

Modelling Web-Oriented Architectures

Gunnar Thies

Gottfried Vossen

European Research Center for Information Systems (ERCIS)
University of Muenster
Leonardo-Campus 3, 48149 Muenster, Germany
Email: {guth|vossen}@wi.uni-muenster.de

Abstract

Service-oriented architectures (SOAs) provide the basis of distributed application frameworks where software components are provided as modular and reusable services. Until today there is no generally accepted method for conceptual modelling of a SOA. Rather, there exist several procedural methods which are used in practice. On the other hand, recent developments in the context of what is commonly termed “Web 2.0” show how easy it can be to link or compose (“mesh”) IT components dynamically, so that original SOA goals like flexibility, reusability, or reduction of complexity can indeed be achieved by relatively simple means. An interesting concept in this context is the *Web-oriented architecture* (WOA), which represents a specialization of SOAs obtained by using simple Web 2.0 technologies and standards (e.g., HTTP, SSL, XML). This paper introduces a methodology for designing WOAs, where the big picture follows existing SOA models. In particular, this WOA methodology comprises conceptual as well as realization issues and breaks WOA design down into three distinct phases.

Keywords: Web-oriented architectures, conceptual modelling, design methodology

1 Introduction

Service-oriented architectures (SOAs) provide the basis of distributed application frameworks (W3C 2004b) where software components are provided as modular and reusable services. The benefits of a SOA are seen in the flexibility of business processes which consist of loosely coupled services, and the resulting potential cost decrease, complexity reduction, reusability potential, and high flexibility. Conceptual modelling is an important factor here, as it not only refers to data modelling, but also needs to take process design into consideration. Indeed, having to deal with all kinds of (legacy) systems and databases makes the development of a complex and business-ready SOA a major challenge. Until today there is no generally accepted method for the conceptual modelling of a SOA or for converting other concepts into a SOA. Rather, there exist several procedural methods, including those from IBM (Ganci 2006) and SAP (Woods & Mattern 2006), which are used in practice. Moreover, the various standards for “easing” the creation of a SOA (e.g., Web services, UDDI, SOAP)

typically make the realization of a SOA more complicated (Vossen 2006). Indeed, the large number of standards reflects an “over-standardization” which makes SOAs difficult and complex to implement. Numerous individual aspects of Web Services are defined by over 70 distinct specifications, yet some of them (like UDDI) are barely used (Hagemann et al. 2007).

On the other hand, recent developments in the context of what is commonly termed “Web 2.0” show how easy it can be to link or compose (“mesh”) IT components dynamically, so that original SOA goals like flexibility, reusability, or reduction of complexity can indeed be reached by relatively simple means. Examples include mashups based on Google Maps (like www.housingmaps.com) or applications like Yahoo!Pipes (pipes.yahoo.com); these are based on Web Application Programming Interfaces (Web APIs), which allow using the functionality of a Web application by a simple (most commonly REST-based) service layer. An interesting and emerging concept in this context is the *Web-oriented architecture* (WOA), which represents a specialization of a SOA obtained by emphasizing the use of simple Web 2.0 technologies and standards. Its important aspect is the fact that no additional standards have been defined, but existing ones such as HTTP, SSL, or XML are employed. Since, as mentioned, there are no generally accepted modelling techniques available for a WOA, this paper introduces a methodology for conceptual WOA design which builds upon existing SOA models. In this context the *Business Process Modelling Notation* (BPMN) is used for the definition of relevant processes in every phase. To this end, the BPMN specification¹ is extended by several additional artifacts.

To motivate our approach, a case study originally presented in previous work (Thies & Vossen 2008) is briefly reviewed here; we will later use it for an illustration of our methodology. Our case considers an enterprise that has so far only run stationary shops and now plans to extend its sales operation beyond regional borders. To this end, it wants to move its operation to the Web, but in such a way that available legacy systems can be incorporated into the new IT landscape. Moreover, core business processes need to be captured and implemented, where some parts can be processed automatically, others only semi-automatically.

The core business of the enterprise is the imprinting of textiles and other merchandise with user-defined templates. Customers are offered a wide variety of imprintable articles, which can be decorated with writings or graphics. Orders handled can range from small (e.g., for individuals or clubs) to large quantities (e.g., for companies). Up to now the en-

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology, Vol. 96. Markus Kirchberg and Sebastian Link, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹Since no standard metamodel is defined in the specification of BPMN, a coarse overview of the elements is shown later.

