Comparing Methodologies for Service-Orientation using the Generic System Development Process

Linda Terlouw

Delft University of Technology, Delft, The Netherlands
Ordina, Nieuwegein, The Netherlands

Abstract. Enterprises use Service-Oriented Architecture (SOA) and Service - Oriented Design (SoD) as a means to achieve better business-IT alignment and a more flexible IT environment. Definitions of both notions are often not clear or even contradict. This paper instantiates the Generic System Development Process (GSDP) for service-orientation and provides a common terminology for comparing the scope of different methodologies. To demonstrate its use we compare several existing methodologies. Methodologies that focus on (nearly) the whole service-oriented development process are Papazoglou's and van den Heuvel's service - oriented design methodology, SOMA and SOAF. More specialized methodologies are the goal-driven approach, BCI3D, and Business Elements Analysis.

1 Introduction

Most large enterprises have a complex application landscape. Functional overlap between applications exists and applications exchange information using different communication mechanisms. Often the relationship between the applications and their supporting business processes is not explicitly described and the interdependence between applications acts as bottleneck in redesigning business processes to exploit new business opportunities.

The concept of *Service-Oriented Architecture* (SOA) aims at improving this situation by exposing application functionality in *services*. These services are accessible in a uniform way and do not show the underlying technology of the application(s). *Service-Oriented Design* (SoD) is concerned with the process of creating these services in a systematic way. Although many methodologies for SoD are available, only few address the complete *design process* and describe the design steps in the necessary depth. Comparing these methodologies is quite a challenge because they lack a common view of SOA, SoD and the steps involved in design. The main contributions of this paper are a clear terminology of SOA and SoD based on the *Generic System Development Process* (GSDP) [1]. Also, we make a brief comparison of several existing methodologies for service-orientation based on the phases of this process.

We start this article with explaining our view of architecture in section 2. Section 3 describes the GSDP and its instantiation for service-orientation. In section 4 we discuss several methodologies. Finally, we end with our conclusions in section 5.

Terlouw L. (2008).
Comparing Methodologies for Service-Orientation using the Generic System Development Process.
In *Proceedings of the 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 101-108
DOI: 10.5220/0001738901010108

2 Architecture

A common misconception in industry as well as in academia is that web services are the same as SOA. One cannot speak of SOA when developers have created some web services using modern integration technology. In fact, SOA does not even require web services. Many definitions of SOA [2–5] mention the notion of architecture, architectural style or paradigm, but lack a clear definition. When we do see a definition of architecture, it is usually one of the following three: the *descriptive* definition, the *prescriptive* definition, and the combination of both.

The descriptive approach defines architecture as high-level models using names like ‘high-level components’ or ‘blue prints’. Zachman uses this approach in his definition [6]: “Architecture is that set of design artifacts, or descriptive representations, that are relevant for describing an object, such that it can be produced to requirements as well as maintained over the period of its useful life”. In the xAF (Extensible Architecture Framework) [7] architecture is defined as: “the normative restriction of design freedom because of the notion that design freedom of designers is undesirable large. Practically, architecture is seen as a consistent and coherent set of design principles that embody general requirements”. Thus architecture is seen as prescriptive. Also, definitions exist that combine the descriptive and prescriptive approach. The IEEE 1471 standard [8], for instance, defines architecture as: “the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principle guiding its design and evolution”.

In this remainder of this article we use the prescriptive definition of architecture. So, we see SOA as a consistent and coherent set of design principles that need to be taken into account in the development process of services.

3 The Service-Oriented Development Process

The GSDP, exhibited in figure 1, applies to any *system*. Two different system notions exist: the teleological and the ontological system notion [9]. The teleological system notion is about the function and the (external) behavior of a system and the corresponding type of model is a *black-box model*. The ontological system notion is about the construction and operation of a system and the corresponding type of model is a *white-box model*. Both the black-box model and the white-box model of the GSDP are relevant to service-orientation; the black-box model for using services and the white-box model for building (or changing) services.

The *object system* is the system that needs to be developed for supporting the *using system*. The development of the object system consists of three phases: *design*, *engineering*, and *implementation*.

