Integration Strategies and Patterns for SOA and Standard Platforms
Helge Buckow, Hans-Jürgen Groß, Gunther Piller, Karl Prött, Johannes Willkomm, Alfred Zimmermann
SOA Innovation Lab e.V.
Workstream SOA and Standard Platforms
c/o Deutsche Post AG
Charles-de-Gaulle-Straße 20
53113 Bonn
info@soa-lab.de, helge_buckow@mckinsey.com, hans-juergen.gross@daimler.com, gunther.piller@fh-mainz.de, karl.prött@capgemini-sdm.com, johannes.willkomm@capgemini-sdm.com, alfred.zimmermann@reutlingen-university.de

Abstract: The SOA Innovation Lab presents a holistic approach for the development of a service-oriented enterprise architecture with custom and standard software packages. Starting point is the construction and analysis of company domain maps with respect to characteristics of SOA and standard software. After assessing the SOA ability of standard software packages within an architecture maturity framework, we show how target architectures can be developed with the help of use case analyses, capability maps and integration patterns. Besides methods and related artifacts, we present current adoption issues for standard software packages in service-oriented contexts.

1 Introduction and Method Overview
The growing complexity of IT landscapes is a challenge for many companies. A large number of standard software packages - mostly extended and modified, individual software solutions, legacy applications, and different infrastructure components - lead to high cost and limited IT agility. Many companies start enterprise architecture management (EAM) initiatives to address this problem. In areas where flexibility or agility is important, SOA is the current approach to organize and utilize distributed capabilities. Here, the use of standard software is often a challenge, in particular when dealing with services on a fine granular level.

The SOA Innovation Lab has developed an approach for the design of a service-oriented enterprise architecture with custom and standard software packages [Bu+10]: In order to identify areas for the use of standard software packages in a service-oriented environment, it is necessary to establish basic EAM capabilities. Corresponding activities and artifacts are sketched in Section 2.

For domains, which benefit from the advantages of SOA and the use of standard software packages, it has to be assessed if a SOA enabled standard package is available as a solution. In order to identify packages that might suffice, the SOA Innovation Lab has developed a questionnaire to evaluate the SOA ability of standard software packages and the vendor’s SOA processes and strategy. This questionnaire is based on a SOA architecture maturity framework, which we constructed by integrating different analysis views, using a consistent meta-model approach based on correlation analysis of intrinsic
model elements. For this purpose we have transformed CMMI [Cm09], which is originally an assessment framework for software processes, into a framework to analyze systematically enterprise architectures for packaged based SOA environments. Hereby we have used assessment criteria, maturity domains, architecture capabilities, and level rankings from different SOA maturity models [Ma09]. Additionally we have selected architecture elements from state-of-the-art architecture frameworks like TOGAF [To09], Essential [Es09], and Quasar Enterprise [En+08].

In addition to the overall SOA ability, the functional fit and its SOA ability of the identified package need to be evaluated against specific SOA use cases. After that a high level system architecture has to be developed, which often can be based on integration patterns for the physical integration of systems (see e.g. [PW10, FP03, HW04]).

In this paper we give an overview of our method and the corresponding artifacts. Section 2 provides details about the identification of domains where SOA and standard software is suitable. Use case descriptions and capability map are sketched in Section 3. In Section 4 we summarize key findings which need to be addressed in the ongoing discussion between enterprises and vendors of standard software packages.

2 Domain modeling

When developing an enterprise architecture under the SOA paradigm, a domain map is the central artifact, which structures and organizes the needed capabilities. It forms the topographic base of the enterprise architecture. There are publications, which provide methods how to identify this topographic base, e.g. [En+08]. However, more work needs to be done, if an architect wants to identify areas, where SOA and the use of standard software packages brings benefit. Here, the SOA Innovation Lab introduced the following steps for refinement, which are explained in more detail in [Bu+10]:

1) **Define degree of differentiation and sharing** for each domain. That means, to categorize each domain whether its services have a high, medium, or low level of differentiation, and whether its services can be easily, rather easily, or not be shared throughout different company units.

2) **Refine domains**
   a. **Modularise domains**, i.e. split them up into two or more
   b. **Generalise domains**, i.e. extract identical functions from various domains and group them within a new domain
   c. **Aggregate domains**, i.e. merge two or more domains whenever loose coupling between these domains does not bring any benefits

3) **Finalization:** While defining a domain map, the stakeholders from business and IT needed to be deeply involved. In addition to this, the architect will have produced different versions of domain maps. In this final step he will need to consolidate the different versions and get a final buy-in for his suggestion.
Once an architect has defined a domain map as described above, he will need to characterise domains and provide information on a more fine-grained level. Goal is, to develop a to-be architecture, which marks clearly those areas where the service-oriented usage of standard software packages brings benefits. The architect will do this in an iterative approach, in which he walks through various levels of granularity by decomposing domains into services and subservices, classifying during each iteration the degree of differentiation and sharing, as well as the level of granularity. Figure 1 illustrates this aspect.

Figure 1: Illustration of iterative approach to identify where SOA and standard software packages is suitable

For the domain Sales, the need for differentiation was rated high, since it is in our example crucial for gaining advantages over competitors, to have unique sales processes. For the domain Human Resources there is no need for differentiation. Here the right level of granularity is reached for identifying standard software packages, which suffice the needs. Since the domain Sales is on a very high level of granularity, we subdivided this domain into its main capabilities: Sales Planning, Sales Processing and Sales Delivery. Goal is to find the right level of granularity where areas, suitable for the use of standard software packages, can be identified.