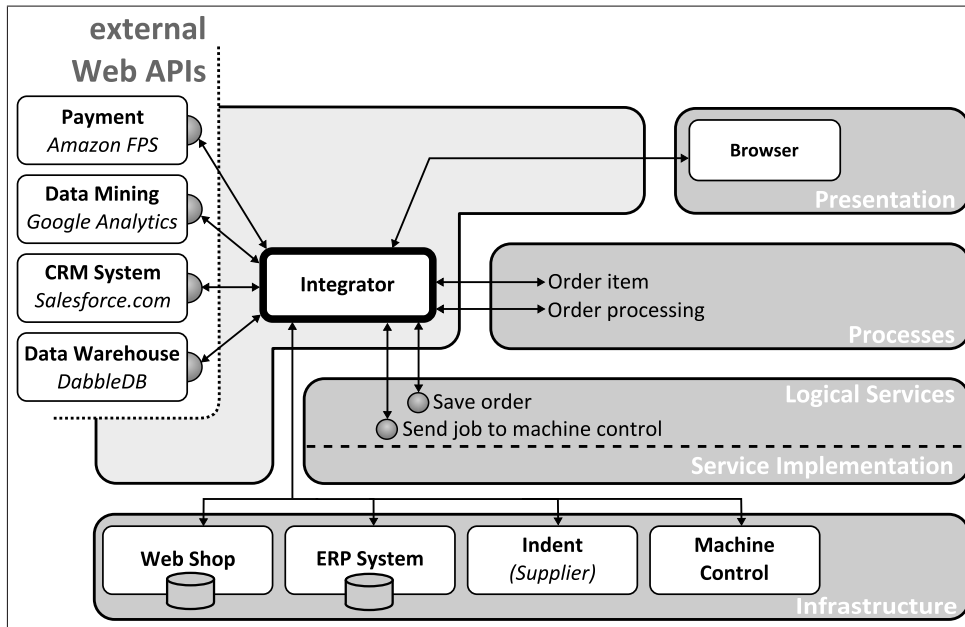


Figure 1: Case study using WOA solution.

terprise runs an ERP system for warehouse and product management. Lithographs are developed together with the customer within a vector graphics application. Moreover, the enterprise runs several machines for printing as well as for flock coating, which are controlled by a central server. The server software is able to accept print commands via a Web service call over the company's intranet. Since machine capacity is not fully used, but the company wants to expand, the business model is to be extended to the Internet and in particular to the Web; consequently, it is to be supported by a new IT system.

The target system is expected to process customer product designs and orders entirely over the Web, so that the customer base can be enlarged considerably. After a transition period, it is even planned to abandon the stationary business completely. Thus, the goal is to implement and run a Web shop alongside the existing ERP system. A possible WOA solution is shown in Figure 1.

As will be seen, the enterprise intends to use as much service offerings from the Web as possible. While this may amount to the design and implementation of a SOA, it has been decided to take a different route: A SOA would typically require a design at several layers of abstraction (Vossen 2006), also indicated in Figure 1, where the infrastructure is lowest, individual services are next, which are topped by service compositions; ultimately business processes as seen by end users are built from these compositions. As Figure 1 shows, a WOA will typically break up this strict division of layers, as individual services obtained over the Web may be employed either as individual service, as a replacement or provisioning of a complex service, or as part of a process. These occurrences may need the help of an integrator component, yet they show that a different design methodology is needed. While SOA design can either follow a bottom-up approach, a top-down approach, or a combination of both ("meet in the middle"), a WOA design can no longer simply follow either of them. A methodology that can be used in WOA design is the subject of this paper.

It should be noted that the fact that a WOA typically breaks up the layer division of a SOA has the

consequence that a certain amount of programming will be needed during the development of a WOA. In other words, the model of a WOA that reflects its architecture and composition and that results from a development process will not be entirely conceptual; instead, it will actually be a hybrid model that comprises both conceptual as well as physical issues. We consider this a kind of "price to pay" for the fact that a WOA is much easier to develop and deploy than a SOA.

The remainder of this paper is organized as follows: In Section 2 we give an overview of related work in the field of conceptual SOA modelling. In Section 3 we then present our methodology which extends and derives from already known methods, in order to meet the requirements for an appropriate WOA methodology. This motivates Section 4, where parts of the case study will be realized using the novel WOA methodology. Finally, in Section 5 we discuss open challenges and future work.

2 Related Work

Service-oriented architectures (SOAs) have been discussed intensively in science and industry since their first appearance in the mid-90s. However, even today there is no generally accepted method for modelling a SOA. The various approaches that have been described in the literature aim at methodologies for planning and implementing a SOA, but none has become a "standard" yet. We review available methods next.

First of all there is the *Web Service Architecture* (WSA) of W3C (2004b), where a system of Web Services and their relations to each other are conceptually described. Here, specific aspects are considered: the Service Model, the Message Oriented Model, the Resource Oriented Model and the Policy Model. This approach mainly aims at the usage of Web Services and therefore is less general than our approach. Another approach inspired by WSA comes from the EU project *Service Centric Systems Engineering* (SeCSE) where Colombo et al. (2005) are focusing more on some clarification in addition to the WSA model e.g., relationships between the concepts

of service description, semantics, and service interface. This approach is also focused on Web Services and not as generic as our approach.

More into a semantic approach is OWL-S as outlined in W3C (2004a), which describes a service and its metadata in detail. Here the focus lies more on the semantic description of a single Service within an ontology than on a whole system. Also used for the semantic description of a service is the *Web Service Modeling Ontology* which currently exists as a final draft only, see WSMO Working Group (2006). Since we are focusing on the whole architecture, these approaches are only relevant as far as functional descriptions are concerned.

Another important approach is the *Service Oriented Modelling and Architecture* proposed by IBM (cf. Arsanjani & Allam (2006)) where modelling of a SOA is divided into three phases. The first phase deals with *Component Business Modelling* (CBM) for describing all business processes within an enterprise. The second phase is used to connect processes with services, and the third phase deals with the implementation of the found services within a SOA. This conceptual view is, among others, the basis of the approach described in this paper. Furthermore, we provide a brief description of a conceptual process model which deals with the creation of a WOA from scratch.