3.1 Service-Oriented Design

The design phase consists of two steps: function design and construction design. Function design has a black-box model as result. Construction design has a white-box model

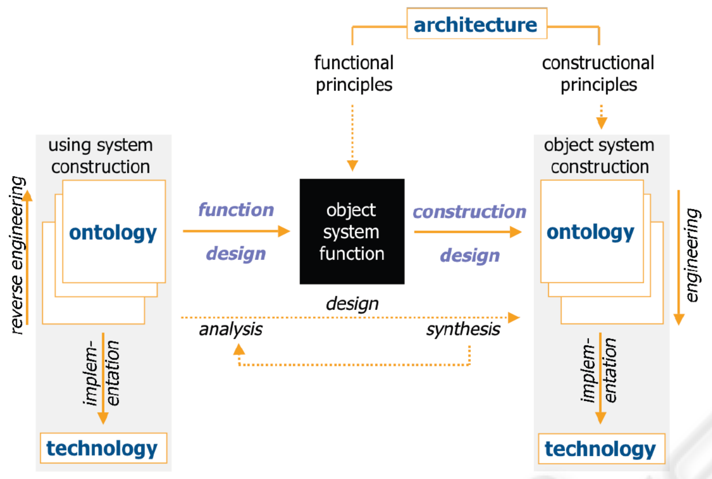

Fig. 1. The generic system development process [1].

as result which is called an *ontology*. The term ontology originates from the field of philosophy having the meaning ‘study of existence’. In our context we define ontology as follows [9]: the ontology of a system is the understanding of the system’s operation that is fully independent of the way in which it is or might be implemented. Applying this to service-orientation we see two phases in SoD: *Service-Oriented Function Design* (SoFD) and *Service-Oriented Construction Design* (SoCD). SoFD deals with determining and specifying the function of a service. *Service identification* is in our view the first step in SoFD. It deals with identifying candidate services in a systematic way. The central question is “What services are required in the scope of the SOA?”. The process of *service specification* (also the QoS requirements) is part of SoFD as well, since it specified the external behavior of a service without caring about it internals. Once the specification is available one can design the construction of a service.

SoCD results in the highest possible white-box model of the service. A service needs to be constructed in such a way that it conforms to the constructional principles of the architecture. A common classification of services is that of *atomic* and *composite* services. A composite service depends on the execution of other, lower-level services. An atomic service does not. For the service consumer there is no notable difference in which way the service provider constructs its service. Composite services can be constructed, for instance, by *orchestration* (BPEL [10], BPML [11]). An orchestration determines in which order and under what conditions which lower-level services are called. Orchestration is especially suitable for creating very high-level services that support multiple steps in a (partly) automated business process. Another way to create composite services is assembly (SCA [12]). The highest-level construction model of an atomic service could, for example, be a (high-level) class diagram if the service is implemented in an object-oriented language.

Table 1. Terminology derived from applying the GSDP to service-orientation.

Term	Meaning	End result
SOA	a coherent set of design principles that need to be taken into account in development process of services	all services in scope conforming to the same principles
Service-Oriented Function Design	deals with the design of the function of the service	the service specification of an identified service
Service-Oriented Construction Design	deals with the design of the highest-level white-box model	the high-level design of the internals of the service
Service Engineering	deals with the decomposition of the highest-level white-box model to the lowest-level white-box model	the full design of the internals of the service
Service Implementation	deals with the mappping of the lowest-level white-box model to technology, i.e. the deployment of services	the deployed service

3.2 Service Engineering

Engineering is the process of deriving lower-level white-box models from the high-level white-box model. *Service Engineering* (SE) deals with decomposing the highest-level white-box model to lower-level white-box models. The lowest-level construction model is also called the implementation model. This model is the source code of the service itself in case of an atomic service or the source code of the referenced services in case of a composite service. In practice, most services are not constructed completely in a top-down way because not only new systems, but also existing systems, are used as building blocks for services. The design process is therefore iterative: the final result of every design process is (or should be) a balanced compromise between reasonable functional requirements and feasible constructional specifications [13].

3.3 Service Implementation

The implementation of the object system is the assignment of technology to the lowest-level white box model (the implementation model). Dietz and Hoogervorst use technology in the broadest possible meaning, e.g. human beings, software systems, electronic machines. *Service Implementation* (SI) deals with mapping the implementation model to computer systems. This is the phase in which the services are deployed. Most enterprises have at least two test environments to deploy to after the service has been developed: a test environment for verification (checking whether the services matches the design) and one for validation (checking whether the service is useful to potential service consumers). Both tests can lead to repeated execution of previous steps in the design process and redeployment to the test environment. Finally, the service is deployed to the production environment.

