# Impact of SOA on Information System Development

**Matjaz B. Juric**
Faculty of Computer and Information Science
University of Ljubljana
Trzaska cesta 25, 1000 Ljubljana

**Abstract**. *Service Oriented Architecture (SOA) has been one of the most important topics in enterprise application development in the last years. SOA has introduced important changes to the architecture of business applications. The Service Component Architecture and the composite application approach with BPMN, BPEL, ESB, and other important pieces have made applications loosely-coupled and flexible. In the paper we give an overview of the SOA, describe the concepts and SOA building blocks. We elaborate on the value and impact of SOA on information system development and outline the reach of SOA towards information systems, business processes, governance, and organization.*

**Keywords.** SOA, IS development, BPEL, BPMN, ESB

## 1 Introduction

SOA provides technical architecture to develop end-to-end support for business processes. SOA achieves this objective by exposing organization's IT assets as reusable business services, which can be composed into processes on one hand and can integrate and communicate more easily on the other hand.

From the bottom-up perspective, SOA is an integration architecture. It provides technologies and approaches for systematic integration of existing applications and development of new solutions. With SOA software architects develop a high-level integration architecture that uses common concepts to share data, information, and business logic between applications in a controlled, transactional manner using a service bus are other supporting technologies, such as rules engines, registries and repositories, etc. SOA is based on typed communication with messages that are based on common schemas. In new generation of SOA, business events are introduced as well. They enable an alternative approach to the realization of one of the most important goals of SOA – loose coupling. Loose coupling is an approach where different software services and components share the lowest denominator of dependencies. This makes application architecture more robust and resistant to changes. This will allow applications, components and services to evolve and change without or with minimal effects on the other applications, components and services.

SOA is also an architecture for designing, automating and optimizing business processes. The objective of SOA is to provide end-to-end automation of business processes. Business processes in SOA are based on composition of services and processes using programming-in-the-large technologies, most importantly BPEL and Executable BPMN.

In the rest of this article we will look at business and IT agility (Section 2), define the SOA (Section 3), summarize the concepts (Section 4), explain the SOA building blocks (Section 5), talk about inception (Section 6) and value of SOA (Section 7). Finally we will describe the changes in the development approach (Section 8) and provide a conclusion (Section 9).

## 2 Business and IT Alignment

The business system usually evolves with a different pace to that of the information system. Over time this has resulted in an important loss of alignment between the business system and the information system. This has resulted in applications that do not fully support business tasks, and which have again hindered the evolution of business processes. The consequence has been less flexible and adaptable organizations with less competitive power on the market. Only companies where applications can be quickly and efficiently adapted to changing business needs can stay competitive on the global market [1].

An IT gap is mainly a consequence of the inability of application developers to modify and adapt the applications to business requirements quickly and efficiently.

The main reason probably hides in the fact that in the past neither programming languages and technologies nor architectural design could have anticipated changes. Existing applications had been designed to last. They had been developed in a tightly coupled fashion, which makes changes to specific parts of applications very difficult. Because of dependencies such changes usually have several, often unpredictable, consequences. In addition to the complexity and size of the modification, an important factor is also the state of the application being modified. If an application has a well-defined architecture and has been constructed keeping in mind future modifications, then it will be easier to modify. However, each modification to the application makes its architecture less robust with respect to future changes. Applications that have been maintained for several years and have gone through many modifications usually do not provide robust architecture anymore (unless they have been refactored constantly). Modifying them is difficult, time consuming, and often results in unexpected errors [2].

We have seen that there are at least three important forces, which have to be considered:

- Alignment between the business and IT, which is today seen as one of the most important priorities.
- Complexity of existing applications and the overall IT architecture. Modifying them is a complex, difficult, error-prone, and time-consuming task.
- Indispensability of existing applications. Companies rely upon existing applications and very often their core business operations would be jeopardized if existing applications fail.

This makes the primary objective of information systems—to provide timely, complete, and easy to modify support for business processes—even more difficult to achieve [3].

## 3 Service-Oriented Architecture

The alignment of business and IT is very difficult to achieve using traditional approaches. However, if a mediation layer between the business and the information system is introduced, the alignment between business and IT becomes more realistic—meet the *Service-Oriented Architecture (SOA).*

To manage problems related to changing requirements, developments in technology and integration of different methods have been proposed and used over time. A Service-Oriented Architecture is the latest architectural approach related to the integration, development, and maintenance of complex enterprise information systems [4].