More specific is the approach by Mos et al. (2008), which is a method for designing a SOA with the goal of using an *Enterprise Service Bus* (ESB) and therefore Java Business Integration. This method is restricted to a Java environment, but the three phases of the (top-down) method are a good starting point and close to the IBM approach: First discover and describe the processes via *Business Process Execution Language* (BPEL) or BPMN by a *Business Process Designer*, then create a Service Component Architecture descriptions by a *Software Architect* and finally implement and deploy the various services within the ESB by a *Technology Team*.

Another approach by Quartel et al. (2007) named *COncceptual Service MOdelling* (COSMO) focuses on basic concepts to represent essential, elementary, and generic service properties. Here processes are described within three levels of abstraction: as a single interaction, as choreography, and finally as orchestration. For the description of services and their interactions the *Interaction System Design Language* and other languages like OWL, SPARQL, and Java are used. This in a sense contradicts the general idea behind a WOA, which is to use simple and commonly known Web techniques. On the other hand, the division of the provider on the one side and the consumer on the other side and their connection pattern within a process is close to the differentiation of services by a “pool” in our approach.

A more programmatic approach uses the Unified Modeling Language (UML) for modelling a SOA (López-Sanza et al. 2008). It defines a UML profile for the design of PIM-level SOA-based architecture models. In contrast to our approach, processes are less important; not only service providers and services are defined within the SOA metamodel, but also so-called *Active Components*, which handle service request and responses and interact with the frontend. These components are implicitly visible in our approach as well, namely as tasks within the pool named “integrator” of a process model.

3 The Approach

In this section we describe our approach to conceptual WOA design. We start out by collecting various requirements and setting several goal. After that

we present the methodology, which is comprised of three phases: process modelling, refinement, and implementation. Each phase will be described in necessary detail.

3.1 Requirements and Goals

The existing methodologies for designing a SOA mostly focus on Web Services as the core connection element. In contrast, a WOA uses Web services as well as RESTful services, i.e., services that can be invoked via REST. As a consequence, a WOA approach is not bound to a specific technology. While Web services can be described by a WSDL document (cf. W3C (2007)) for RESTful services (cf. Fielding (2000)) there is no explicit standard for their description. However, a reasonable solution for the time being is the *Web Application Description Language* (WADL) which is much simpler than WSDL, yet allows the description of important features of a RESTful service such as resource, input, output, and even a textual description, see Hadley (2006). Most services today come with textual descriptions only (as can be seen, for example, at www.programmableweb.com), so no matching algorithm can be specified. Therefore, most services have to be “discovered” by the software architects themselves. This can be done in different ways, but our approach currently assumes that the search for suitable services is done by hand in the second phase. As long as no accepted semantic service description is available and used for a large portion of the services offered on the Web, according to experience no reasonable registry mechanism will work.

One of the important points of our approach is the usage of well-known and well-understood formal languages in every phase, including BPMN for process descriptions and WSDL as well as WADL for the functional description of individual services.² The advantage of BPMN is that it is easy to read and still allows for a powerful visual design of processes.

The following simplified steps are identified to model, construct, and implement a WOA:

- Development a set of inter-connected business process models;
- refinement of these process models with functional details of each service task;
- provisioning of the process models with explicit data flow (especially a specification of which data is used as input for a service request);
- implementation of the defined process models (in a development environment or workflow engine).

Similar to approaches that have been described in the literature (e.g., the SOMA method of IBM Arsanjani & Allam (2006) or Mos et al. (2008)) the basis of our approach is built upon three different roles and has three main phases (see Figure 2). These roles are the *Process Designer*, the *Software Architect* and the *Technology Team*. In a small enterprise these roles will often be overlapping and assigned to the same person. Otherwise, communication is obviously needed between those with one of these roles assigned.

3.2 The Methodology

We go through the three phases of our design methodology next.

²Plans for the near future contain the implementation of a *WSDL2WADL* tool which is able to convert service descriptions between these two description languages without loss of information.

3.2.1 Process Modelling Phase

In the first phase the Process Designer has to define the process models for a specific domain. We propose the usage of BPMN because of its readability (even for non-technologists), its widespread usage, and the good tool support. A standard-conforming BPMN model basically consist of the following components (compare Object Management Group (2008)):

- Flow Objects: Event, Activity (Task), and Gateway
- Connecting Objects: Message Flow, Sequence Flow, and Association
- Artifacts: Data Object, Group, and Annotation

A process contains at least one “pool” representing an organization or a role in the process. “Lanes” can be defined within a pool to sub-divide a pool (e.g., according to different roles or sub-systems).

In this first phase the Process Designer has to model each business process of the enterprise with non-functional descriptions. The following steps should be carried out for each particular business case:

1. Identify the systems comprised (e.g., ERM or CRM system) and add a new pool for each.
2. Analyze the business case, whether there are different roles or parts of a system and add as many lanes as needed within the specific pool.
3. Construct the process following the BPMN specification.
4. Mark each task which requests a service as “service task”. These tasks will later be refined by a Software Architect.