3.4 Service-Oriented Architecture

The architecture prescribes general *functional* and *constructional* requirements (the principles). As stated in section 2 we define SOA as a consistent and coherent set of design principles that need to be taken into account in the development process of services. Marks and Bell [14] provide nine criteria that services must meet: coarse-grained, well-defined service contracts, discoverable, business aligned, reusable, durable, loosely coupled, composable, and interoperable. More principles for service design exist, but it is not our goal to present a complete list. Instead, we want to show our way of thinking. The functional requirements deal with the external function and behavior of a service, e.g. “A service is coarse-grained” and “A service is business aligned”. The constructional requirements deal with the internal construction and operation of the service, e.g. “A service is composable” and “A service is loosely coupled”.

Are all of these principles consistent? In our view they are not. For instance, if a service is completely business aligned to the business process of one service consumer, it is probably not reusable and vice versa. Enterprises, however, need a consistent set of principles in order to make design decisions in a systematic way. Therefore, an enterprise has to formulate explicitly whether and under what conditions one principle has priority over another one.

Table 1 summarizes the terminology explained in this section.

4 Comparing Methodologies

Table 2 depicts the scopes of several methodologies for service-orientation. We have divided them into general methodologies and specialized methodologies.

4.1 General Methodologies

Methodologies that cover the largest part of the service development process are the methodology of Papazoglou and van den Heuvel [15], the IBM methodology SOMA [16], and SOAF [17]. Papazoglou and van den Heuvel distinguish between the phases, that they call, planning, analysis and design, construction and testing, provisioning, deployment, execution and monitoring. This methodology is in our view the most comprehensive approach at this moment in time. Not only does it deal with SoFD, SoCD, and SE, but also with SI (in the deployment phase). It provides guidelines on how to perform these steps.

SOMA distinguishes between the three major activities, that the authors call, identification, specification, and realization. The methodology does not make a clear distinction between the function and construction of the service. It does not deal with SI. Since only a high-level description of SOMA is publicly available, we cannot say anything about the depth in which the design activities are described.

SOAF addresses the phases, that the authors call, information elicitation, service identification, service definition, and service realization. It clearly described what steps to take and what the relationship between the different steps are. However, it focuses mainly on what steps to take and not on how to take these steps. It is described at a higher level of detail than the methodology from Papazoglou en van den Heuvel.

Table 2. Classification of SOA methodologies.

Methodology	**SoFD**	**SoCD**	**SE**	**SI**
BCI3D	x			
Business Element Approach	x			
Goal Driven Approach	x			
SOAF	x	x	x	x
SoD and Development Methodology	x	x	x	x
SOMA	x	x	x	
SMART		x	x	

4.2 Specialized Methodologies

Also, more specialized methodologies exist. In the area of service identification (part of SoFD) there are the goal-driven approach [18], BCI3D [19] and Business Elements Analysis [20]. The goal-driven approach finds its basis in the world of *component based development* and derives services from business goals. The services are depicted in a goal-service graph and service are allocated to enterprise components. These enterprise components are identified by clustering highly interdependent (coupled) use cases.

BCI3D also has a background in component based development. It identifies business components on the basis of a formal organizational model, called the enterprise ontology [1]. An algorithm is used to cluster business process steps from the business process model and object class from the information model forming a business components. The calls between these business components are regarded as the highest level services an organizations requires.

Like the last two methodologies Business Elements Analysis has its roots in component based development. It defines Resource Business Elements (RBE's), which group the (information) resources and Service Business Elements (SBE's), which consist of the highest-level immediate steps. An RBE consists of a focus resource, which is independent (e.g. Customer) and its auxiliary resources (e.g. Address), which always belong to a certain focus resource. A Delivery Business Element (DBE) is a grouping of Service and Resource Business Elements that together deliver a business solution to a business problem, and which provides services to requesters.

