Designing a Platform-Independent Use-Case for a Composite Application using a Reference Architecture

Helge Hofmeister* and Guido Wirtz
Otto-Friedrich-University Bamberg, Distributed and Mobile Systems Group
Feldkirchenstr. 21, 96052 Bamberg, Germany
hofmeister@ecoware.de, guido.wirtz@wiai.uni-bamberg.de

Abstract

This paper describes the design of an industry-scale use-case for a composite application. The described design is based on former work of the group which basically investigated the reference architecture and development methodology for composite applications [8, 9]. The paper puts its focus on the platform independent design of the composite application from the business-process based requirements down to the design of the required services. Additionally a meta-model is provided that describes the hierarchy of services that need to be designed in order to meet business requirements in a more flexible way.

Keywords: SOA, composition, reference architecture

1. Introduction

Service-oriented architecture (SOA) is currently a buzz-word hitting the industry. Following the discussions in, e.g. [4] or [14], service orientation might be more or less just a relaunched distributed object approach. Although being recognized in the software industry most customer-industries, however, are still questioning the benefit of this approach.

We decided to evaluate the business value of SOA in a large-scale project for an IT service provider in the area of chemical industry. One key part of this (ongoing) evaluation is to apply the service-oriented paradigm to a real-life industry-scale use case. Service-orientation should proof that it brings value into a solution landscape with various legacy applications partially supporting business processes.

Our previous research on SOA in the same context resulted in the definition of a reference architecture [8, 9] as well as a proposal for a methodology [10] that can be used to design service-oriented applications - so-called composite applications - according to the reference architecture. This paper discusses our findings in applying this methodology to the design of the real-life use-case from industry and validates our results.

We refer to [8–10] for an in-depth discussion of related work regarding our overall approach. The work on service-design as described in [6] and [11] as well as the work of the SDO [3] specification in terms of data standardization have influenced the design we present in this paper. Moreover, the transformation of event-driven process chains (EPC) into WS-BPEL [2] executable processes as described in [5] is important for the case study since it allows for the business-process centered approach we follow.

The rest of the paper is organized as follows. After giving a sketch of our approach in section 2, section 3 discusses the design case study in detail. The paper concludes with an outlook on future work in section 4.

2. A Sketch of the Approach

The design of the pilot was done using a development methodology as well as using a reference architecture mutually corresponding. In order to meet both, a clean design and the industry-scale reality of predetermined tools of the corporate environment, we decided to split the design into three major sections: The platform-independent model (PIM), the platform-specific model (PSM) and a landscape architecture[1]. The aim is to design the composite application based on the tool-independent reference architecture. Using a generic mapping of the architecture to a specific set of tools and products, the PIM is specified with regards to the actual products in a later phase. The PSM and the concrete realization are then vivid demonstrations of the real-life applicability of the elaborated concepts. The actual tool-independent application of the concepts to real-life business requirements proofs the applicability of the design principles. Finally, the landscape architecture is the topology of deployed products and servers.

3. The Case Study

The use-case of the pilot project was chosen by a dedicated internal working group. Lacking a formal methodology, the group analyzed on a business expert level several business processes and how steps that are included in these processes

*Helge Hofmeister is an external PhD student with the Distributed and Mobile Systems Group at University of Bamberg.

[1] We still investigate the suitability of the model-driven architecture (MDA) [7] to our approach.

might be used in other processes as well. Hence, the advantage of service-oriented architecture that was best understood by the business analysts was reusability across system boarders. The business process finally chosen was the so-called ”agreement management” business process. This process describes the procedure how the company reacts on customer demands by estimating the request and providing an offer to the customer.

3.1. BP-based Functional Requirements

The business process is used as a functional requirements specification for the composite application (see Figure 1 for an extract modelled as an EPC). It involves control and data flow as well as the organizational/system flow. The process describes a collaborative sequence where different experts from single delivery units work out descriptions of the offer as well as cost estimations. In the subsequent phase, the account management (AM) uses this information to generate an offer that might lead to a contract with the customer.
Up to now, this process is supported by various applications: Sales and distribution (SD) as well as project management (PM) use an ERP system. Additionally, there is a change-management database that is employed in the different ITIL processes of the company [1]. This database is used to store service-level agreements, service descriptions and assets that are required for performing the services. The account management today uses a CRM application for managing offers and contracts. The aim of the case study is to automate this process using the existing system landscape by building a composite application unifying also the user access to the systems.