After the first phase, all process models will be defined and give an overview of all systems and roles used within the enterprise. The result can now be compared, for example, to a *Component Business Matrix* of the IBM methodology. The lane *Process Designer* in Figure 2 shows this step as the first task of the methodology process (also designed in BPMN).

3.2.2 Refinement Phase

The second phase is carried out by a Software Architect, since now the tasks of a process, which have been identified as service tasks in the first phase, will be described in detail. This phase is divided into two main actions:

1. First, the appropriate services for a service task are searched for, and their functional specification and behavior is described in an adequate way. For a Web Service WSDL will be used, while a RESTful service will be described using WADL. In this phase the resource URL, the input and output parameters as well as the service description are defined.
2. The data flow within the process models is described for each service. Any (specific) mapping of data from one service to another will be described with regard to the functional description of the service (meaning an exact overview of in- and outflow of data).

In the first part of this phase some kind of service repository can be employed. This could be a repository like www.programmableweb.com or any other internal or external repository of the given enterprise

(see Hagemann et al. (2007) for an overview). A certain difficulty obviously lies in searching for an adequate service which optimally fits the purpose of the business process. Notice that in SOA models, often a repository based on UDDI is proposed, which has once been designed for an automatic matching of services to a given description. The various problems that arise while detecting a suitable service for a SOA (not limited to UDDI) have been investigated in Letz (2007), and as shown in Hagemann et al. (2007), repositories based on UDDI are no really in use anymore.

So far the most complete service directory for RESTbased services is www.programmableweb.com. If that is not sufficient, either a formal description needs to be connected with a service task (done by the Process Designer) which can be understood by a Software Architect, or the Process as well as the Software Designer need to communicate while refining the processes. In some cases, these two roles will be combined in one person. Also, the Software Architect should have a broad knowledge of the internal IT systems and hence be able to specify any service parameter. If a service task can be linked to a real service, the process model needs to be adapted. Therefore, adding or changing a pool or a lane within the process is needed to show the sequence flow from system to system. A pool called “Integrator” will always exist, from which all requests made by services will be started and their responses computed. To support the addition of functional descriptions to a task, the underlying BPMN metamodel needs to be extended with an artifact element called “functional description”. An instance of this element can then be connected to a task.

In the second part of the refinement phase, the data flow within a service is described. Through the information of the given WSDL and WADL descriptions, the parameters needed to call a service are outlined. Data objects can be used to define data that is saved or available throughout the process. Besides the sequence flow of a BPMN process, which defines the deterministic flow of the process, the message flow shows when and between which tasks of different pools messages are exchanged. Therefore, to define which parameter is used for a service request additional information is defined for the message flow connector. Even if our approach is designed to be technology independent, here the BPEL construct of assigning data to an activity will be used (see OASIS (2007)). We do so because on the one hand this construct supports the possibility to convert BPMN process models to BPEL processes and on the other hand because of the small footprint in contrast to other alternatives (like the *Web Service Flow Language* which builds heavily onto WSDL (Leymann 2001)).

Listing 1 shows how the assignment of parameters is defined for a message flow connector in XML notation. The `container` attribute in the `from` tag refers to a data object of the underlying process, where the `part` points to attributes within a data object. The `to` tag, in contrast, refers to a service task.

```
<assign>
  <copy>
    <from
      container="Session Data"
      part="customerId"/>
    <to
      container="Check Id"
      part="id"/>
  </copy>
  <copy> ... </copy>
</assign>
```

Listing 1: Assignment of data.

