The SOA's Layers

Christian Emig, Kim Langer, Karsten Krutz, Stefan Link, Christof Momm, Sebastian Abeck

Cooperation & Management, Universität Karlsruhe (TH)
{ emig | langer | ... }@cm-tm.uka.de

Abstract

In this paper we introduce a reference model for the layering of service-oriented architecture (SOA). Opinions on how this architecture appears are often disjunctive or contradictory within the SOA community - a standardized layering has not as yet been established. Therefore, it is our objective to provide a definition and better understanding of the layers encountered within an SOA as well as the relationships between them, thus aiming at the provisioning of common understanding and nomenclature for SOA.

Basically SOA aims at the provisioning of abstract software functionality through services that can be flexibly composed to implement business processes. Through the deployment of reusable services a process-oriented alignment of business and IT is achieved, allowing fast adaptations on changes in business processes [EW+06]. Furthermore, SOA enables the integration of existing applications by exposing their functionality as services improving the value of existing software assets and avoiding redundancy in IT infrastructure. To yield these benefits and to support the service-oriented paradigm, SOA adds further layers to the conventional architecture responsible for application integration and service orchestration hence making the underlying systems transparent for business analysts.

The bottom layer of the reference model comprises the existing systems, called operational systems [Ar04] or legacy systems. It holds detached systems that are traditionally developed and therefore usually per se not service-oriented. From the SOA perspective, it does not matter if these legacy systems are internally monolithic or of multi-tier architectures. It is the task of an SOA to leverage these systems by exposing their functionality as reusable services that build the foundation for the alignment of IT and business processes. Up to the point of time when SOA principles are directly applied in application development, most of the scenarios will have the migration character described before.

At the **c**ore services border the wrapping of existing functionality to services is achieved with a common interface description and communication protocol that can be used by potential consumers. The act of exposing this functionality as services to gain reusability is done by software developers and involves changes in the applications' source code or proxy-style wrapping. Such services are described by providing service descriptions held in a service registry that gives information about the syntax (e.g. through Web Service Description Language WSDL [KL04]) and semantics (e.g. using Ontology Web Language - Services OWL-S) [MP+04]) of the service interfaces.

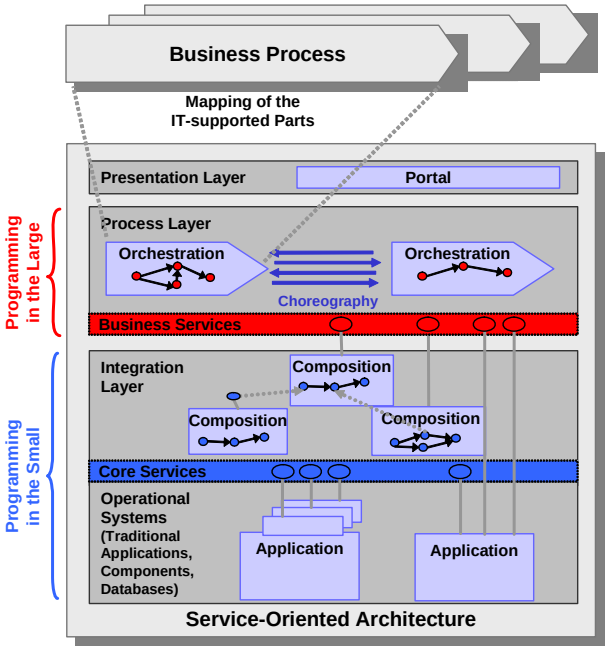

Figure 1: The SOA's Layers

Through service composition at the integration layer, functionality can be further aggregated. The term composition in this context denotes the combination of services yielding a more complex and coarser-grained service [ZT05]. At the integration layer, core services are

composed that quite often have strong dependencies on the underlying software systems. This is the field of classical Enterprise Application Integration (EAI), but now standardized service interface description (e.g. WSDL) and communication protocols (e.g. Simple Object Access Protocol SOAP) are applied. System integrators may use the Business Process Execution Language (BPEL) at this layer in web service based SOA.