SMART [21] is a methodology described in detail on how to construct identified services from legacy systems (SoCD and SE). SMART takes into account that in practice it is often not easy to construct services from legacy systems. Since almost no organization has the luxury to build up its entire IT-environment from scratch, it is important to realize the risk involved in migrating to SOA. According to the SMART methodology an organization needs to thoroughly assess the capabilities of its legacy systems and carefully analyze the risk of migrating. A migration plan describes the necessary steps to take for the migration.

5 Conclusions

At this moment in time, most SoD methodologies do not make a clear distinction between the notions of SOA and SoD. The main contribution of this paper is a clear terminology of these notions based on the Generic System Development Process. We introduced the following phases in the development process for services: Service-Oriented Function Design, Service-Oriented Construction Design, Service Engineering, and Service Implementation. Using this high-level classification of phases one can determine the scope of different SoD methodologies. We have provided a brief comparison of several existing methologies. Some methodologies cover (almost) the whole development process. These methodologies tend to focus on what steps to take and not on the approaches used to execute the steps. We do need to note that since IBM's SOMA methodology is not publically available, we based our analysis on available articles which may not cover its full contents. Some of the specialized methodologies provide an in-depth contribution to specific steps of the development process. In our view, however, there is still a large need for more in-depth research on specific parts of the development process. In our further research we will focus on the specification of services, which is part of the Service-Oriented Function Design phase.

References

1. Dietz, J., Hoogervorst, J.: Enterprise ontology and enterprise architecture, how to let them evolve into effective complementary notions. GEAO Journal of Enterprise Architecture **2** (2007)
2. OASIS: Reference model for service oriented architecture, committee draft 1.0. (2006) `http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf`.
3. OMG: Service oriented architecture sig (2006) `http://soa.omg.org/`.
4. The Open Group: Service oriented architecture (2006) `http://www.opengroup.org/projects/soa/`.
5. W3C: Web service architecture (2006) `http://www.w3.org/TR/ws-arch/`.
6. The Zachman Institute for Framework Advancement: Enterprise architecture: A framework (2007) `http://www.zifa.com/framework.pdf`.
7. Dietz, J.: The extensible architecture framework (2004) `http://www.lac2004.nl/docs/fvbg2hdsb83/Track8/J.%20Dietz.pdf`.
8. Maier, M.W., Emery, D., Hilliard, R.: Software architecture: Introducing ieee standard 1471. Computer **34** (2001) 107–109
9. Dietz, J.: Enterprise Ontology, Theory and Methodology. Springer, Berlin Heidelberg, Germany (2006)
10. OASIS: Web services business process execution language version 2.0 (2007) `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.
11. Thiagarajan, R.K., Srivastava, A.K., Pujari, A.K., Bulusu, V.K.: Bpml: A process modeling language for dynamic business models. In: WECWIS '02: Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02), Washington, DC, USA, IEEE Computer Society (2002) 239
12. Open SOA: Service component architecture (2007) `http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications`.

13. Albani, A., Dietz, J.: Enterprise ontology based design of inter-enterprise information systems. Technical report, Delft University of Technology (2007)
14. Marks, E., Bell, M.: Service-Oriented Architecture, A planning and implementation guide for business and technology. John Wiley & Sons, Inc., Hoboken, New Jersey (2006)
15. Papazoglou, M., van den Heuvel, W.J.: Service-oriented design and development methodology. International Journal of Web Engineering and Technology 2006 **2** (2006) 412–442
16. Arsanjani, A., Allam, A.: Service-oriented modeling and architecture for realization of an soa. In: SCC '06: Proceedings of the IEEE International Conference on Services Computing, Washington, DC, USA, IEEE Computer Society (2006) 521
17. Erradi, A., Anand, S., Kulkarni, N.N.: Soaf: An architectural framework for service definition and realization. In: IEEE SCC, IEEE Computer Society (2006) 151–158
18. Levi, K., Arsanjani, A.: A goal-driven approach to enterprise component identification and specification. Communications of the ACM **45** (2002) 45–52
19. Albani, A., Dietz, J.: The benefit of enterprise ontology in identifying business components. In: WCC '06: Proceedings of the IFIP World Computer Congress, Santiago de Chile, Chile (2006)
20. McGovern, J., Sims, O., Jain, A., Little, M.: Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
21. Lewis, G., Morris, E., Smith, D.: Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. In: CSMR '06: Proceedings of the Conference on Software Maintenance and Reengineering, Washington, DC, USA, IEEE Computer Society (2006) 15–23