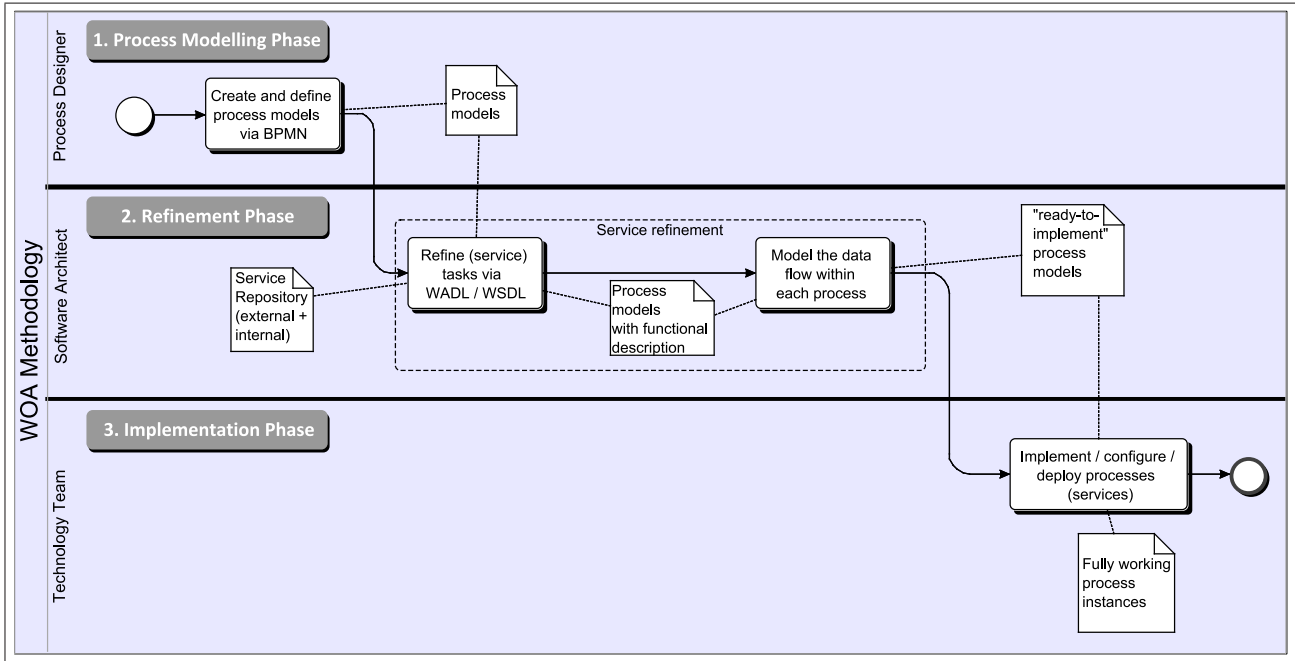


Figure 2: The three phases of our WOA methodology.

We mention that this phase is equivalent to the second step of the IBM method where services are bound to processes of the CBM matrix. The result of this phase are process models with a functional description of the specific services and their data flow.

So far there is no specific graphical representation for defining a functional description or a data flow element in a BPMN process, besides the normal artifact description. Therefore Figure 3 shows a prototypical construction of a process part, where a service task is refined with a WADL description and an in-going data flow from another task.

Since the BPMN specification explicitly allows the extension of BPMN models by artifacts, the necessary data flow element as well as the functional description are added as an artifact type. Additionally, the data flow artifact can only be bound to a message flow. In Figure 4 an overview of all the BPMN elements is shown including the gray elements for use in the second phase as extensions to the standard.

3.2.3 Implementation Phase

The third phase finally brings the WOA to life. The result of the two preceding phases are fully defined processes with the specific services and the integrator as a coordinating platform. One of the great hopes in service computing has always been a fully automatic code generation where a process is “designed, not programmed”. In some cases this is indeed possible, e.g., for a process which has no manual task in it and which uses mainly external services where the logic inside the integrator is minimal. Therefore, the BPMN process should be either directly run by an engine which can handle data in data objects, can send requests to any Web-based service (described by WADL or WSDL), and understands the flow logic of the integrator, or the process should be translated into a common workflow format like BPEL and then run in a workflow engine.

Since BPMN has no underlying metamodel, there are no generic workflow engines which could run a BPMN process. However, some tools are able to simulate a process. The latter approach has been inves-

tigated by Ouyang et al. (2006), and even the specification of BPMN addresses a conversion of BPMN patterns into BPEL4WS. Unfortunately, a standard BPEL process is not able to handle RESTful services out of the box, so the “RESTful BPEL” approach propagated by Overdick (2007) can be helpful in this respect. In most cases, however, the technology team has to integrate external services into the process flow, either by programming a wrapper for a service (in case that there is only something like a JavaScript API) or by taking a look at the *Service Level Agreements* (SLAs) of the service. Our approach aims at a WOA, so the integrator component should be a Web-based platform. This could be providers of Platform-as-a-Service solutions like *bungeeConnect* or *force.com*, or other integrating components.

Basically, the following steps are required for each process which is not automatically processable:

- Check for every pool that is not the integrator pool whether the services described are reachable and also whether or not the functional description is correct.
- Check the given SLAs and verify the guarantees if possible.
- Test the services and consider alternatives (think about load balancing and redundancy scenarios).
- Write the code needed for the integrator component and bind the services to the process.
- Conduct test runs of the finished processes.

These steps are closely related to a standard software engineering process, where the business cases are implemented, tested, and then enabled for production use. There is also a similarity to the third step of the IBM method, where the services are deployed in a SOA environment, or to the model of Mos et al. (2008), where the software components are finally deployed in an ESB.

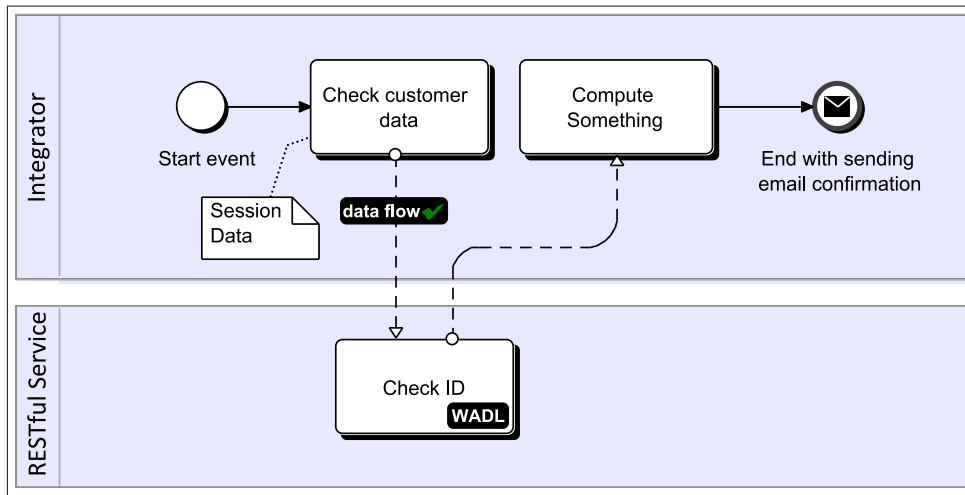


Figure 3: Refined process model after second phase.