Reaching the border of business services, the relation to the software systems has to be reduced to a minimum. The hiding of the technical aspects is the chance to enable business analysts to map their business processes to the process layer combining (i.e. orchestrating) business services.

The process layer sets up on the integration layer. It implements, as the name implies, the IT-supported parts of business processes that are mapped to business services. The act of plugging together business services in order to accomplish business logic and processes is called orchestration. More precisely, the term orchestration refers to the realization of a business related workflow owned by a single entity [Er05] through the combination of business relevant services. Processes realized by orchestration expose a service interface themselves [ZT05]. Orchestration is business driven and therefore directly related to the authoritative business processes. Processes realized in this kind of fashion can in turn influence and exchange information among each other. The coordination of business processes is called choreography and describes the communication protocols between business services. As with orchestration, it is associated with the process layer but unlike orchestration which yields processes owned by a single entity, choreography considers the interaction of these, often federated processes. In doing so, choreography enables collaboration between orchestrations [Er05] by the definition of a set of rules for this collaboration.

In literature, it is usually not differentiated between service composition and process orchestration and both are put together as the so-called programming-in-the-large [Le03]. In a contrary to this point of view, we see two aspects that should lead to a separation into the integration layer and the process layer: First of all, the job is done by different people/roles. System integrators need standardized interfaces that reduce artificial dependencies for integration. Business analysts are concerned with the process layer where the orchestration of business related services takes place. Secondly, the types of services are different in respect to system-related dependencies and business relation at both layers. This is the reason why we suggest separating these two layers and adding the integration aspects to the programming-in-the-small and the orchestration aspects to the programming-in-the-large. Programming-in-the-large, as done in the process layer by business analysts, implies that the programming is done on an abstract and technology independent level. As orchestration and choreography are directly related to business processes it is to be carried out by business experts such as business analysts than system integrators.

The presentation layer on top of the process layer integrates human users [Ar04]. It provides users with a means to interact with the present processes. This interaction can range from operational actions to business related operations like starting processes or providing business relevant user input. BPEL4People and Web Services for Remote Portlets (WSRP) are promising upcoming standards to be applied in this layer.

Various sources divide services into two categories [NL04, Er05], coarse-grained and fine-grained services. Since this type of classification is relative and no common metrics have been defined [Le03], we pursue an approach for service classification that has an absolute measure. Our reference layering model therefore groups services into core web services and business web services. The layer of core web services defines the border at which SOA is reached by existing applications. Services from different underlying applications can now be composed at the integration layer by integration specialists. In contrast to core web services, business services directly participate in a business process and have immediate business relevance whereas core web services tend to be invoked by coarser grained services instead of being directly incorporated into process flows though they syntactically have the same interfaces and communication.

References

[Ar04] Ali Arsanjani: Service-Oriented Modeling and Architecture, IBM developer works, 2004.

[Er05] Thomas Erl: Service-Oriented Architecture: Concepts, Technology and Design, Prentice Hall PTR, ISBN 0-13-185858-0 August 04, 2005.

[EW+06] Christian Emig, Jochen Weisser, Sebastian Abeck: Development of SOA-Based Software Systems – an Evolutionary Programming Approach, International Conference on Internet and Web Applications and Services ICIW'06, February 2006.

[KL04] Donald Kossmann, Frank Leymann: Web Services, Informatik Spektrum, Band 27, 117-128, Springer Verlag, 2004.

[Le03] Frank Leymann: Web Services - Distributed Applications without Limits, Business, Technology and Web, Leipzig, 2003.

[MP+04] D. Martin, M. Paolucci, S. McIlraith et. al.: Bringing Semantics to Web Services: The OWL-S Approach, 2004.

[NL04] Eric Newcomer, Greg Lomow: Understanding SOA with Web Services, Addison Wesley Professional, ISBN 0-321-18086-0, December 14, 2004

[ZT05] zapThink, Ronald Schmelzer, Jason Bloomberg: zapThink's Service-Oriented Architecture Roadmap, http://www.zapthink.com/report.html?id=ZTS-GI103, 2005