A Model Driven Approach to Design Web Services in a Web Engineering Method[1]

Marta Ruiz, Pedro Valderas, Victoria Torres, Vicente Pelechano

[1] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera s/n, Valencia-46022, España
{mruiz, pvalderas, vtorres, pele}@dsic.upv.es

Abstract. Probably one of the most difficult tasks in the development of a Service Oriented Architecture (SOA) is how to obtain well designed Web Services. In this work, we present an extension of a web engineering method (OOWS) to provide a methodological guide that allow us to identify well designed Web services in a SOA from the OOWS conceptual model.

1 Introduction

The emerging Web Engineering discipline is being worried on how to develop well designed Web services. A web service should provide public operations with an appropriate granularity level in order to provide flexibility and to facilitate its connection and integration into distribute business processes over Internet.

Some Web Engineering methods (OOHDM [1] and WebML [2]) are trying to introduce web services into their web conceptual modelling approaches. However these approaches do not give support to the design and develop of Web services.

The OOWS [4] approach introduces a model driven approach to develop web application. The OOWS method integrates navigational design with a classical OO conceptual modelling providing systematic code generation (OO-Method [3]). This work is an initial effort to introduce SOA and the Web services technology in the OOWS method. Our proposal defines a methodological guide that allows us to identify a set of functional groups that define public and coarse grained services in a multi-tier SOA. This is done by taking the conceptual models that the OOWS method provides and providing a model driven strategy to systematically obtain the functional groups (web services) by extracting all the useful knowledge that the models include.

This paper is organized as follows: section 2 presents a methodological guide to obtain well designed Web services in a SOA. Finally, we present some conclusions in section 3.

[1] This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

2 A Model Driven Approach to Design Web Services.

In this section, we introduce a methodological guide to identify a set of functional groups that define public and coarse grained services in a multi-tier SOA. A coarse grained service is defined as a Web service that may perform a great amount of operations.

These functional groups are identified from the models that the OOWS method provides. In this sense, to easily understand the presented guide a (necessary) brief overview of the OOWS method is first presented. Moreover, a muti-tier architectural style for SOA is also introduced.

2.1 The source Models. The OOWS approach

In this section we present a brief overview of the OOWS method [4]. The OOWS development process is divided in three major steps: user identification, task description and conceptual modelling.

In the **user identification** step, a *User Diagram* is defined to express which kind of users can interact with the system. Fig. 1A shows the User Diagram of an application like the Amazon Web site where products can be electronically purchased. We have defined users of three kinds: *Visitor*, *Client* and *Administrator*.

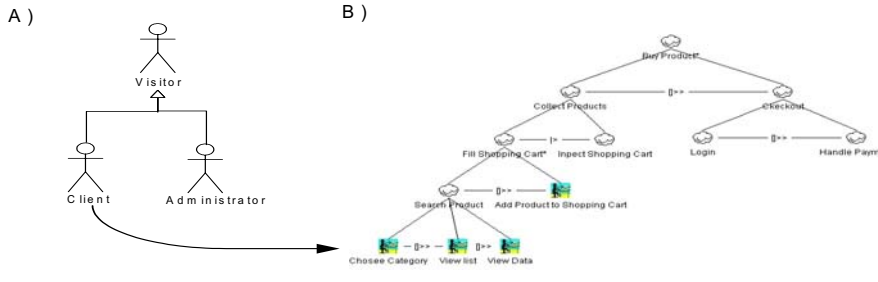

Fig. 1. User and Task diagrams.

In the **task description** step, a *Task Diagram* is defined for each kind of user. In this diagram, we describe which tasks the user can achieve by interacting with the Web application. To define the task diagram we use the CTT (ConcurTaskTree) approach [5]. Fig. 1B describes how a *Client* can buy a product.

In the **conceptual modelling** step, we define a web conceptual schema that gives support to the tasks identified above. The OOWS conceptual schema is made up of several models which describe the different aspects of a Web application. The system static structure and the system behaviour are described in three models (*class diagram* and *dynamic*-and *functional* models) inherited from an object oriented software production method called OO-Method [3]. The navigational aspects of a Web application are described in a *navigational model* [4].

In the OOWS navigational model, for each kind of user we define a directed graph (which defines its allowed navigational structure) whose nodes are *navigational contexts* and its arcs denote *navigational links* (see Fig. 2A). A navigational context (rep-

resented by an UML package stereotyped with the *«context»* keyword) defines a view on the class diagram (see Fig. 2B) that allows us to specify an information recovery. There are links of three kinds (see Fig. 2A): (1) *Sequence links* (represented by solid arrows) that represent a reachability relationship between two contexts; (2) *Exploration links* (represented by dashed arrows) that are defined from the root of the navigational map (depicted as a user) and ends in a navigational context. This kind of link can be activated from any context of the navigational map providing access to the context where the link ends; and (3) *Service link*s (see Fig. 2B) that represent the target navigational context that the user will reach after that service execution.