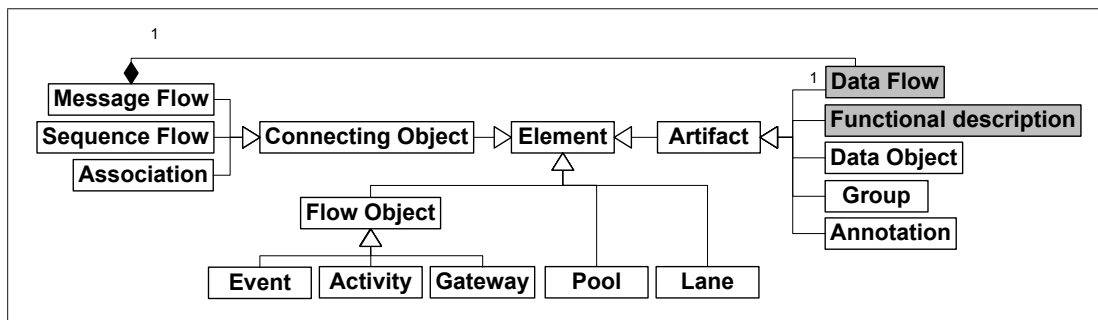


Figure 4: Extended BPMN object model.

4 Applications

We now employ our WOA design methodology using the payment for items in the *Web Shop* which has been presented in detail in Thies & Vossen (2008) as a case study. The starting point of the methodology is the creation of a process model in the *Process Modelling Phase*. So the Process Designer models the Order process (at a coarse level). Only the “integrator” pool is modelled in the first place, where all the process steps take place. The process covers the following action:

“A customer chooses items from the web shop (so there is a filled shopping cart) and clicks on a “pay” button. If the customer is logged in, the “payment process” task is starting, otherwise he/she is forwarded to a login page. The task returns either a positive or a negative response. The latter case results in handling an error and end the process. If the payment is successfully done, the “order” task is starting, where the items in the shopping cart are saved as order in a CRM system. The process ends with an email notification to the customer.”

Two tasks are identified as services by the Process Designer and marked as “service tasks:”

1. *Payment process service*
2. *Create and save order service*

Both tasks are placeholders for a Web or RESTful service which will later be designed in the Refinement Phase. Figure 5 shows the final process of the first phase.

The following step in the *Refinement Phase* is to extend the process and hence to find suitable processes for the identified service tasks. For the payment process the *Amazon Flexible Payments Service* (FPS) is a candidate, the order process is covered by an imaginary test system (to show how a RESTful service would look like). First, two additional pools are created by the Software Architect, one for each external service, and each pool gets a task, which stands for the specific Web or RESTful service. After that the message flow is designed to reach the specific task within each new pool. For a better understanding of the process, the Software Architect adds grouping artifacts to the process, so that a former single task (like the payment process) is graphically outlined.

The important part now is to describe the tasks with functional descriptions. For the payment process, the WSDL document provided by Amazon is connected to the task. The order process gets a functional description via a WADL document. Both description are indicated by a small black rectangle in the task symbol, which displays the special type of service description.

The final step in this phase is to define the data flow between the different pools in the process. As shown in Figure 6, the data flow is displayed as a data flow artifact connected with the message flow. It shows a (green) checkmark if the data flow is already defined; otherwise it shows a (red) cross. In our example, the data flow from task “Save order” to task “Create and save order” is not defined yet. We assume that the WADL definition of the order process needs parameters `apiKey`, `customerId`, and `shoppingCart` as input (see the `param` tags in lines

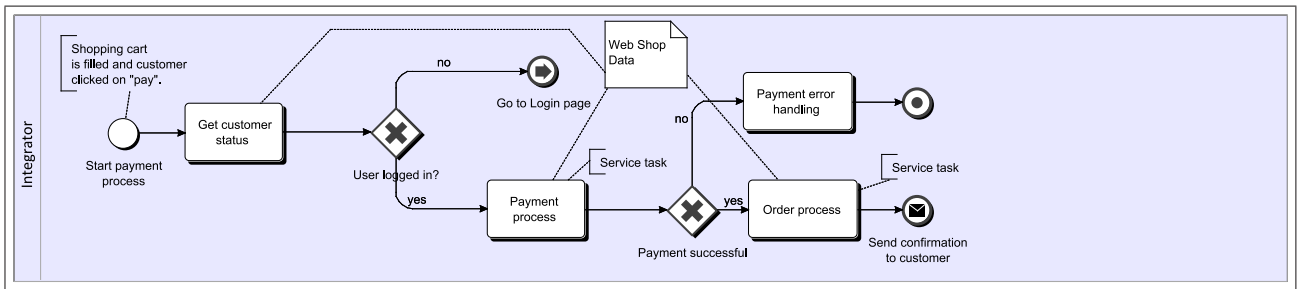


Figure 5: Order process after first phase.

