in the context of the development of the System in Kompanija Boksit. After that, the results of the exploratory case study have been presented that identify and address challenges, best practices, as well as the advantages and disadvantages related to architectural and methodological aspects in the application of SOA middle-out delivery strategy and service-based approach in the given context.

4.1. Implementation of SOA middle-out delivery strategy and service-based approach

In this section, the results of a descriptive case study are presented to get the answer to the question: *How SOA middle-out strategy delivery and service-based approach can contribute to the harmonization of business and digital strategy?* The case study describes the context, the vision of the system, and the key architectural and methodological aspects of the development and delivery process of the software in the combined ecosystem.

4.1.1. Context and vision of the system

Profile - Kompanija Boksit was founded in 1959. As a three times winner of the award for the most successful company in the Republic of Srpska, Kompanija Boksit is one of the most important companies in the Republic of Srpska and Bosnia & Herzegovina. From the classic enterprise for research, production and preparation of bauxite ore, the company has diversified and developed other business activities, from mining to food production, with more than 800 employees and annual revenues of over 30 million EUR; it operates globally, while the most important markets are the countries of the western Balkans and the EU. The company is organized on the principle of profit centers - production and service divisions, and supporting departments, including IT department. In order to effectively plan, organize, operationalize and control business processes, modern methods, techniques, technical means, and IT systems are used, such as integrated video surveillance, ERP system and quality management system (QMS).

Motivation - Due to the widely diversified business activities, Boksit is positioned on the market as a bidder and a user of a many of different products and services, and has business connections with a large number of customers and suppliers. Therefore, Kompanija Boksit has identified the need for the development of a software system for performance management of business processes (hereinafter System) to improve its IT resources and operational capabilities in order to more effectively and effectively plan, organize, operationalize and control business processes. Automation of performance management of business processes aimed at more effective operationalization of the business strategy with the improvement of the quality management system, which is based on the requirements of ISO standards, as the basis for continuous monitoring, measurement, and improvement of processes, products, and services.

System vision - The System had to provide a detailed insight into key performance indicators (KIPs) of business processes, including the ability to obtain detailed overview and drill-down options over different periods of time and business analytics, such as employee, organizational unit and partner. The System had to limit access to information in accordance with the positions and authorizations of users using authentication, authorization, and personalization of content. In addition, the possibility of future connection of the System with other systems of customers, suppliers, and websites should be foreseen. The System had to utilize the relevant data from the existing ERP system and other records necessary for calculating KIP's of business processes. It was necessary to ensure the daily update of data and logging all the activities of users on the System. The System had to be implemented as an SOA solution, while, for the interaction of users with the System, development of the front-end web application (Web Portal) was envisaged. In the first phase of the development of the System, the priority was building of a functional prototype for KIP's of procurement and sales process, and after that for production, service and support processes. The System had to be open to new functional and non-functional requirements.

The conceptual architecture of the System - is shown in Figure 1 as a way to, based on the key demands at the highest level of abstraction, present scope of the System and interaction with the environment. It represents the service-oriented vision of the System at the highest level, which is the basis for considering business aspects, in order to identify and build services that are in line with the business and suitable for reuse.

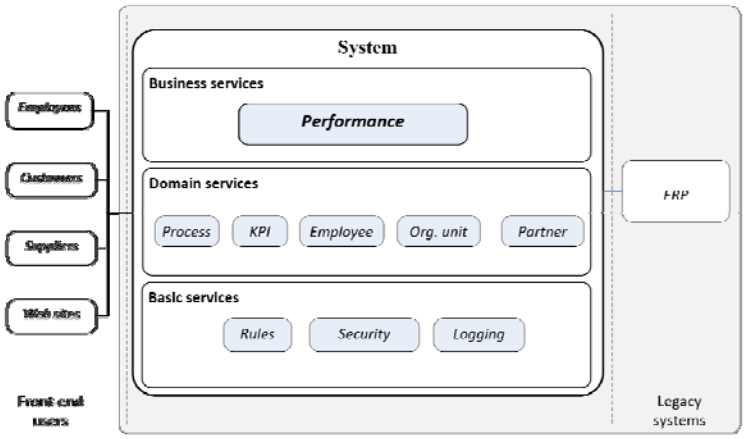

Figure 1: Conceptual architecture of the System
Source: Dragičević (2010), adapted by Authors

4.1.2. Architectural and methodological aspects of software development and delivery process

The process of the System development was based on the SOA methodology proposed by Rosen et al. (2008). The incremental and iterative nature of the development process is noted, where each iteration is treated as a mini SOA project in line with the business strategy, tactics, goals and priorities.

The first iteration was aimed at developing a prototype of minimal functionality, including defining the initial business architecture, identifying the initial set of services, implementing minimal functionality by developing a minimal subset of identified services. In addition, it includes implementing the minimum of the functionality of the front-end web application, defining and configuring the key elements of the development, test and production environments that have enabled the independent delivery of implemented services and front-end web applications, in order to get quick feedback from the users.

Each of the subsequent iterations, based on the user experience and the selected functional and non-functional requirements of the highest priority for implementation, involved the following activities: 1) identifying the need for redefining business architecture, 2) identifying potential new services, need to remove, divide or merge existing services, 3) design and implementation of new services or modification of the functionality of existing services; 4) adding functionality to the front-end web application with the integration of new/modified services, and 5) the independent delivery of new/changed service and/or a front-end web application in the test environment, and/or a

production environment, with the knowledge of user acceptance of the changes. Delivery in a test environment was considered optional, depending on the size and complexity of the increment of functionality. Details of these activities in the development and delivery process, as well as the elements of the test and production environment essential for the delivery, monitoring, and management of services, are described below.