SOA is not a radically new architecture, but rather the evolution of well-known distributed architectures and integration methods. Integration between applications has evolved from early days into well-defined integration methods and principles, often referred to as Enterprise Application Integration (EAI). EAI initially focused on the integration of applications within enterprises (intra-EAI). With the increasing need for integration between companies (business-to-business), the focus of EAI has been extended to inter-EAI.

SOA improves and extends the flexibility of earlier integration methods (EAI) and distributed architectures, and focuses on the reusability of existing applications and systems, efficient interoperability and application integrations, and the composition of business processes out of services (functionalities) provided by applications. An important objective of SOA is also the ability to apply changes in the future in a relatively easy and straightforward way [4].

SOA defines the concepts, architecture, and process framework, to enable the cost-efficient development, integration, and maintenance of information systems by reducing complexity, and stimulation of integration and reuse. Let us look at the definition of SOA, as provided by Bernhard Borges, Kerrie Holley, and Ali Arsanjani:

SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions.

## 4 Concepts

SOA is more than just a set of technologies. SOA is not directly related to any technology, although it is most often implemented with Web Services. Web Services are the most appropriate technology for SOA realization. However, using Web Services is not adequate to build SOA. We have to use Web Services according to the concepts that SOA defines [5]. The most important SOA concepts are:

- Services and service abstraction
- Self-describing, standardized interfaces with coarse granulation
- Exchange of messages
- Support for synchronous and asynchronous communication
- Loose coupling
- Reusability
- Service registries and repositories
- Quality of Service
- Composition of services into business processes

### Services

Services provide business functionalities, such as an application for business travel, an application for a loan, and so on. This differs considerably from technology-oriented functionalities, such as retrieving or updating a table in a database. Services in SOA must provide business value, hide implementation details, and be autonomous. Services should be abstract and autonomous. Service consumers are software entities, which call the service and use its functionality [2].

### Interfaces

Service consumers access the service through its interface. The interface of a service defines a set of public operation signatures. The interface is a contract between the service provider and a service consumer.

The interface is separated from the implementation, is self-describing, and platform independent. Interface description provides a basis for the implementation of the service by the service provider and a basis for the implementation of the service consumers. Each interface defines a set of operations. In order to define business services, we need to focus on the correct granulation of operations, and we should standardize interfaces. SOA services are best modeled with coarse granulation [2].

### Messages

Operations are defined as a set of messages. Messages specify the data to be exchanged and describe it in a platform, and language, independent way using schemas. Services exchange only data, which differs considerably from object-oriented and component approaches, where behavior (implementation code) can also be exchanged. Operations should be idempotent (an operation is idempotent if repeated invocations have the same effect as one invocation). WSDL is a service description language that meets SOA criteria [2].

### Synchronicity

Service consumers access services through the service bus. This can be either a transport protocol, such as SOAP, or an ESB. Service consumers can use synchronous or asynchronous communication modes to invoke the operations of services. In synchronous mode, a service operation returns a response to the service consumer after the processing is complete. The service consumer has to wait for the completion. Usually we use the synchronous mode with operations in order to complete processing in a short time. In an asynchronous mode, a service operation does not return a response to the consumer, although it may return an acknowledgement so that the consumer knows that the operation has been invoked successfully. If a response is needed, usually a callback from the service to the consumer is used. In such a scenario, a correlation between messages is needed [2].

### Loose Coupling

Through the self-describing interfaces, coarse granulation, exchange of data structures, and support for synchronous and asynchronous communication modes, a loose coupling of services is achieved. Loosely coupled services are services that expose only the necessary dependencies and reduce all kinds of artificial dependencies. This is particularly important when services are subject to frequent changes. Minimal dependencies assure us that there will be minimal number of changes required to other services when one service is modified. Such an approach improves robustness, makes systems more resilient to change, and promotes the reuse of services [2].

### Reusability

SOA is about the consolidation of functionalities. Therefore, the common goal is to have a single service for each business functionality. In other words, we should not allow having more than one service with equal or similar functionalities. To achieve this it is essential to reuse services in different contexts. Reuse is not easy to achieve. First, we have to develop services that are general enough to be useful in different scenarios. Second, developers should first look at existing services, before developing a new one. If an existing service fulfills the need, they should reuse it. Reuse is fostered by registries and repositories [6].