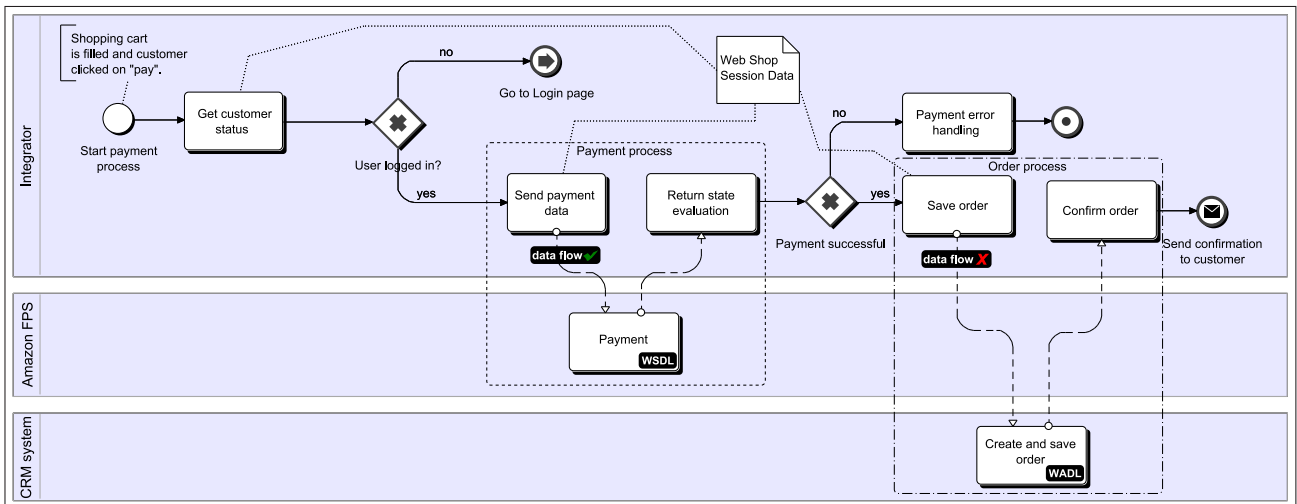


Figure 6: Extended order process during the second phase.

10, 13, and 14 of Listing 2). These parameters are held by the data object “Web Shop Session Data” in the integrator pool.

```

1 <?xml version="1.0" ?>
2 <application targetNamespace="urn:crmsystem"
3   xmlns:crm="http://api.testsystem.org/
4     namespaces/">
5   <doc xml:lang="en" title="documentation">
6     Simple service for creating an order.
7   </doc>
8   <resources base="http://api.testsystem.org/
9     Order/V1/">
10    <resource path="{apiKey}/createOrder">
11      <param name="apiKey" style="query"/>
12      <method name="GET" id="search">
13        <request>
14          <param name="customerId" type="
15            xsd:string" required="true"/>
16          <param name="shoppingCart" type="
17            crm:shoppingcart">
18        </param>
19        </request>
20        <response>
21          <representation mediaType="text/plain"/>
22          <fault status="400" mediaType="text/
23            plain"/>
24        </response>
25      </method>
26    </resource>
27  </resources>
28 </application>

```

Listing 2: WADL description for the “Create and save order” service.

der”, where the parameters of the session data are mapped to the parameters of the RESTful service. Every parameter is copied via the from tag to the destination, the “Create and save order” task. With the data now defined, the service request can be implemented later by the Technology Team without knowing details about the business case.

```

<assign>
  <copy>
    <from
      container="Web Shop Session Data"
      part="apiKey"/>
    <to
      container="Create and save order"
      part="apikey"/>
    </copy>
    <copy>
      <from
        container="Web Shop Session Data"
        part="customerId"/>
      <to
        container="Create and save order"
        part="custId"/>
      </copy>
      <copy>
        <from
          container="Web Shop Session Data"
          part="shoppingCart"/>
        <to
          container="Create and save order"
          part="shoppingCart"/>
        </copy>
      </assign>

```

Listing 3: Assignment of data flow.

Listing 3 declares the data flow between “Save order” and the CRM system task “Create and save or-

The final *Implementation Phase* is not described in detail here. Basically, the integrator platform needs

to hold a data object with the session data of the Web Shop (therefore the Web Shop provides interfaces for reading and writing session data). The Technology Team also has to implement one Web Service request to Amazon FPS and a RESTful service request to the CRM system.

5 Conclusions and Future Work

In this paper we have presented a WOA design methodology which abstracts from technology and complex standards and only uses simple Web standards like HTTP, SSL, and XML for communication. In addition, the usage of the SOAP protocol is also needed in cases of Web Service-based systems. For the definition of a service we propose WADL (or WSDL for Web Services) and a simple data flow syntax to describe data mappings for any request within a process. No further standards are needed to describe a fully working WOA.

Our methodology comprises three phases, which are partially also reflected in some of the approaches for the construction of a SOA (as outlined in Section 2). Every phase addresses a distinct role, so that different kinds of knowledge about the processes of an enterprise can be covered. BPMN is used for all phases to define and model the respective business processes, because the benefit of BPMN is the readability and the extensibility. The usage of service descriptions is proposed and with WADL a high degree of technological independence is possible, since WADL is not bound to a specific service or protocol format.

As mentioned in the Introduction, we have seen that our methodology is not just purely conceptual, but rather a hybrid one that meshes conceptual as well as physical aspects of a WOA. We consider this a consequence of the fact that a WOA no longer needs to follow the strict layering of a SOA, but we believe that it is exactly this aspect what will make them more successful than SOAs.