Business Architecture

Business architecture, as the basis for achieving business-IT compliance, provides a basic overview of the resources and processes of the organization that need to achieve operational, tactical and strategic goals. Building business architecture meant defining a following elements: a business motivation model, a value chain, and a business context diagram. In order to link the key business services to the tactics and business goals of the enterprise at the strategic and tactical level, a business motivational model is used, where the modeling of links between services, business goals, strategies and tactics allowing monitoring of service and business compliance (OMG, 2015).

Business services represent IT resources aimed at supporting the implementation of tactics and tactical goals, but also to influence the review of strategy and strategic goals, both at the enterprise level and at the level of profit centers. Presented business motivation model of Kompanija Boksit directly links the business service with tactics, strategy and business goals (Figure 2).

Based on the process model of the Kompanija Boksit, a value chain (Figure 3) is identified that shows the main business processes and their priorities in terms of importance of creating additional value (Porter, 1985). The value chain of Boksit consists of the realization processes, which include procurement, production and service processes, sales and post-sales services.

Finally, the business context diagram is defined (Figure 4) as the first step in the business analysis, which enables an understanding of business interaction between actors and systems, as well as the information they exchange.

Service identification

In order to identify an initial set of potential services, as well as a better understanding of the user-system interaction, a top-level process model is defined describing typical user interaction with the system. Based on business requirements, business context diagrams and process model, different use cases have been identified. In the context of the enterprise, the security aspects of the solution were examined and the initial service model of the top-level software solution architecture, the entity model and the information model were defined.

Security aspects are considered in the initial SOA implementation iteration because security requirements affect the overall SOA design. User authentications is done on the Web Portal, while the Web Portal, as well as the potential systems of customers and suppliers, in the future interactions with the services, must confirm their identity.

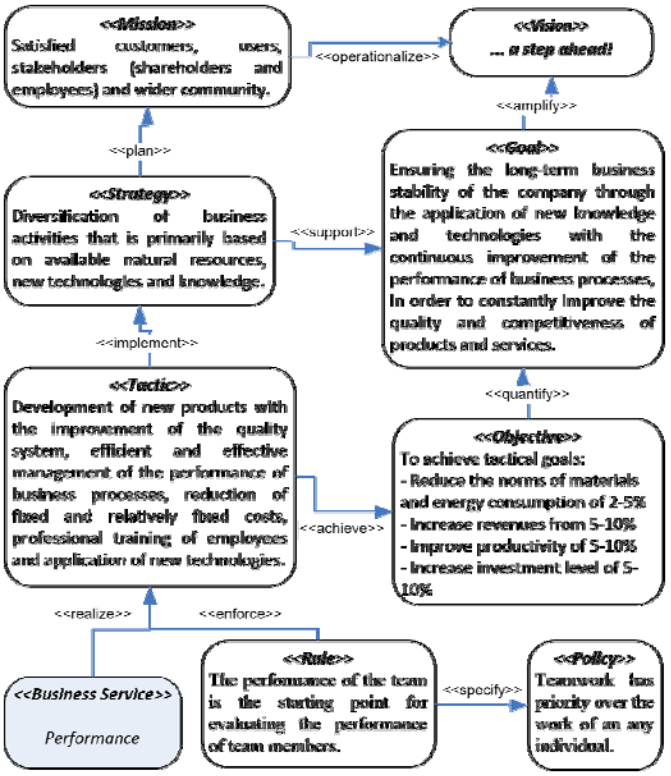

Figure 2: Business motivation model of Kompanija Boksit
Source: Dragičević (2010), adapted by Authors

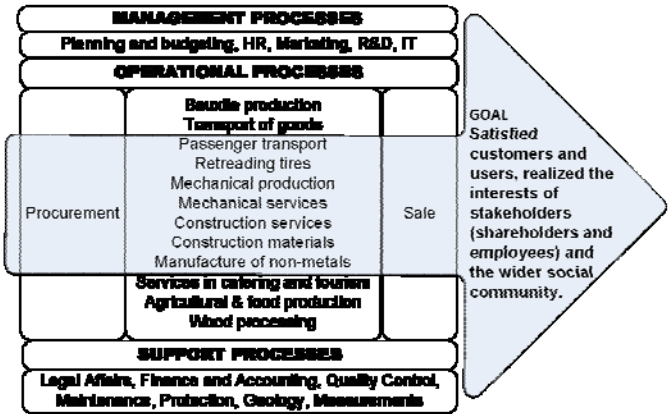

Figure 3: Value chain of Kompanija Boksit
Source: Dragičević (2010), adapted by Authors

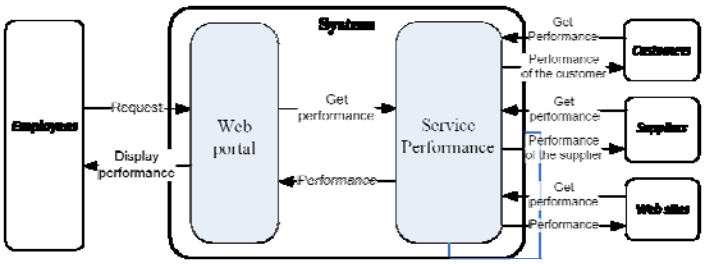

Figure 4: Business context diagram of the System
Source: Dragičević (2010), adapted by Authors

Authorization is based on the mechanism of roles and rules. The basic rule is that the employee using the system, depending on the roles, has the ability only to see own data, data related to the organizational unit it belongs to, as well as to the subordinated organizational units. There is a possibility of defining rules for exceptions, in the direction of the extension these rights, and towards narrowing down these rights.