For Sales Processing – again – we have a high need for differentiation since the company needs to assure the shortest throughput times and the best adherence to delivery dates to assure its competitiveness in the market. Because of this, processes must be changed according to changed market requirements and products as quickly as possible – the underlying IT systems must be agile.

For the domain Sales Planning and Sales Delivery, we rated the need for differentiation and agility low. Since, in our example, it would make only little difference, whether the corresponding processes were performed similar to those of the competitors or not. Furthermore, these processes have not changed much in the last years and probably will not change in the years to come.
For the latter two there is a high potential to use a standard software package in a service-oriented way sustainably - if a package can be found, which automates the needed functionality appropriately and is SOA enabled.

For the former the architect will have to do some refinement in order to identify areas on more fine-grained level, where the service-oriented usage of standard software packages makes sense. He will have to identify subdomains and rate them again (illustrated in the right bottom of Figure 1).

These refinement and rating steps have to be iterated until either agility and differentiation is of low relevance, or the domains represent capabilities that correspond to exactly one role and one goal. This stop criterion is necessary to control the level of granularity and to limit the complexity of an overall SOA landscape.

After the architect has iterated through these steps, he will have to identify concrete standard software packages for those domains, for which agility and differentiation are of low relevance. On this basis the architect will develop a to-be architecture, in which fine granular services on function or data level enhance the standard software packages with needed functionality and where standard software packages offer services, which can be combined to higher-level processes.

3 SOA Use Cases and Capability Map

After having identified areas where the use of standard software is suitable, and after having found corresponding standard software packages by a SOAMMI based assessment (see [Zi09]), we are only half way through the process of finding a sustainable SOA architecture, which combines the advantages of different standard software packages and custom built solution. We need to complete the picture by a bottom up approach, in which we deal with concrete use cases and show how to integrate the identified packages.

For this purpose SOA Lab members have collected most important use cases where SOA and standard platforms bring benefits, but with which significant implementation challenges are expected. These use cases go down to the level of singular services, tools and technologies. They are documented on a template basis, which we developed to make the collected use cases comparable and comprehensible. These use cases form the basis for deep dives with architects and vendors. In order to make these discussions as effective as possible, we have formulated key-areas for which specific answers and artifacts are needed, e.g.:

- **Dealing with federated information objects:** Method and best practice to deal with information objects, which are required within different software packages (custom and standard), that need to be integrated in a service-oriented way and which underlie federated governance processes (in the course of a business process there are different stakeholders responsible for such business objects).
- **Dealing with semantic integration:** Method and best practice to semantically integrate in a service-oriented way different software packages which have a functional overlap.
• **Patterns for integration**: Patterns which summarize best practices when integrating software packages on various levels (GUI, process, logical and data integration).

• **Requirement analysis and package identification**: Methods and best practices on how to document functional and non-functional requirements when using standard software packages in a service-oriented way, as well as methods for the identification of standard software packages that suffice these requirements.

In particular we have selected those use cases, for which a detailed investigation leads to answers and artifacts related to the key-areas sketched above. Through this procedure, we have formed a basis for focused workshops with vendors of standard software, which enable us to discuss the service-oriented integration of standard software packages on the basis of real-life problems. In these workshops we are able to focus on the needed level of granularity and can therefore formulate methods, best-practices and patterns by extracting rules that can be generalized for a selected group of problems. This procedure leads - among others results – to patterns for integration, i.e. technical solutions which can be used for a class of problems with practical relevance.

Last but not least, architects deal with quite a number of reoccurring technical tasks every time they want to integrate two or more systems. With tasks we mean, for example, the technical or functional transformation of interfaces, the addressing of components independent of the hardware they runs on, authentication-procedures, etc. Because they occur every time two or more systems need to be integrated, the level of reuse is quite high. Our integration patterns and capability map for integration technologies enable architects to leverage reusable solutions in an efficient way. A detailed description of integration patterns and capability map can be found in [PW10].

### 4 Summary and Outlook

We have sketched a holistic approach for developing a service-oriented enterprise architecture with custom and standard software packages. Besides a description of our method, we have introduced important artifacts. General observations after the assessment of first vendor platforms are:

• SOA is clearly embedded in the strategy of most vendors of standard software

• SOA is often used to provide service-oriented access to data and information of standard software packages

• Currently SOA is rarely used to break-up and disentangle different components of standard software packages

• Until now only few big implementation projects for SOA and standard software platforms have been realized

• Methods and concepts for the implementation and governance of SOA with standard software are mostly available

The SOA Innovation Lab plans further investigations on the usage of SOA within an EAM framework: One area of future activities deals with the construction of software landscapes in the context of monolithic business applications. As a result, we will obtain solution proposals for software landscapes with standard software packages and SOA.
Part of this project is the development of a requirement catalogue for vendors, aiming at more flexibility in the usage of standard software components. Furthermore we intend to detail our general results related to domain modelling and road mapping for complex application landscapes. Here the systematic development of architecture principles in a SOA world is key. Finally, we plan to investigate methods and solutions for the integration of internal and external services in mixed application landscapes, which consist of on-premise and on-demand solutions.

**Literature**