One important aspect of services provided over the Web are SLAs, which define the guaranteed benefits of a provided service. SLAs are an agreement between a service customer and a service provider and often contain information about the availability, stability, and – most importantly – costs of a service. In this paper SLAs have not been considered, but in future work they will be discussed in detail in the context of the Refinement Phase of our methodology. This will lead to the possibility of estimating overall cost of a service, which will be helpful for a *return on investment* (ROI) calculation.

Tool support is another important factor for the use of a methodology. After some testing, we have chosen a plug-in called *SOA Tools Platform Project* (STP) for the Eclipse IDE, which allows for the modelling of BPMN processes. Choosing an Eclipse plug-in also makes it possible to extend the plug-in in the future. So all the added artifacts (and maybe more for SLA usage) to the BPMN metamodel can be constructed and used to create a WOA modelling tool in eclipse.

Another interesting part will be an elaboration of the integrator functionality. Besides existing platforms like *bungeeConnect* or *force.com*, which are so called Platform-as-a-Service providers, there is much space for the development of a BPMN workflow engine which is able to run and administrate BPMN processes directly.

Acknowledgement. The authors are grateful to the reviewers of this papers, whose comments have led to several improvements over an earlier version.

References

- Arsanjani, A. & Allam, A. (2006), Service-oriented modeling and architecture for realization of an soa, in '2006 IEEE International Conference on Services Computing (SCC 2006), 18-22 September 2006, Chicago, Illinois, USA', IEEE Computer Society, p. 512.
- Colombo, M., Di Nitto, E., Di Penta, M., Distanto, D. & Zuccal, M. (2005), Speaking a common language: A conceptual model for describing service-oriented systems, in 'ICSOC 2005', Springer Verlag Berlin Heidelberg 2005, pp. 48–60.
- Fielding, R. T. (2000), Architectural Styles and the Design of Network-based Software Architectures: PhDThesis, PhD thesis, University of California, Irvine.
URL: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- Ganci, J. (2006), *Patterns: SOA foundation service creation scenario*, IBM Redbooks, 1st ed. edn, International Technical Support Organization, Poughkeepsie NY.
- Hadley, M. J. (2006), 'Web application description language (wadl)'.
URL: <https://wadl.dev.java.net/wadl20061109.pdf>
- Hagemann, S., Letz, C. & Vossen, G. (2007), Web service discovery – reality check 2.0, in 'Proc. 3rd International Conference on Next Generation Web Services Practices (NWeSP), Seoul, Korea', pp. 113–118.
- Letz, C. (2007), Web Service Detection in Service-oriented Software Development: A Semantic Syntactic Approach, PhD thesis, Westfälische Wilhelms-Universität, Münster.
- Leymann, F. (2001), 'Web services flow language'.
URL: <http://xml.coverpages.org/WSFL-Guide-200110.pdf>
- López-Sanza, M., J. Acuña, C., Cuestaa, C. E. & Marcosa, E. (2008), Modelling of service-oriented architectures with uml, in 'Electronic Notes in Theoretical Computer Science Vol. 194 Issue 4, Proceedings of the 6th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2007)', pp. 23–37.
- Mos, A., Boulze, A., Quaireach, S. & Meynier, C. (2008), Multi-layer perspectives and spaces in soa, in 'SDSOA '08, May 11, 2008, Leipzig, Germany', ACM 2008.
- OASIS (2007), 'Web services business process execution language version 2.0'.
URL: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>
- Object Management Group (2008), 'Business process modeling notation, v1.1'.
URL: <http://www.omg.org/docs/formal/08-01-17.pdf>
- Ouyang, C., van der Aalst, W. M., Dumas, M. & ter Hofstede, A. H. (2006), 'Translating bpmn to bpel'.
URL: <http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2006/BPM-06-02.pdf>
- Overdick, H. (2007), Towards resource-oriented bpel, in '2nd Workshop on Emerging Web Services Technology', Aachen.

- Quartel, D. A., Steen, M. W. A., Pokraev, S. & van Sinderen, M. J. (2007), Cosmo: A conceptual framework for service modelling and refinement.
- Thies, G. & Vossen, G. (2008), Web-oriented architectures: On the impact of web 2.0 on service-oriented architectures (to appear), *in* Proc. 2008 IEEE Asia-Pacific Services Computing Conference, Yilan, Taiwan.
- Vossen, G. (2006), Have service-oriented architectures taken a wrong turn already?, *in* 'IFIP TC 8 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2006), Vienna, Austria', pp. xxiii-xxix.
- W3C (2004a), 'Semantic markup for web services'.
URL: <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
- W3C (2004b), 'Web service architecture'.
URL: <http://www.w3.org/TR/ws-arch/wsa.pdf>
- W3C (2007), 'Web services description language 2.0'.
URL: <http://www.w3.org/TR/wsdl20-primer/>
- Woods, D. & Mattern, T. (2006), *Enterprise SOA: Designing IT for Business Innovation*, O'Reilly Media, Inc.
- WSMO Working Group (2006), 'Web service modeling ontology'.
URL: <http://www.wsmo.org/TR/d2/v1.3/20061021/>