The service model of the top-level software solution architecture (Figure 5) determines the granularity of the service and supports the integration of the legacy systems (ERP), business services and front-end web application. The initial service model contains a broader set of services (applications, work, domains, services, basic and integration) in relation to the conceptual architecture, describing the main responsibilities of individual services and enabling decision-making regarding the inclusion of certain functionalities in the design and implementation of the service. The application service Portal is responsible for separating the presentation from the business logic, for determining individual rights and user preferences (authorization and personalization), as well as for implementing the logic at the individual user level. This increases the potential for reuse of the service. Considering the context of the enterprise, significant attention is paid to identifying common information that will be exchanged between the services. In this regard, the main entities from the business domain are identified, followed by their connections and information exchanged between the service, which will then be declared in the service interfaces.

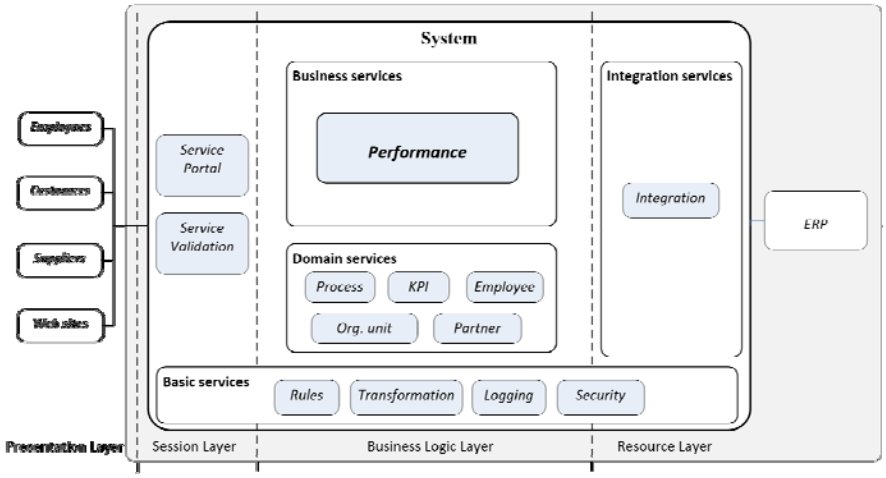

Figure 5: Service model of the top-level software solution architecture
Source: Dragičević (2010), adapted by Authors

Service interface design

For each service selected for implementation in the current iteration, an interface is designed to include a minimal set of operations and documents, i.e. the parameters passed to the operations, and the result that returns the operation (Figures 6 and 7). The basic difference between service operations and object methods is that service operations are more granular. In order to minimize dependencies between services, a simple interaction is designed, especially when it comes to entity services and basic services, in order to preserve their huge potential for reuse. The parameters and results of operations are defined in the light of common semantics, with a minimal set of data, avoiding exposure of information that is not needed or which should not be exposed, with the use of naming conventions that simplify and facilitate communication between the services (CoC - Convention over Configuration). The granularity of the interface, as well as the granularity of the service, is related directly to the potential of reuse of the service, and it differs depending on the purpose of the service. Application, business, and integration services have the highest granularity, domain services are medium-sized, while service and basic services are the least granular. In the design of the integration service, the existing functions and data from the ERP system are transformed into new functionalities and information that contribute to the realization of the strategy and strategic goals. When identifying the need to change the functionality of the service, instead of the versioning, the replacement of the old with the new version of the service is applied, i.e. the old version of the service is discarded, while the new version of the service ensures the functionality for old and new service users.

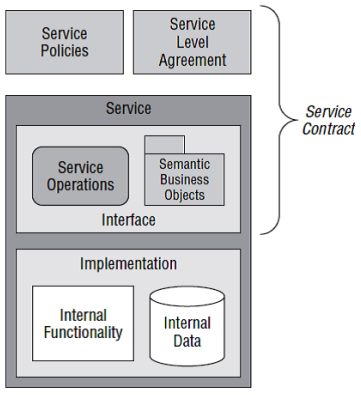

Figure 6: Anatomy of service
Source: Rosen et al. (2008, p. 51)

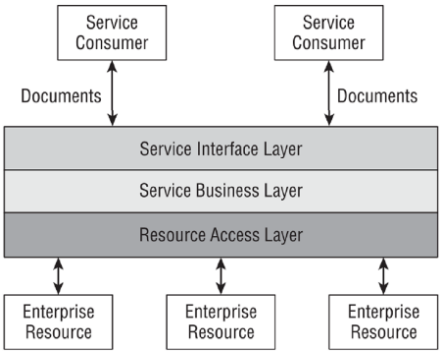

Figure 7: Architecture of service
Source: Rosen et al. (2008, p. 255)

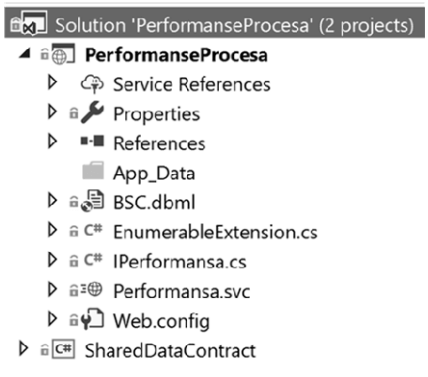

Figure 8: Implementation of service
Source: Kompanija Boksit, source code

Service design and implementation

Services are designed and implemented in accordance with the design principles proposed by Erl (2007). Each service was developed using WCF, .NET technology, and a *3-tier* architectural approach to separate interface, business logic, and access to resources (Figures 7 and 8). Any implemented service can be deployed in a production environment, independently of other services. The services can be accessed through various communication modes that can realize less or more advanced security mechanism for communicating with other services, which are defined in the configuration file. Business, domain and integration services share a common data source (System Database), while basic services have their own data sources using redundant data, which do not share with other services.