3.1.1. Enterprise Service Matching

The business process describing the requirements is used as the top-level orchestration of the involved processes. Following this approach, EPC functions (aka. Business tasks; green boxes) are so-called enterprise services. In order to allow the concrete implementation in terms of a service orchestration, the enterprise service themselves might need to be aggregated services of lower levels. In order to determine whether such an aggregation is required, the design methodology starts with an ”enterprise service matching”-phase. This phase describes the determination of native services being exposed by applications (application services) that might fit to the described function. Since our pilot project is the first experience with implementing composites, there are no matching enterprise services.

3.1.2. Data Interaction Analysis

In order to set-up new enterprise services matching the business functions, a data-interaction analysis based on the functional requirements is performed next. This step mainly analyzes which data are required by the single functions as their respective input and output. Having identified these data, the required data structures may be deduced from the enterprise’s canonical data model.

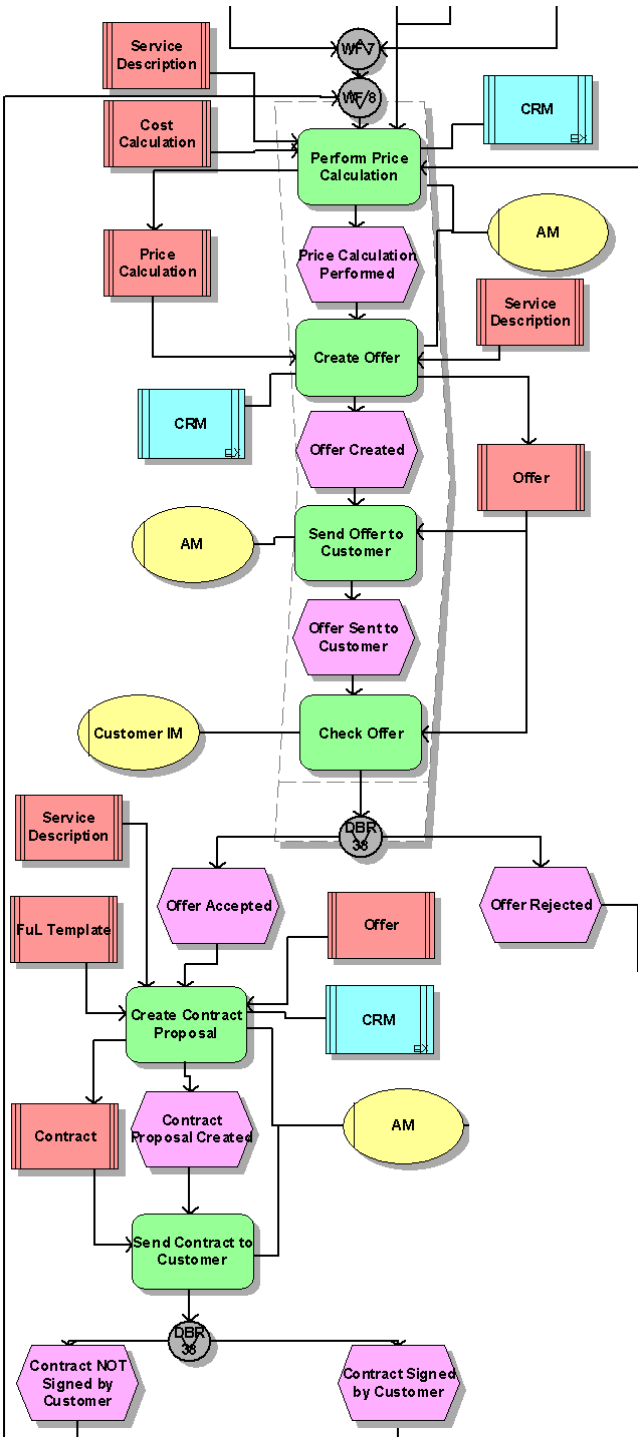

Figure 1. EPC for Functional Requirements

3.1.3. Transactional Property Identification

Knowing the data and the respective data access performed by the to-be enterprise services, some transactional properties of the process flow can be identified. Meeting the characteristics of business processes being long-running transactions, no ACID transactions involving multiple enterprise services are

identifiable. Though, global transactions with relaxed ACID properties can be identified. In the business process snippet in Figure 1, the set of enterprise services performing a global transaction are marked by an underlying arrow (”Perform Price Calculation” till ”Check Offer”). In the given example the functions are grouped as a single global transaction since the state reached after checking the offer might require a new cycle of the offer phase that would replace the previously created offer. It is notable that all global transactions identified in the business process occur in process branches being encapsulated by a workflow pattern (8 in [13]) and a data-based routing pattern (38 in [12]).

