Interface-based enterprise and software architecture mapping

Aziz Ahmad Rais
Department of Information Technologies
University of Economics, Prague
Prague, Czech Republic
aziz.rais@vse.cz
aziz.ahmad.rais@gmail.com

DOI: 10.20470/jsi.v7i2.253

Abstract: Information technology (IT) becomes more and more complex because of various technologies, methodologies, techniques and practices. Even though the goal of all technologies, methodologies, practices and techniques is to facilitate construction, to simplify, and to increase the reusability of information systems, in practice integrating all these becomes a challenge. This challenge can be met by creating more abstract levels in the information systems in question. Higher-level abstraction simplifies different views of complex problems, but at the same time it generates a knock-on issue regarding how actually to implement such an abstract-level view, and/or how to map it back to the lower levels of abstraction.

The goal of this article is to simplify the implementation of enterprise architecture and map it to software architecture using an interface-based analysis technique.

In order to achieve this goal, service-oriented architecture (SOA), which is composed of multiple concepts, will be used. The concepts are flexible, so they can be applied in enterprise architecture as well as in software architecture.

Key words: TOGAF, enterprise architecture, service-oriented architecture (SOA), software architecture

1. Introduction

In order to simplify and implement enterprise architecture using service-oriented architecture (SOA), we first need to describe the fundamentals of both architectures styles. The architecture style according to the ISO/IEC/IEEE 24765 (2010) is “characterization of a family of systems that are related by sharing structural and semantic properties”. This means finding commonalities between enterprise architecture and SOA structure and semantic properties to map, or rather to integrate them.

Service-oriented architecture is a paradigm applicable in different types of architecture, such as, for example, software architecture, component design, enterprise architecture, and so on. In other words, the Open Group’s SOA Working Group (SOA, 2009) explains SOA as follows:

Service-Oriented Architecture (SOA) is an architectural style that supports service orientation. Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services.

A service:

Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports)

1. Is self-contained
2. May be composed of other services
3. Is a “black box” to consumers of the service

Other standards, like the advancing open standards for the information society (OASIS, 2006), describe SOA as a paradigm with the following concepts: visibility, interaction, real-world effect, execution context, contract and policy, service description, and service.

According to both standards, the concept of “service” is key for SOA. However, OASIS defines the concept of service as:
1. The capability to perform work for another
2. The specification of the work offered for another
3. The offer to perform work for another

Sometimes, SOA is described as an integration concept, (Schmutz, 2010), for which, for example, an enterprise service bus can be used (ESB). The service is orchestrated as independent, distributed service calls, as in the following diagram:

![Service-oriented architecture (Schmutz, 2010)](image)

The ESB has the capability of orchestrating requests, and supports the collaboration between different systems in a decoupled way. The interpretation of SOA as an integration concept is also implied by the Open Group’s definition of SOA provided above, especially by its third sub-definition.

Following from the above definitions and concepts, any implementation without the use of reference architecture would hinder SOA construction. SOA reference architecture offers another view of SOA. According to The Open Group Standard Group’s clarification of SOA reference architecture (SOA Ref. Arch, 2011), SOA is composed of layers that are divided into two types: the logical (the Service Component Layer, the Services Layer, the Business Process Layer, and the Consumer Layer) and the physical (Operational Systems Layer).

![Logical Solution View of the SOA RA (The Open Group Standard, 2011)](image)
The layers can be identified as follows: Governance Layer (availability, registries and repositories), Integration Layer, Quality of Service Layer (administration, monitoring and management) and the Information Layer (defining events) support five vertical layers. In practice, a supportive layer means an optional layer. Thus, if architecture does not implement them but is nevertheless service-driven, it can still be labeled SOA.

The architecture development method (ADM) explained by the Open Group Architecture Framework, describes enterprise architecture, like SOA, as the composition of multiple layers (TOGAF, 2011). This is illustrated in the diagram below:

2. Mapping enterprise architecture to SOA

The term “service” is generic and can be applied at different system levels, such as, for example, those of business system, information system and software system. Thus, when we make each of these three systems service oriented, the results will be business service, information system service, and software service. According to the open group architecture framework in (TOGAF, 2011), the information system service is: “The automated elements of a business service. An information system service may deliver or support part or all of one or more business services”.

By identifying the layers of enterprise architecture and SOA, we can find the proper layer of business service. In enterprise architecture, business service are delivered by the business domain, and thus by business architecture. According to the SOA paradigm, business service is involved in the business
process layer and service layer. The consumer layer can be another information system service, or human. In both cases, the goal of SOA is to design a service from a consumer perspective. Thus, the consumer layer is also part of business service.

The service components in the service component layer of SOA reference architecture are one or more software components that can share or run in separate runtime environments. In TOGAF, application architecture is understood as “A description of the structure and interaction of the applications as groups of capabilities that provide key business functions and manage the data assets”. Every application in application architecture can be considered application software, because the latter is, according to the SO/IEC/IEEE 24765 (2010) definition, software or a program that helps the user perform a task or a business function, and manage the data asset. In conclusion, application software can itself be a software component. To conclude, the TOGAF information system service which is composed of application and data architecture is actually equivalent to SOA service components layer.

The operation layer in SOA is the runtime environment for software components, and is equivalent to technology architecture. Technology architecture, according to TOGAF, is a service platform provided by technologies such as the operation system, network, hardware, and devices that all together create runtime environments for application software (TOGAF, 2011).

Last, to use SOA within enterprise architecture, or to build service-oriented enterprise architecture, we need to carry out the mapping below.