Implementation of a service-oriented solution

In order to support the reuse of existing services and enable faster and cost-effective development of the business software based on new or changed business requirements, layered *N-tier* solution architecture is applied (Figure 9). The presentation layer is responsible for adapting the content for a user device, thus making the other layers independent of the device, as well as communicating with the session layer. The session layer is where the system allows multiple interactions with a single user. Services in this layer are responsible for coordinating and managing the user session and session-related user data, authorizing users and applying business rules at the user level, as well as for communicating with the business logic layer. The business logic layer contains the services responsible for implementing business logic and domain entities, and for the availability of their functionality through service interfaces. Services from this layer maintain the integrity of shared resources, enforce system business rules, provide a framework and control for transactions, and provide business services for users. The boundary between the session layer and the layer of business logic enables the separation of enterprise resources and resources required to support the user, and therefore, better protection and governance of enterprise resources. The resource layer is responsible for managing the shared resources of the enterprise. This layer provides access to shared enterprise resources (data, databases and legacy systems). The Web Portal was developed as an independently delivered, monolithic web application, based on the application of .NET and Ajax technology, which at the request of the authorized user provides insight into the performance by combining various tabular and graphic displays, allowing drill-down into details. The link between the Web Portal and the business service is realized through the application service Portal, which has the role of a facade, i.e. separating the presentation layer from the business logic layer. Any iteration in the development process required the alignment of the implementation of new functionalities at the level of the Web Portal and at the level of individual services, in order to allow the rapid delivery of each subsequent increment of functionality.

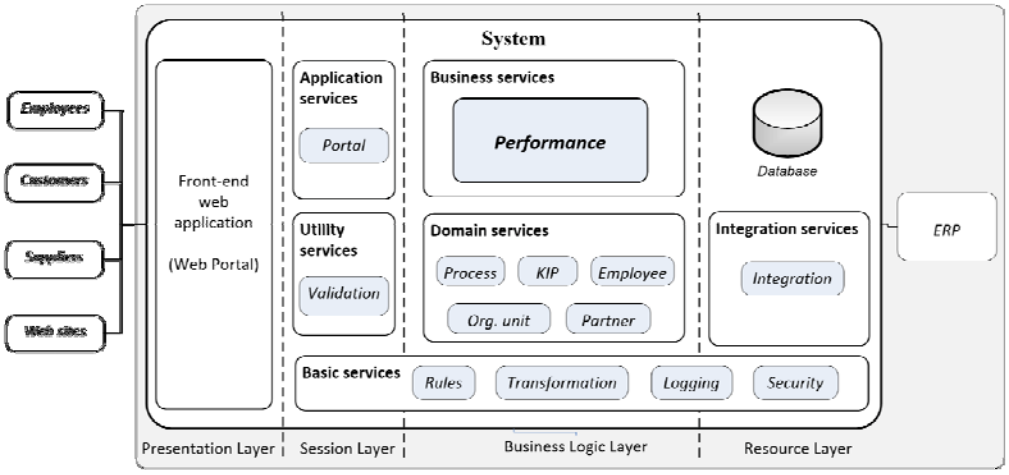

Figure 9: *N-tier* solution architecture
Source: Dragičević (2010), adapted by Authors

Testing, delivery and monitoring in the combined ecosystem

The combined DevSecOps ecosystem contains development, testing, and production environment. Particular attention is paid to security, so the test and production environment is implemented as a private cloud, protected by a firewall, with a Proxy server in the demilitarized zone, while the Web and Database server is behind the firewall in the separated part of the LAN. Scripts and tools were used for delivery, testing, and monitoring. Delivery of services from a development environment into a desired, test or production environment, is realized in two steps: 1) by choosing the appropriate publish profile installation files are generated and 2) by launching the corresponding deploy script simultaneously removing the old one and installing a new version of the services and/or front-end web application, with the simultaneous backup of old and new versions. In the production environment, only one version of the service is active at one point, while the previous versions can be restored at any time using the same deploy script. For monitoring purposes, a tool has been developed that automatically sends e-mail alerts and reports on the realization of scheduled tasks and identified errors.

4.2. Challenges and best practices

This section presents the results of the qualitative analysis of data in order to get the answer to the question: *What are the challenges and best practices associated with the application of SOA middle-out and service-based approach?* A total of 27 challenges and 27 practices related to the application of SOA middle-out and service-based approach have been identified and grouped by activities and artifacts of the development and delivery process in the combined ecosystem (Table 1).