3.1.4. Service Composition

This phase is the main phase of the PIM because it describes functionality that is independent of any target architecture. Nevertheless is this the phase creating the design of the enterprise service's implementation. As composite applications are mainly orchestrations of functionality being already existent in legacy applications, the main aspect of our design phase is the description of how application functionality that is accessible by the means of ”application services” is aggregated by the service coordination layer. The design of the service composition involves all layers of the underlying reference architecture, though. This is because the service coordination layer's design describes also the requirements for lower layers. These are mainly requirements for the data-exchange and transformation layer and the data repository. Often, the coordination describes an additional requirement for the design of new application services. This is because application functionality may not always be defined by means of services[2]. The design method and the example service design are described in the next section.

3.2. Enterprise Service Design

The design of the service coordination is based on three requirements: The description of the business functions that results in the enterprise services, the data flow and the corresponding data model that are computed by the business functions as well as the organizational perspective describing how humans and application systems are considered to interact in order to perform a certain business task. Notable about these three items is that the actual service designed is completely determined by the requirements expressed in the business process. Even the application systems that need to act as agents for the service provider part of the application services, are identified by that description. The approach incorporates these aspects since business experts turn out to often know precisely which legacy systems are capable of providing the required functionality - and these determinations are usually to be kept in order to allow for an agile deployment of composite applications.

[2]”means of services” refers not to the characteristics of services of using a common protocol. Furthermore does this refer to the encapsulation of functionality into coarse-grained and stateless units of work - services.

The starting point of the service coordination design is the actual business task. As an earlier step may already have identified existent enterprise services (”perfect matches”), the service coordination might be obsolete. Whenever no enterprise services have been matched, the initial decision of the coordination's design is whether the business task could be mapped to either an application system or a user interaction[3]. This is a design decision that has to consider two influence factors: The possibility of a single service agent to implement the functionality completely and the potential reusability of the enterprise service. Whenever parts of the required functionality can be mapped to existent application services, the services shall be reused and the enterprise service therefore will be composed of several application services.

Usually the coordination involves both reusing application services and designing new application services. During the design phase of the case study it turned out that the design of new application services is an informal procedure that relies on a subjective design of the solution architect. This is the case since the to-be application services need to fulfill immediately the requirements of the business task. Hence, the first step of the application service design is to model application services that support the requirements. This approach results in a process flow at the coordination layer that orchestrates several application services that compute data. Which data is actually computed results out of the data interaction analysis of the business process and the identified data model.

Since the identified data determines the context of both, the business process as well as the coordination process, the reference architecture relies on a data repository the context is kept in. Even if the context data is either retrieved from legacy applications or is created by human users during the process cycle, data access is not directly modeled as calling application services. The data repository is included in the coordination design as a separate entity (cf. [8, 9]).

The design steps described so far lead to two classes of services: enterprise services (that are by definition the tasks of the business process) and the steps of the coordination layer. The latter are referred to as ”coordination services”. Coordination services are introduced in order to allow for the design of non-existent application services. Coordination services might be entity-centered or task-centered. Entity-services are used to read or manipulate data kept by the legacy applications. Entity-centered services utilize application services in order to make data available to the context. Task-specific coordination services finally provide the functionality that operates on this context. The meta-model of our service hierarchy is shown in Figure 2.

Including the data repository into the reference architecture enables stateless services that are better reusable as the data repository respectively the entity-services manage the state of the composite while all participating services do not need to keep a state. In order to allow for reusable services, stateless

[3]Even if the reference architecture handles user-interactions like every other legacy functionality, it turned out to be helpful to describe application services and user interactions separately.