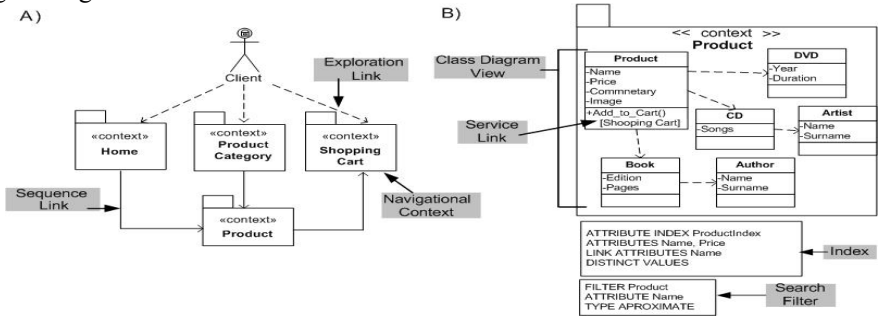

Fig. 2. The OOWS Navigational Model

Furthermore, for each context, we can also define (see Fig. 2B): (1) *Search filters* that allow us to filter the space of objects that retrieve the navigational context and (2) *Indexes* that provide an indexed access to the population of objects. Indexes create a list of summarized information allowing the user to choose one item (instance) from the list. Fig. 2 shows a partial view of the *Client* navigational model that we obtain from the tasks that this kind of user can achieve.

2.2 Architecting Web Service Applications.

The approach introduced in this paper is based on a multi-tier architectural style for SOA that is shown in Fig. 3.

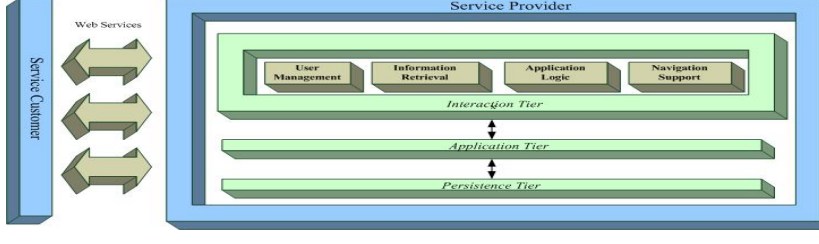

Fig. 3. Multi-tier architecture

In the Service Provider tier, we have three subtiers: *Persistence Tier*, *Application Tier* and *Interaction Tier*. In this work, we focus on designing the *Interaction Tier* (also called *Façade)* of the service provider which is the public part (at operations level) of

the application. This *Interaction Tier* is made up of four functional groups: *User Management*, *Information Retreival*, *Aplication Logic* and *Navigation Support*. Next section introduces a methodological guide to obtain well designed Web services for each functional group, with their operations.

2.3 A Methodological Guide for Designing Coarse Grained Web Services.

In this section we show how a set of functional groups that structure the interaction tier can be identified form the models of the OOWS method. These functional groups are the Web services published in the application and their operations are going to be designed in an appropriate way. These operations are represented in a hierarchical structure: the root node represents the Web service, the first level of nodes represents the public operations offered by this service and the second level of nodes represents the private operations that can perform a public operation.

To easily understand this approach, we follow the example of an application like the Amazon Web site (thereafter known as "Amazon Example") where products can be electronically purchased (see section 2.1).
Next, each service that defines a functional group of the interaction tier is identified.

User Management Service
This service provides the operations for the authentication, authorization and management of the potential users that interact with the application. It is composed by two kinds of operations: (1) those that provide support for the user identification (*loginUser*, *logoutUser*, *changeRol*) and (2) those that give support to generic user administration (*newUser*, *remindPassword*, *modifyUser*, *deletelUser*).

Application Logic Service
This functional group provides operations to implement functional requirements of the application. The operations that constitute this service are obtained from the OO-Method and OOWS models (see Fig. 4): (1) the *task diagram* is used to obtain which are the public and coarse grained operations and their composites; (2) the *class diagram* is used to obtain the parameters of each operation.

Fig. 4 shows the task and the class diagrams from this application. From the task diagram we can obtain two public and coarse grained operations: `collectProducts` and `CheckOut`. The `collectProducts` operation is decomposed into two private operations: `fillShoppingCart` and `inspectShoppingCart`. To obtain their parameters, we should see at the class diagram. For the `collectProducts` we need to detect which are the necessary attributes for each composed operation. To do this, we match the composed operation with operations of the class diagram. Then, the composition of all attributes will be the parameters of the `collectProducts`.