### Registries and repositories

To simplify and automate searching for the appropriate service, services are maintained in service registries, which act as directory listings. Service providers publish services in registries; service consumers look up the services in the registries. Lookup can be done by name, service functionality, or business process properties. UDDI is an example of a service registry. Service registries can improve reuse. In addition to registries, repositories are becoming important for storing artifacts, such as WSDL interfaces, XML schemas, and so on. Registries and repositories play an important role in SOA governance [6].

### Quality of Service

Services usually have associated Quality of Service attributes. Such attributes include security, reliable messaging, transaction, correlation, management, policy, and other requirements. The infrastructure must provide support for these attributes. Quality of Service attributes are often important in large information systems. In Web Services, Quality of Service attributes are covered by WS-* specifications, such as WS-Security, WS-Addressing, WS-Coordination, and so on. Quality of Service is also provided by the ESB [7].

### Composition of services into business processes

The final, and probably the most important, SOA concept is the composition of services into business processes. Services are composed in a particular order and follow a set of rules to provide support for business processes. The composition of services allows us to provide support for business processes in a flexible and relatively easy way. It also enables us to modify business processes quickly and therefore provide support to changed requirements faster and with less effort. For composition, we will use a dedicated language, BPEL, and an engine on which business process definitions will be executed. Only when we reach the level of service composition can we realize all the benefits of SOA [8].

## 5 SOA Building Blocks

Let us now have a closer look at the SOA building blocks that enable us to realize the above-mentioned concepts [9, 13]:

**Service Component Architecture (SCA)** defines a programming model for composite SOA applications. SCA is based on the idea of service composition (orchestration). SCA provides a model for the composition of services and for the creation of service components, including the reuse of existing applications within SCA composites.
**BPEL**: This is for business process automation with service composition.
**Services**: This is for achieving modular and flexible architecture. For service development, Web Services technology is usually used.
**Enterprise Service Bus** (**ESB**): It provides a means for services and processes to communicate, and enables management and control over the communication. ESB is the backbone of SOA.
**Registries and repositories**: They are central directories of services and useful for locating and reusing services, as well as SOA governance.
**Human task support**: Business processes often involve human interaction. SOA supports human interactions in different ways, such as **WS-HumanTask** and **BPEL4People**. Human task support is related to Identity Management.
**Process monitoring or Business Activity Monitoring** (**BAM**): It allows the monitoring of the execution of processes, such as total execution time, average execution time, execution time of certain activities, and so on. It also allows us to monitor the **Key Performance Indicators** (**KPIs**), which is particularly interesting for management, as it allows them to understand better how the business operations perform.
**Business Rules Management Systems** (**BRMS**) **or Rule Engine**: This is a central place for managing business rules. With BRMS we can put business rules into a central location instead of hard coding them.
**Adapters**: They provide easy access not only to external systems, such as ERP, CRM, SCM, but also DBMS systems.
A very important aspect of SOA is **SOA governance**. SOA is a complex architecture, which has to be governed in order to be consistent. SOA governance is a set of activities related to control over services and processes in SOA. Typical activities are related to managing service portfolios and lifecycles, ensuring service consistency, and monitoring service performance.

## 6 Inception

SOA is a long-term project and it is very important that it is seen as such. In other words, SOA is the long-term development of the overall IT architecture. Because SOA is a long time project it has to be managed like that [10]. To be successful we have to start deliberately and plan the project carefully:

First, we have to set the objectives. We have to identify the goals of SOA. It is important that we articulate the objectives very precisely. Just saying that we would like to improve the efficiency of business processes is not adequate. We have to identify which processes we would like to improve, why, when, and how much. Only when we will have a deep understanding of this, we will be able to move forward.

Then we have to identify the risks. There are many risks involved with the SOA project, starting with the organizational aspects, selection of processes, technology-related risks, etc.

Next, we have to take the necessary organizational steps and at the same time educate the SOA team members. Here it is very important, that we understand that SOA introduces many changes to all aspects of application development. Team members have to understand these changes. They also have to understand the new technologies and languages.

Next, we have to select the appropriate SOA platform. Major vendors today offer SOA platforms, which differ in several important aspects. Careful selection is therefore necessary, and we have to include specific aspects of the environment, existing systems, and existing knowledge into account to make a good decision.