Table 1: Challenges and best practices

Activity	Artefacts	Challenges	Best practices
Context and vision of the system	*«document»* - Conceptual architecture of the system	- Selection of system implementation strategy - Defining external system boundaries - The flexibility of system architecture	- *Lean thinking* (timely thinking and planning) - Early focus on non-functional requirements (quality attributes) - Continuous, incremental value delivery - Minimal documentation
Business architecture	*«document»* - Business motivation model - Value chain - Business context diagram	- Understanding the context and interaction of the system and the environment - Prioritization of business processes and activities - Identification of links between business and IT resources	
Service identification	*«document»* - Model of process - Use cases - Service model of the system architecture - Types and taxonomy of the services	- The interdependencies of the requests - Determining the size of a functional and/or architectural increment - Identifying the services that will implement the increment of the functionality - Determining service granularity - Identifying shared information - Enabling service reuse	- *Model-Driven Development (MDD)* - *Walking skeleton* - *Minimal Viable Architecture (MVA)* - *Minimal Viable Product (MVP)* - *Application of the service design principles* - *Services of different types and granularity* - *Common service semantics* - *Service replacement (no versioning)* - *Independently delivered services* - *3-tier service architecture* - *N-tier system architecture* - *Continuous Architecting (CA)* - *Continuous Integration (CI)* - *Continuous Delivery (CD)* - *Continuous Refactoring (CR)* - *Convention over Configuration (CoC)* - *Micro-team (1-3 members)*
Service interface design	*«source code»* - The mode of communication between the service - End-points - Operations - Documents (parameters and results of operations)	- Interdependence of services, especially between business and integration services - Choosing the mode of communication between the services - Determining the granularity of the service interface - Defining the scope and visibility of the service - Choice of approach to service versioning	
Service design and implementation	*«source code»* - Business rules - Business logic - Service local data - Shared data	- Selecting a type of service implementation - Preserving the autonomy of the services - Managing shared resources	
Implementation of a service-oriented solution	*«source code»* - Front-end communication with services *«document»* - Service layout by layers of *N-tier* architecture	- Defining internal system boundaries - Deploying services to the layers of *N-tier* system architecture - Scalability of the system - User experience	
Testing, delivery and monitoring in a combined ecosystem	*«tool»* - Tools and scripts for build, publish, test and deploy *«report»* - Reports of the execution of scheduled tasks - Error reports	- Defining and configuring the development, testing, and production environment in combined ecosystem - Effective testing - Monitoring of services	- DevSecOps ecosystem - Virtualization - Private cloud - Combined, semi-automated testing - Semi-automated delivery - Development of tools for monitoring and error reporting

Source: Authors

4.3. Advantages and disadvantages

This section presents the results of the qualitative analysis of data in order to get the answer to the question: *What are the key advantages and disadvantages of implementing SOA middle-out and service-based approach?* In total, eight key advantages and seven key disadvantages were identified that are associated with the application of SOA middle-out and service-based approach (Table 2).

Table 2: Advantages and disadvantages

Advantages	Disadvantages
1. Better alignment of business and IT resources Direct linking of business services with business strategy, tactics, and goals, identification of services at the enterprise level, as well as iterative approach with small increments and daily deliveries, contribute to increasing the alignment of business and IT resources, i.e. harmonizing business and digital strategy.	*1. Up-front architecture and design* The lack of all relevant information, when it comes to expected functionality and user experience in the initial iteration of the development process, increases the risk of identifying and realizing services, or their functionalities, that will require a big refactoring or prove to be unnecessary.
2. Increasing agility and flexibility The great potential for reuse of application, basic, infrastructure and integration services, due to their autonomy and the possibility of independent delivery, contributes to increasing of agility and flexibility at enterprise level and software development and delivery process level.	*2. Increased risk of data inconsistency* Data redundancy contributes to an increase in service autonomy; however, redundancy of data, with service delivery at short time intervals and distributed transactions, increases the risk of inconsistency in the data.
3. Agilization of SOA methodology The application of various agile, lean and continuous practices with SOA methodology has made it possible to reduce increments, increase delivery speed to the level of multiple daily software deliverables, and to get faster feedback from users.	*3. Poor user experience* Multiple daily deliveries to production environment lead to frequent interruptions of the current sessions or user activities that have a negative impact on the user experience in working with the system. On the other hand, the number of users affected by service delivery depends directly upon the scope, visibility, and granularity of the delivered services, while the delivery of monolithic front-end web application can affect all active users.
4. Reducing the complexity of the system The greater granularity of application, business and integration services prevents an uncontrolled increase in the number of services, which contributes to reducing the complexity of the system architecture.	*4. Shared resource scaling problem* Shared resources, and in particular shared SQL databases, are not suitable for scaling.
5. Increasing ability to scale services Independently deliverable services enable easier scaling, both of the individual parts (services) and the system as a whole.	*5. Lack of competent people* It is difficult to find and train, and it is even more difficult to retain people with broad knowledge, competencies, and experience, as well as exceptional discipline and professionalism that are necessary for the role of business analysts, architects and/or full-stack developers, given the complexity that adds the combined DevSecOps ecosystem. The lack of competent people has a negative impact on the ability of the team scaling.
6. Fewer errors and bugs A micro-team with good communication, a service design that supports testing at the service level, small increments and fast deliveries on a daily basis, contribute to reducing the number, significance, and consequence of errors and bugs, and their faster resolution.	*6. Redundancy of data* More discipline is needed in order to preserve the consistency of the entire system due to redundant data.
7. Faster delivery of functionality Independent service delivery, small increment and fast feedback from the user support faster delivery of expected functionality to users.	*7. Difficult testing of the whole system* Increase in a number of services, frequent changes of the interfaces and replacement of services with discarding old versions make it more difficult to test the entire system.
8. Increasing security An early focus on quality attributes, including security and personalization, layered architecture and a combined ecosystem with a test and production environment in private cloud, contribute to increasing of system security.	

Source: Authors

5. DISCUSSION

This section provides answers to research questions, with a reference to the transformation of the applied traditional SOA methodology, which is motivated by the increase in the speed of software delivery. After that, possibilities for further research and limitations of the conducted research are presented.