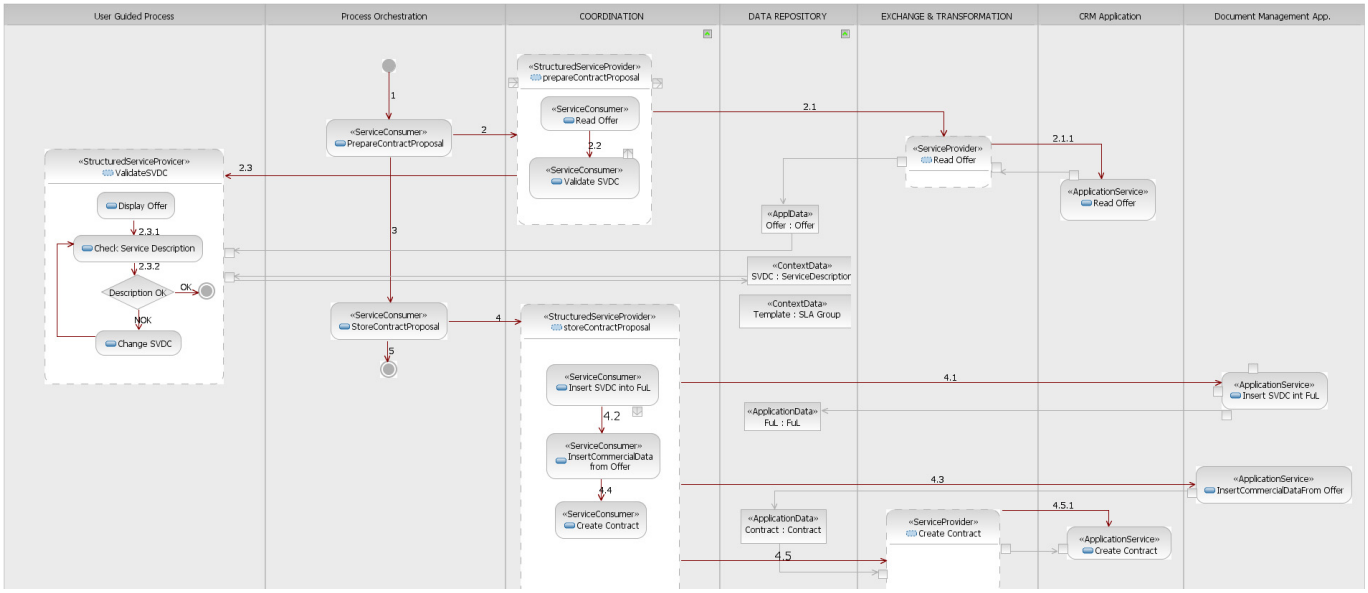

Figure 3. Create Contract Proposal Design

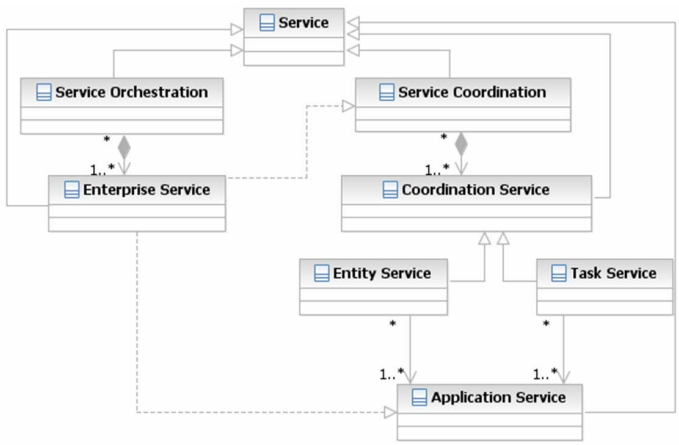

Figure 2. Service Hierarchy Meta-Model

services are not sufficient, though. Services also need to be employable in any arbitrary context. Thus, the services need to be loosely coupled. One aspect of that design principle is that services do not need to be aware of each other. One way to partly achieve this goal is to set-up service registries for a brokered communication among services. Services are completely unaware of other services whenever their dependence to these services is managed by a dedicated instance. This is the reason why the service coordination aggregates the single services without passing the control to other services.

Following the principles descibed so far, the platform-independent design of the entity services is as exemplified in Figure 3. The control-flow is numbered and marked in red. In the design shown, the CRM application is used as application service provider for the entity-specific coordination services *Read Offer* and *Store Contract*. A newly developed application for document management is required in order to support the task-specific services that manage the data manipulation (*Insert SVDC into FuL*[4] and *Insert Commercial Data from Offer*).

The next step of the service design is to categorize the identified task-services and the activity that is required to be performed in order to support the actual enterprise service. The modeled activities pose initial candidates for the services' operations. According to the underlying task or entity, the service is then completed by adding additional operations and eventually modifying the operation candidates extracted from the enterprise service model. The coordination services that can be identified to be supported by the CRM application system are shown in Figure 4. The operation candidates *Read Offer* and *Create Contract* were modified to the operations *callUpOffer* and *bookContract*, respectively.