A SOA project is usually started with a pilot project, which should be done with the help of external SOA experts. Within SOA pilot, several aspects can be addresses. The most important is probably that our team gets used to the round-trip development of business process modeling and their transition to executable BPEL processes. In other words, the SOA team has to feel comfortable with the composition approach to the development.

## 7 Value of SOA

SOA introduces important benefits: IT departments are under constant pressure of changes. SOA makes continuous changes easier and reduces the negative effects of changes [11].

Duplicated data through different databases and systems is quite common in existing systems. SOA fosters consolidation of data and introduces master data management solutions that are based on SOA concepts, services and loose coupling.

In existing applications, we are usually faced with duplicated functionalities. SOA fosters consolidation of such duplicated functionalities. Using services, we can expose composed functionalities.

When talking about business processes, companies often have variants of business processes that differ only in details. SOA enables support for such variants with common base process and their modification.

With the diversity of devices, it becomes more and more important to enable access to applications and data through different channels (PCs, palmtops, cell phones, voice, etc.). SOA enables access to processes through different channels [14].

IT departments often do not develop everything in-house. In the past (with existing approaches), it has been often quite difficult to separate the roles between the external partners (outsourcing partners) and in-house development. Too often, it has happened that the outsourcing partner has gained control over the application it has developed; and that the IT department (and the company as a whole) has become dependent on that partner. SOA enables easier separation of responsibilities, where services can be outsources, while their composition into business processes stays in-house. This way IT departments retain the control and the most valuable know-how a company has – the know-how about business processes.

Finally, SOA enables the development of service networks, which enable the development of virtual value chains, not only within the company, but also between the companies. This can open completely new possibilities in how IT can be used to optimize the business.

## 8 Changes in the Development Approach

SOA has also learned from the experiences of existing software development methods. Those have been based on requirements specification and have foreseen the analysis, design, implementation, testing, deployment and maybe some other phases. However the core assumption of majority of them has been, that the requirements have to be specified as precise as possible. Changing the requirements has always been seen as something that is not wishful [3, 12].

The fact today however is that changes are imminent. To be successful the software development approach has to consider this. SOA has considered change from the start. Therefore, it has introduced some important changes to the development approach. Instead of the classic approach, as shown on Figure 1, SOA introduces a modified approach, shown on Figure 2.

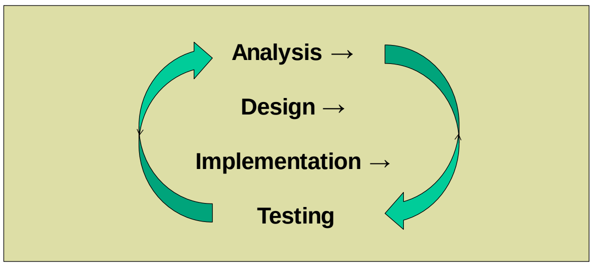

Figure 1. Classic approach to IS development

We can see that the phases are quite different. Instead of the analysis, the SOA approach foresees modeling, which refers to business process modeling. This way, the development is better aligned to the actual needs of the business. The traditional analysis has been based on requirements specification and a lot has been said and written how it is very difficult to specify the requirements. This is the first advantage of the SOA approach.

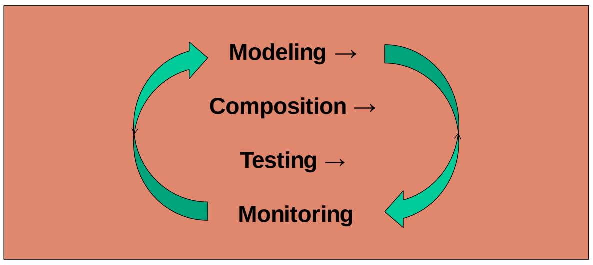

Figure 2. SOA approach to IS development

The second phase in the SOA approach is composition. Composition refers to the way business processes are developed. Instead of traditional implementation in a programming language such as Java or C#, SOA approach foresees that we will develop business processes so that we will reuse services and compose (orchestrate) them into processes. This approach works best when we already have a portfolio of services. We get such services from existing applications, where we expose business logic as services; or we buy services or outsource their development of external companies; or we develop services in-house.