Answers to research questions:

The results of the research indicate that, in order to harmonize business and digital strategy of the enterprise in the context of the development of a business software system, it is possible to successfully apply the SOA middle-out and service-based approach by combining 1) the traditional SOA methodology that supports middle-out delivery strategy (Rosen et al., 2008), 2) services of different type and granularity, 3) the appropriate practices of agile architecture (Dragičević & Bošnjak, 2018), and 4) DevSecOps combined ecosystem, in a way that simultaneously supports reuse of services and fast, even multiple daily, delivery of services and/or front-end web application. In addition, a special focus

was on the first iteration, which is based on the initial context and vision of the system, as well as the initial business architecture and service model, that produces the initial software architecture (walking skeleton) and functional prototype. In each subsequent iteration, using as small as possible increments, new/changed services and/or front-end web applications with new or modified functionality were delivered fast with fast user feedback. However, the described approach is not easy to implement, which best illustrates the 27 identified challenges associated with the different activities of the development and delivery process in the combined ecosystem. Overcoming these challenges required the implementation of 27 identified practices of agile architecture in the combined ecosystem, that enable fast simultaneous incremental delivery of required functionality, and to provide an agile, decentralized, resilient architecture that supports the reuse of services. The described approach has its advantages and disadvantages. The key identified advantages are better compliance of business and IT resources and faster delivery of functionality as key factors for the alignment of business and digital strategy, which is in line with the results of Yeow et al. (2018), Grover & Kohli (2013) and Mithas et al. (2013). Other identified advantages are increasing agility and flexibility both at the enterprise level and at the level of the development process, reducing the complexity and number of errors and increasing the security and potential for service scaling. On the other hand, a key disadvantage of this approach is a lack of competent, experienced people who can successfully take on multiple roles, from business analysts, through the software architect, and to a full-stack developer. In addition, the price to be paid is a significant up-front and a continuous focus on architecture and design, data redundancy and the difficulty of maintaining their consistency, with additional problems in scaling shared resources, as well as poor user experience and difficult testing of the entire system due to frequent deliveries of modified/new services and/or front-end web application in the combined ecosystem.

Transformation of traditional SOA methodology:
Development of a service-oriented software system is based on the existing traditional SOA methodology (Rosen et al., 2008). However, differences in the realization of certain activities are identified, especially considering fact that the used SOA methodology was presented in 2008, prior to the emerging of Continuous Delivery (CD) practice in 2010, and an expansion of the use of digital technologies, which caused an increase of speed of software delivery. In this regard, the following key differences are noted in relation to the approach proposed by Rosen et al. (2008):

- Continuously and timely thinking and planning, instead of up-front plans for defining priorities and software architecture design.
- Minimal documentation and increased use of source code as a source of documentation, which includes interfaces and service operations, semantic information model, service inventory and service architecture.
- Micro-team responsible for the complete life cycle of the service and software system, with broad competencies of team members, who are able to simultaneously realize multiple roles, from business analysts, software architect, to the full-stack developer, instead of a large team of specialists in many different fields.
- Agilizing SOA methodology with the application of various agile, lean and continuous practices, instead of the traditional SOA methodology. Even though agilization of the traditional SOA methodology by agile, lean and continuous practices can be considered as a process opposite to the traditionalization of the agile process of software development by using architectural practices (Matković et al., 2011), the ultimate goals are the same.
- Independently delivered services, fast delivery and user feedback, instead of monolith application.
- Replacing the services, instead of versioning the services.
- The simpler DevSecOps ecosystem, supported by scripts and tools for semi-automated testing, delivery, and monitoring, instead of a complex production middleware that supports ESB service orchestration and automation of business processes.

Opportunities for further follow-up research:
Realized research is based on the single case of the development of a business software system in one company, so additional empirical researches are proposed at the successful and unsuccessful implementation of SOA middle-out and service-based approach in the digital era. Given that there is no consensus on how to use existing agile methods and practices in the development of service-oriented systems (Carvalho & Azevedo, 2013), further research should be done regarding the possible positive impact of agile, lean and continuous principles, methods, techniques and practices on the transformation of traditional SOA methodologies for the development of business software systems in the digital era. Bearing in mind the perceived disadvantages of SOA middle-out and service-based approach in the digital era, future research should identify a possibilities for overcoming them by integrating microservices and a service-based approach in the context of the development of business software systems.

Research limitations:
There are three limitations of the research to be taken into account. First, the research is based on a specific example of the development of business software in just one company; therefore, two separate case studies were conducted - first a descriptive case study, in order to describe in detail and better understand the observed phenomenon in its natural environment, and then an exploratory case study, in order to identify the challenges, practices, advantages and disadvantages of the applied architectural-methodological approach, and identify the possibilities for further research. Second, one of the authors in various positions in Kompanija Boksit played a roles both in defining business and digital strategy, as well as in implementing System. Therefore, in order to reduce the risk of bias, more researchers are involved in a planning and implementation of the research. Third, the research is based exclusively on qualitative data,

therefore, for the needs of qualitative analysis, various sources of qualitative data, including documents, source code, and semi-structured interviews, have been used.