In order to finish the design of the coordination services, the data model needs to be analyzed. This analysis reveals the data that are required to perform the operations of the services. Two messages are assigned to each operation - an input and an output message. In the presented use-case, the messages for the operations of the CRM-related services are exemplified by showing the message structure for the *bookContract* operation of the *ContractService* entity-specific service. As shown in Figure 5, the attributes of the service-inbound messages consist either of the necessary attributes to create a contract or a contract object itself.

Knowing the interfaces of the coordination services, the final step of the PIM is finishing the interface description of the enterprise services themselves. Since the data manipulation is performed by the included coordination services and the data is kept within the data repository, the enterprise service does not require input or output parameters at all. The business functions are clustered with regards to their business semantics into services while the single business tasks be-

[4]"FuL" stands for "Funktions- und Leistungsbeschreibung" and is a fixed term used in the company for "service description".

Figure 4. CRM Coordination Services

Figure 5. *bookContract* **Operation Messages**

come the operations of these services. All subsequent phases of the methodology depend on actual platform-dependent constraints (e.g how application services could be realized - both technically as well as functional). Hence, the later phases are considered designing the PSM which is out of the scope of this paper (cf. [8, 9]).

4. Conclusion and Future Work

We have presented the first application of our architectural concepts to a real-life business process. The case study shows that the concepts are applicable in real-life. Both, the methodology and the architecture are helpful when it comes to implementing the business process. In addition to it's guidance during the design phase, the concepts showed that they quite naturally lead to a distinct description of coordination services. This distinction is both, in line with accepted best-practices for application design as well as with more recent findings regarding service-design [6]. Currently we are involved with assigning the identified coordination services to the identified backend systems. So far it showed that only little functionality is remotely accessible yet but implementing it in a remote-accessible fashion seems achievable and efficient. The inherent heterogeneity issues identified up to now can be handled by the exchange and transformation layer. In parallel we are also busy to start the detailed implementation of the use-case. On the conceptual side we are validating the pattern language described in [8] in order to check for formal support of our approach. Formalizing the findings, we are currently working on a design metrics that will help to identify business cases that are suitable for the service-oriented paradigm and that will support the design decisions even more than the methodology, patterns and architecture presented here accomplish already.

References

[1] British Government Office of Government Commerce: ITIL IT-Infrastructure Library.

[2] *OASIS WSBPEL Technical Committee: Web Services Business Process Execution Language 2.0*, 31.02.2007.

[3] J. Beatty, H. Blohm, C. Boutard, S. Brodsky, M. Carey, and J.-J. Dubray. Service data objects for java specification. Technical report, BEA and SAP and IBM, 2005.

[4] K. P. Birman. Like it or not, web services are distributed objects. *Commun. ACM*, 47(12):60–62, 2004.

[5] J. Dehnert and W. M. P. van der Aalst. Bridging the gap between business models and workflow specifications. *Int. J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.

[6] T. Erl. *Servcie-Oriented Architecture*. Prentice Hall, Inc., Upper Saddle River, NJ USA, 2005.

[7] D. S. Frankl. *Enterprise Patterns and MDA*. Prentice Hall, Inc., Addison-Wesley, 2004.

[8] H. Hofmeister and G. Wirtz. Approaching a methodology for designing composite applications integrating legacy applications using an architectural framework. In *Proc. GI-FG Treffen EMISA, Hamburg, LNI, Vol. 95, Springer*, 2006.

[9] H. Hofmeister and G. Wirtz. A Pattern Taxonomy for Business Process Integration Oriented Application Integration. In *Proc. 18th Intern. Conf. on Software Engineering and Knowledge Engineering, San Francisco Bay, USA*, 2006.

[10] H. Hofmeister and G. Wirtz. Using patterns to design composite applications. In *Intern. Conf. on Enterprise Information Systems and Web Technologies 2007, Orlando, FL; USA*, 2007.

[11] H. Reijers. A cohesion metric for the definition of activities in a workflow process. In *CaiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Desing (EMMSAD, 03). Velden, Austria.*, 2003.

[12] N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow data patterns: Identification, representation and tool support. pages 353–368, 2005.

[13] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[14] W. Vogels. Web services are not distributed objects. *IEEE Internet Computing*, 7:59–66, November/December 2003.