In all three scenarios, we have to follow certain guidelines. The most important one is that we develop services, which are **reusable**. Reusable services are very important for SOA, because they represent "big" building blocks, which contain business logic. Developing applications (processes) with such existing building blocks is much faster compared to traditional approach in Java or C# (even if we have some libraries available). Therefore, the SOA approach to development is sometimes called *programming-in-the-large.*

The third phase of SOA development approach is testing. Testing SOA applications refers to testing the process and the related services. However, we reuse services. And reusable services have already been tested! Therefore, the effort required for testing is reduced as well [4].

Finally, we come to the monitoring phase. This phase refers to run-time monitoring of the process performance and includes: Monitoring of business activities – BAM, which provides valuable information about the performance and efficiency of business processes and can serve to identify future optimization points. Monitoring of QoA aspects of processes and services, such as response time, security, availability, etc. This is often related with the definition of SLA (Service Level Agreement) for processes and services.

These four phases are done iteratively and incremental. If organized accordingly, we can deliver working processes in 3 to 4 months periods [4].

The above-described changes to the development approach considerably reduce the overall complexity of the development. To some estimation, SOA reduces the complexity by approximately 50%. This is very important because the increased complexity of application systems has been an important source of the problems related to the too-long response times, needed for application modifications.

## 8 Conclusion

We have seen that SOA provides the technology platform for implementation of business processes – for the development of application, which provide end-to-end support for business processes. SOA is an architecture, which has introduced several important new concepts in the application development. One of the most important concepts is the composition of services into business processes. With this SOA has provided an architecture, which is flexible enough to accommodate business needs related to agility, adaptability, and to all other aspects related to the optimization of operations and improvement of business process efficiency.

We have to look at the SOA from three different perspectives: from business perspective, from technical perspective, and from organizational perspective. Only if we address SOA from all these three points of view, we can minimize the risks and maximize the benefits of SOA inception. The benefits are related to improved flexibility, better alignment of IT and business, faster and simplified application development with reduced complexity, and most importantly, end-to-end automation of business processes and reducing the semantic gap between business and IT. Risks are related to various organizational, technology, and business issues.

## References

[1] A. Poduval, Markus Zirn, Matjaz B. Juric, Todd Biske, Jerry Thomas, Do more with SOA Integration, PACKT Publishing, 2011

[2] Matjaz B. Juric, Swami Chandrasekaran, WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, PACKT Publishing, 2010

[3] H. Gaur, M. Zirn, M. B. Juric, Oracle Fusion Middleware Patterns, PACKT Publishing, 2010

[4] Matjaz B. Juric, Frank Jennings, Poornachandra Sarang, Ramesh Loganathan, SOA Approach to Integration, PACKT Publishing, 2007

[5] Matjaz B. Juric, Kapil Pant, Business Process Driven SOA using BPMN and BPEL, PACKT Publishing, 2008

[6] Matjaz B. Juric, A. Poduval, D. Todd, H. Gaur, J. Bolie, J. Thomas, K. Geminiuc, L. Pravin, M. Zirn, M. Cardella, P. Ramachandran, S. Carey, S. Blanvalet, T. H. Nguyen, Y. Coene, BPEL Cookbook: Best Practices for SOA-based integration and composite applications development, PACKT Publishing, 2006

[7] Matjaz B. Juric, Benny Mathew, Poornachandra Sarang, Business Process Execution Language for Web Services 2nd Edition, PACKT Publishing, 2006

[8] Matjaz B. Juric, B. Brumen, I. Rozman, WSDL and UDDI Extensions for Version Support in Web Services, Journal of Systems and Software, doi:10.1016/j.jss.2009.03.001, 2009.

[9] Matjaz B. Juric, I. Rozman, WS-BPEL Extensions for Versioning, Information and Software Technology, Volume 51, Issue 8, (2009) 1261-1274.

[10] Matjaz B. Juric. WSDL and BPEL extensions for event driven architecture. Inf. softw. technol.. [Print ed.], 2010, vol. 52, iss. 10, 1023-1043, doi: 10.1016/j.infsof.2010.04.005.

[11] FRECE, Aleš, JURIČ, Matjaž B. Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models. J. vis. lang. comput., Aug. 2012, vol. 23, no. 4, 223-247.

[12] KRIŽEVNIK, Marcel, JURIČ, Matjaž B. Data-bound variables for WS-BPEL executable processes. Comput. syst. struct., Dec. 2012, vol. 38, no. 4, 279-299.

[13] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, 2005.

[14] T. Erl, SOA Principles of Service Design, Prentice Hall, 2007.