CONCLUSIONS

The paper presents the results of empirical research that describe the key architectural and methodological aspects of the implementation of SOA middle-out delivery strategies and service-based approach, in the context of business software system development, that are significant for harmonization of business and digital strategy, as well as identified challenges, best practices, the advantages and disadvantages of such an approach. The results of the research indicate that effective implementation of SOA middle-out delivery strategies and service-based approach can help businesses to respond faster, more efficiently and more effectively to the changing business environment in the digital era. The importance of SOA implementation in enterprises is that it is a good opportunity to define, revise and harmonize business and digital strategy, redefine strategic and/or tactical goals, and build or improve the business architecture. Implementation of SOA in the digital edge is an evolutionary process, which starts with a better understanding of the context, based on existing business and IT resources, in order to upgrade them iteratively. The result of this process is building new IT resources and extending the functionality of existing ones, adding new value to them, avoiding duplication of responsibility and inconsistency. To do all this, business architecture and service design process must be mutually supportive and coordinated. There must be a clear link indicating that the resulting business architecture elements directly support the design of the services. Starting from vision, business strategy, strategic goals and business resources, on the one hand, and digital strategies and available IT resources, on the other hand, through business processes, which are most important for achieving business goals, we can identify desired functionalities, services, business entities and the information by which these functionalities are implemented.

Research contributions are a better understanding of the relationship between business and digital strategy in the context of SOA implementation, the challenges and best practices, as well as the advantages and disadvantages, related to the architectural and methodological aspects of the implementation of the SOA middle-out delivery strategy and service-based approach in the digital era. The findings will contribute to the discussion of the evolution of SOA in the digital era toward increasing of agility and reducing of complexity, in order to more effectively align business and digital strategy, and help practitioners to more effectively implement SOA initiatives in the development of business software systems.

Future research should focus on additional empirical research related to application of SOA middle-out and service-based approach in the digital era, including a transformation of traditional SOA methodologies using an agile approach. In addition, future research should aim toward identifying a possibility of combining microservices and service-based approach in the context of the development of business software systems.

REFERENCES

Al-Hamed, F., Al-Doweesh, S., Al-Omar, R., Al-Doweesh, W., & Najjar, A. (2018, April). Service Oriented Software Engineering (SOSE): A Survey and Gap Analysis. In *2018 21st Saudi Computer Society National Computer Conference (NCC)* (1-6). IEEE.

Arsanjani, A. (2004). Service-oriented modeling and architecture. *IBM developer works,* (January), 1–15.

Bani-Ismail, B., & Baghdadi, Y. (2016). SOA Maturity Models as guidance to select service identification methods: A research agenda. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)* (pp. 1-6). IEEE.

Berman, S. J. (2012). Digital transformation: opportunities to create new business models. *Strategy & Leadership*, *40*(2), 16-24.

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital business strategy: toward a next generation of insights. *MIS quarterly*, 471-482.

Bogner, J., Zimmermann, A. & Wagner, S. (2018) 'Analyzing the Relevance of SOA Patterns for Microservice-Based Systems', Zeus 2018, 9(February), pp. 9–16.

Carvalho, F., & Azevedo, L. G. (2013). Service agile development using XP. In *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering* (254-259). IEEE.

Cerny, T., Donahoo, M. J., & Pechanec, J. (2017). Disambiguation and comparison of soa, microservices and self-contained systems. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*(pp. 228-235). ACM.

Dragičević, Z. (2010). Implementation of SOA using .NET and Ajax technologies (Master's thesis), Faculty of Economics, Subotica.

Dragičević, Z. & Bošnjak, S. (2018). Agilna arhitektura u eri digitalizacije - trendovi i prakse. In *Strategic Management and Decision Support Systems in Strategic Management* (98–116). Subotica: Faculty of Economics in Subotica.