The conclusion above can also be inferred from the Open Group’s explanation of SOA, which appears in The Open Group's SOA Working Group (SOA, 2009) as follows: “An enterprise architect looks at the overall construction of the enterprise. SOA is a particular construction technique that can be used to build enterprise IT.”

![Mapping enterprise architecture to SOA](image-url)

**Fig. 4: Mapping enterprise architecture to SOA (the author)**

### 3. Mapping software architecture to SOA

Software architecture can also be composed of multiple layers, implies from (VOŘIŠEK, 2008). Each layer can be created based on a particular concern. These concerns can be, for example, how users will employ the software functionalities and how to access them (the presentation layer); what types of functionality the software will provide (the software service layer); how to reuse the functionalities of other software or software services (the integration layer or, for example, with the help of an ESB); how software will gather and store or keep the business data; or, finally, what technologies to use for software construction and for runtime software (the technology layer or operation layer). The integration layer helps the software layer compose software services from other software services. Based on the concerns outlined here, an example of application software can be composed from the following layers:
Multi-layer software architecture makes mapping software architecture to SOA easy, because the layers are similar in both types.

The functionality provided by the software-service layer in software architecture has lower granularity than the information system service in enterprise architecture. The presentation layer in software architecture is the same as the consumer layer in SOA; it presents the software service to the user or consumer. The presentation layer also controls and executes the business process. The business process consumes the software service implemented by the software service layer.

Service-component implementation by software has to be done in multiple sub-layers, because software has to be composed from objects, classes, and methods. According to Service oriented architecture Modeling Language in (SoaML, 2012), a software service model is composed of various objects: interface, concrete instance, request, channel, and service data. Additionally, a software service needs other objects: participants, contracts and capabilities. The first five SoaML objects are also important for model-driven architecture and development (OMG MDA, 2014), because the model can be transformed into software source code skeletons. So the service layer in 0 represents the software service interface, and the concrete instance of the service is composed by invoking the data access and integration layers. The data layer in 0 represents the service data gathered from the incoming request.

The following diagram maps multi-layered software architecture to SOA reference architecture as illustrated in 0:
4. Mapping enterprise architecture to software architecture

In the two types of mapping outlined above, we identified that the linchpin of integration between all architecture styles is the service layer. Thus, by mapping the service layers properly, we will be able to produce accurate enterprise architecture. This means mapping technologies and technology architecture will not cause any complexity when business services are implemented in an enterprise.

Software service components are identified by SoaML and, in order to compare and map between both types of service, we need to understand business service.

The TOGAF Core Content Metamodel in (TOGAF, 2011) shows that the main business service components are Process, Function, Role, Actor, and Organization Unit; see the diagram below:

Fig. 7: Relationship between business service components and entities in the TOGAF Core Content Metamodel (The Open Group Standard, 2011)
The most important implication of the TOGAF Core Content Metamodel is that business service is realized through the application component (in the Core Metamodel, the application component is the application software component). In the section entitled “Mapping enterprise architecture to SOA”, it was shown that the application component can be one or more software components. Furthermore, each software component provides a software service to support the business service, as described in the section entitled “Mapping software architecture to SOA”.

To summarize, it has been clarified that, from a component perspective, business service and software service are the same, and the former uses the latter for its realization. In addition, the TOGAF Core Content Metadmodel implies on the one hand that business service involves the whole organization in its process, and the consumer of the business service’s process and functions is human. On the other hand, software service involves only part of the organization, and its consumer can also be other software.

The goal of this article is to simplify the implementation and understanding of enterprise architecture, and we can see that the key to mapping between business service and software service is mapping the interfaces of both services. Business service or business architecture can be described using the enterprise architecture modeling language ArchiMate, which is an extension of UML, or by using UML itself (ArchiMate, 2013). Due to the fact that ArchiMate provides different meta-objects, which can cause additional complexity, UML used with the proper stereotypes is sufficient for mapping and understanding enterprise architecture and software architecture.

According to BABOK in (IIBA, 2009), an information system can be described with the use of interface analysis techniques. TOGAF defines an information system as a combination of application software and data (TOGAF, 2011). Further, ISO/IEC/IEEE 24765 (2010) describes it as a processing system associated with organization assets, such as human resources, technical resources, and financial resources. Combining these definitions, therefore, we can understand application software and data as technical resources consumed by organization employees, handling finances, and likewise requiring financial investment themselves. Moreover, according to the TOGAF Core Content Metadmodel, a service is bounded by the functions that can be grouped in an interface. Consequently, the following enterprise architecture model is constructed with the help of an interface, and describes only the information-system part of the enterprise.

The example of an information system in 0 below models an enterprise providing a service of “flower sales”, which is composed of several sub-services: flower ordering, flower delivery, and accounting.

![Diagram](image-url)

**Fig. 8:** Information-system model and its mapping to the application software component (the author)
From the TOGAF Core Content Metadmodel, we already know that a service is realized by a process, and this process will, therefore, be mapped to the interfaces of an application software component. The following model describes how the ordering service processes are mapped to the interfaces of an application software component.

Fig. 9: Mapping a business process supported by an application software component

(5) Conclusion

In the introduction section, we saw a high-level description of the SOA concepts used to describe enterprise architecture in a service-oriented way. Mapping between both types of service is simplified because software can also be service oriented, thereby removing technology architecture from enterprise and software architecture. The TOGAF concept of service implies that the latter is implemented by a business process or a process in general, and is bounded by functions. As a result, we can group functions into interfaces and map them to processes. Thus, the example of the business service and information system of enterprise architecture of a flower sales business, with its three types of sub-service, was described with the use of interfaces, where one service interface was mapped for demonstration to the application software services through mapping the sub-process of the “flower order” service to the interfaces of an application software component.

References


JEL: L86, M15