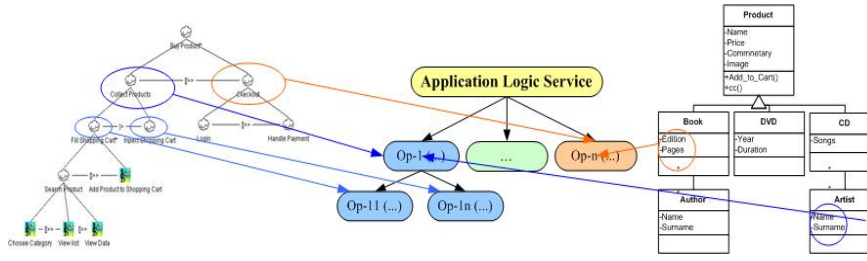

Fig. 4. Application Logic Operations

Information Retrieval Service

This service defines operations to retrieve information about the classes shown in the OOWS contexts. Three operations are obtained from the *navigational context* specification shown in Fig. 5: (1) The `RetrieveInfo` that is defined to obtain information about several classes, (2) the `Index` operation that is defined from the *index* mechanism and (3) the `Filter` operation that is defined from the `Filter` mechanism. These three operations are public and coarse grained because they encapsulate a lot of functionality.

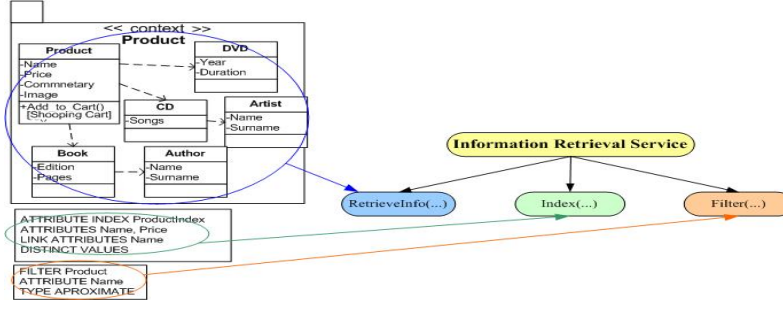

Fig. 5. Information Retrieval Service Operations

Navigation Support Service

This service offers operations to implement the navigation support defined in the navigational maps. The service moves the navigational logic to the *interaction tier*, making easy the implementation of personalization mechanisms and web applications adaptation.

This service has three operations (see Fig. 6). The first two ones are obtained from the *Navigational map* and the last one is obtained from *Navigational Contexts*. The `explorationLink(id_sesion)` operation is obtained from the *Exploration navigational contexts* and the `sequenceLink(id_sesion, contex)` operation is obtained from the *Sequence navigational contexts*. The `ServiceLink(id_sesion, service)` operation is obtained to support the *Service links*.

The three operations are public and fine grained because they provide simple functionality that is clear defined and only performs one operation.

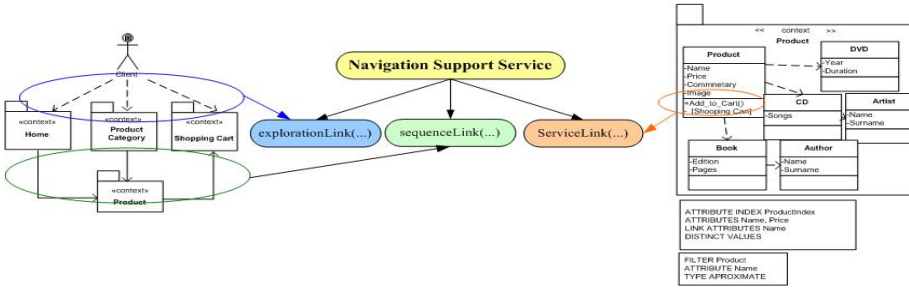

Fig. 6. Navigation Support Operations

3. Conclusions and Future Work

In this work, we have introduced a multi-tier architecture based on levels when we distinguish four functional groups: User Management, Information Retrieval, Application Logic and Navigations Support. Each of these groups is a Web service composed by a set of operations. We have presented a methodological guide to obtain well designed Web services in a SOA.

This methodological can be generalized to other Web Engineering Methods, because the OOWS method share the most common models and primitives which that source information to obtain the web services. As further work, we are defining the transformations using Graph Grammars [6] to automatically generate the Web services from the OOWS models.

References

[1] Schwabe D., Rossi G. and Barbosa D.J. "Systematic Hypermedia Application Design with OOHDM". Proc. ACM Conference on Hypertext. pp.166. 1996.

[2] Ceri S., Fraternali P., Bongio, "A. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites". In WWW9, Vol. 33 (1-6), pp 137-157. Computer Networks, 2000

[3] Pastor, O., Gomez, J., Insfran, E., Pelechano, V. "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming". Information Systems 26, pp 507–534 (2001)

[4] Joan Fons, Vicente Pelechano, Manoli Albert y Oscar Pastor. "Development of Web Applications from Web Enhanced Conceptual Schemas". Springer-Verlag, Lecture Notes in Computer Science. Proc. Of the International Conference on Conceptual Modelling, 22nd Edition, ER'03, pp 232-245. Chicago, EE.UU, 13 - 16 October 2003.

[5] F. Paternò, C. Mancini and S. Meniconi, 1997. "ConcurTaskTrees: a Diagrammatic Notation for Specifying Task Models", In Proceedings of INTERACT'97, Chapman & Hall, 368-366.

[6] Dániel Varró. "Automated Program Generation for and by Model Transformation Systems". In Proc. AGT 2002: Workshop on Applied Graph Transformation, 2002.