Eisenhardt, K. M., & Graebner, M. E. (2007). Theory building from cases: Opportunities and challenges. *Academy of management journal*, *50*(1), 25-32.
Emadi, M., Jazi, M. D., Moghadam, R. A., & Bahredar, F. (2012, May). An improved methodology for service oriented architecture. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (Vol. 2, pp. 350-354). IEEE.
Erder, M., & Pureur, P. (2016). What's the Architect's Role in an Agile, Cloud-Centric World?. *IEEE Software*, *33*(5), 30-33.
Erl, T. (2005). *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, Upper Saddle River: Prentice Hall.
Erl, T. (2007). *SOA: Principles of Service Design, (Vol. 1).* Upper Saddle River: Prentice Hall.
Fitzgerald, M., Kruschwitz, N., Bonnet, D., & Welch, M. (2014). Embracing digital technology: A new strategic imperative. *MIT sloan management review*, *55*(2), 1.
Ford, N., Parsons, R. & Kua, P. (2017). *Building Evolutionary Architectures: supporting constant change*. O'Reilly Media, Inc.
Fowler, M. (2015) Microservice Trade-Offs. Retrieved January 18, 2018 from: https://martinfowler.com/articles/microservice-trade-offs.html
Garro, A., Russo, W., & Tundis, A. (2011). Developing Service-Oriented Applications: a method engineering based approach. In *Proceedings of the International Conference on Semantic Web and Web Services (SWWS)* (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
Gimnich, R., & Winter, A. (2007). SOA migration: approaches and experience. *Softwaretechnik-Trends*, *27*(1), 13-14.
Grover, V., & Kohli, R. (2013). Revealing your hand: caveats in implementing digital business strategy. *Mis Quarterly*, 655-662.
Kohlborn, T., Korthaus, A., Chan, T., & Rosemann, M. (2009). Identification and analysis of business and software services—a consolidated approach. *IEEE Transactions on Services Computing*, *2*(1), 50-64.
Marks, E. & Michael, B. (2006). *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology Service-Oriented Architecture*. John Wiley & Sons.
Matković, P., Tumbas, P., & Sakal, M. (2011). The RSX model: Traditionalisation of agility. *Strategic Management*, *16*(2), 74-83.
Microsoft (2006) Enabling 'Real World SOA' through the Microsoft Platform. Retrieved March 15, 2018 from: https://www.brightts.com/course/downloads/1.pdf
Mirarab A., Fard N. G. & Kenari, A. (2014). A New Framework for Service Identification in SOA. *Global Journal of Computer Science and Technology*, *14*(5).
Mircea, M. (2012). SOA adoption in higher education: a practical guide to service-oriented virtual learning environment. *Procedia-Social and Behavioral Sciences*, *31*, 218-223.
Mithas, S., Tafti, A., & Mitchell, W. (2013). How a firm's competitive environment and digital strategic posture influence digital business strategy. *MIS quarterly*, 511-536.
Nahrkhalaji, S. S., Shafiee, S., Shafiee, M., & Hvam, L. (2018). Challenges of Digital Transformation: The case of the Non-Profit Sector. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (1245-1249). IEEE.
Nikravesh, A., Shams, F., Farokhi, S., & Ghaffari, A. (2011, October). 2PSIM: two phase service identifying method. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (625-634). Springer, Berlin, Heidelberg.
OMG (2015) Business Motivation Models, Object Management Group. Retrieved March 21, 2019 from: https://www.omg.org/spec/BMM/1.3/pdf
Pahl, C., & Jamshidi, P. (2016, April). Microservices: A Systematic Mapping Study. In *CLOSER (1)* (137-146).
Papazoglou, M. P., & Van Den Heuvel, W. J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, *16*(3), 389-415.
Porter, M. E. (1985). *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press.
Posadas, J. V. (2017, August). Application of mixed distributed software architectures for social-productive projects management in peru. In *2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)* (1-4). IEEE.
Pulparambil, S., Baghdadi, Y., Al-Hamdani, A., & Al-Badawi, M. (2017). Exploring the main building blocks of SOA method: SOA maturity model perspective. *Service Oriented Computing and Applications*, *11*(2), 217-232.
Rademacher, F., Sachweh, S., & Zündorf, A. (2017, April). Differences between model-driven development of service-oriented and microservice architecture. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)* (38-45). IEEE.

Ramollari, E., Dranidis, D., & Simons, A. J. (2007, June). A survey of service oriented development methodologies. In *The 2nd European Young Researchers Workshop on Service Oriented Computing* (Vol. 75).
Richards, M. (2016). *Microservices vs. Service-Oriented Architecture*. O'Reilly Media, Inc.
Rosen, M., Lublinsky, B., Smith, K. T. & Balcer, M. J. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley Publishing, Inc.
Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, *14*(2), 131.
Sebastian, I., Ross, J., Beath, C., Mocker, M., Moloney, K., & Fonstad, N. (2017). How Big Old Companies Navigate Digital Transformation. *MIS Quarterly Executive,* (September), 197–213.
Šereš, L. & Tumbas, P. (2014). ERP & Globalization: Challenges and Responses. *Strategic Management*, 19(4), pp. 50–57.
Slimani, S., Baïna, S., & Baïna, K. (2013). Toward a semantic SOA implementation strategy. In *2013 3rd International Symposium ISKO-Maghreb* (1-4). IEEE.
SOA Manifesto (2013). Retrieved March 12, 2019 from: http://www.soa-manifesto.org/SOA_Manifesto.pdf
Svanidzaitė, S. (2012). A comparison of SOA methodologies analysis & design phases. In *Baltic DB & IS 2012, Tenth International Baltic Conference on Databases and Information Systems* (202-207).
Taibi, D., Lenarduzzi, V., Pahl, C. & Janes, A. (2017). Microservices in agile software development. In Proceedings of the XP2017 Scientific Workshops on - XP '17 (1–5). ACM.
Terlouw, J., Terlouw, L., & Jansen, S. (2009). An assessment method for selecting an SOA delivery strategy: determining influencing factors and their value weights. In *Proceedings of the Busital workshop*.
Valipour, M. H., AmirZafari, B., Maleki, K. N., & Daneshpour, N. (2009). A brief survey of software architecture concepts and service oriented architecture. In *2009 2nd IEEE International Conference on Computer Science and Information Technology* (34-38). IEEE.
Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)* (583-590). IEEE.
Wang, X., Hu, S. X., Haq, E., & Garton, H. (2007, June). Integrating legacy systems within the service-oriented architecture. In *2007 IEEE Power Engineering Society General Meeting* (1-7). IEEE.
Westerman, G., Bonnet, D. & McAfee, A. (2014). *Leading digital: Turning technology into business transformation*. Harvard Business Press.
Westerman, G., Calméjane, C., Bonnet, D., Ferraris, P., & McAfee, A. (2011). Digital Transformation: A roadmap for billion-dollar organizations. *MIT Center for Digital Business and Capgemini Consulting*, *1*, 1-68.
Yeow, A., Soh, C., & Hansen, R. (2018). Aligning with new digital strategy: A dynamic capabilities approach. *The Journal of Strategic Information Systems*, *27*(1), 43-58.
Yin, R. K. (2003). *Case Study Research: Design and Methods*. Sage Publications, p. 53.
Zimmermann, O. (2017). Microservices tenets. *Computer Science-Research and Development*, *32*(3-4), 301-310.
Zimmermann, O., Schlimm, N., Waller, G., & Pestel, M. (2005). Analysis and Design Techniques for Service-Oriented Development and Integration. In *GI Jahrestagung (2)* (606-611).