Applying a Web Engineering Method to Design Web Services*

Marta Ruiz, Pedro Valderas, and Vicente Pelechano

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia,
Camí de Vera s/n, Valencia-46022, Espana
{mruiz, pvalderas, 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. Some Web Engineering methods provide support to introduce Web services in the software development process but do not give support to the systematic design and implementation of them. In this work, we present an extension of a Web Engineering method (called OOWS) to provide a methodological guide for designing Web Services. This allows identifying and designing the operations and arguments of Web Services following a model-driven approach, taking the OOWS conceptual models as a source. To document our approach, we apply our ideas to the design of the Amazon Web Service and compare our proposal with the solution provided by Amazon.

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 distributed business processes over the Internet.

Some Web Engineering methods are extending their proposals to introduce Web services into their web conceptual modelling approaches (OOHDM [1], WebML [2] and UMLGuide [3]). Those approaches introduce some kind of syntactic mechanisms to include web service calls into the navigational model. However, these approaches do not give support to the design and development of Web services.

The OOWS [4] approach proposes a model driven approach to develop web applications. The OOWS method integrates navigational design with a classical OO conceptual modelling providing systematic code generation (following the strategy proposed in OO-Method [5]). The present work is an initial effort to introduce SOA and the Web services technology in the OOWS method. The main contribution of our proposal compared to other Web Engineering methods is the definition of a methodological guide that allows systematically identifying a set of functional groups that define public operations in a SOA.

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

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 576 – 581, 2005.

The structure of the paper is the following: section 2 presents an overview of the OOWS approach, introducing the steps and the models provided by the development method. Section 3 presents the methodological guide to obtain the operations that constitute the functional groups. Section 4 compares the operations that are obtained following our strategy with those published by Amazon. Finally, we present some conclusions and further work in section 5.

2 The OOWS Approach. An Overview

In this section, we present a brief overview of the OOWS method [4]. In order to build a web application OOWS introduces a development process that is divided into three main stages: *User identification*, *Task description* and *Conceptual modelling*.

In the **user identification** step, a *User Diagram* is defined to express which kind of users (roles) can interact with the system, providing a role-based access control (RBAC [6]).

In the **task description** step, a *Task Diagram* (see Fig. 1-A) is defined for each kind of user. In this diagram, we describe in a hierarchical way which tasks the user can achieve by interacting with the Web application.

In the **conceptual modelling** step, we define a web conceptual schema that gives support to the tasks identified above. The navigational aspects of a Web application are described in a *navigational model* [4].

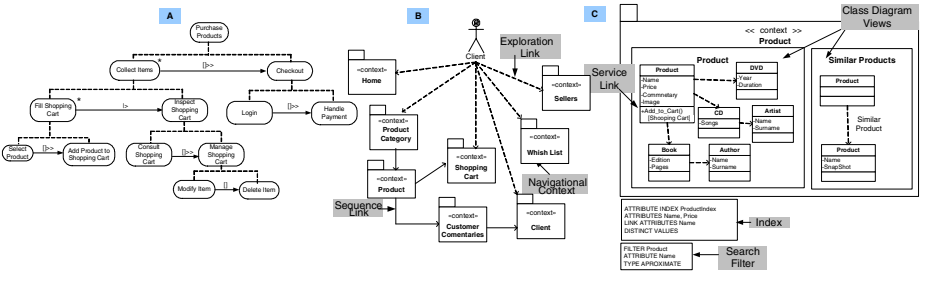

Fig. 1. OOWS models

The OOWS navigational model is defined from a set of navigational maps that describe the navigation allowed for each kind of user (specified in the user diagram). Each navigational map (see Fig. 1-B) is represented by a directed graph whose nodes are *navigational contexts* and its arcs denote *navigational links*.

A navigational context (see Fig. 1-C) (represented by an UML package stereotyped with the *«context»* keyword) defines a view on the class diagram that allows us to specify an information recovery. There are links of three kinds: (1) *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; (2) *Sequence links* (represented by solid arrows) that represent a reachability relationship between two contexts; (3) *Operation link*s that represent the target navigational context that the

user will reach after an operation execution. Furthermore, for each context, we can also define: (1) *Search filters* that allow us to filter the space of objects that retrieve the navigational context. (2) *Indexes* that provide an indexed access to the population of objects.

3 A Methodological Guide for Designing Web Services

In this section we present the main contribution of our proposal: a methodological guide that allows us to obtain the operations that implement the requirements of a Web application in a SOA. These operations are obtained in a systematic way from the OOWS models. Analyzing these models and taking into account the kind of requirements that they capture, our proposal identifies a set of functional groups (fg) that define the public operations in a SOA. We identify four fg: *User Management*, *Information Retrieval*, *Application Logic* and *Navigation Support*. These fg constitute the public interface of the designed Web service.

3.1 User Management Group

The User Management (UM) group provides the operations for the authentication, authorization and management of the potential users that interact with the application. The operations of this group can be detected using the OOWS user diagram. Afterwards, the operations of this service are detected from both the *user diagram* and the RBAC model [6] and are classified into three types: (1) Those that provide support for the user identification: `loginUser`, `logoutUser`, `obtainRol`, `changeRol` and `remindPassword`. (2) Those that give support for the generic user administration: `newUser`, `modifyUser`, `deleteUser`. (3) Those that only can be executed by an Administrator user: `newRol`, `deleteRol`, `addUserToRol`, `removeUserToRol`, `addPermission` and `removePermission` inherited from the RBAC model [6].

3.2 Information Retrieval Group

The Information Retrieval (IR) group defines operations to retrieve the information that must be shown in each navigational context (see Fig. 1-C) (a web page in the running example): (1) The `retrieve`*`ViewName`*`(id_sesion, [attributeID])` operation allows us to obtain the information specified in the navigational context views. The operations detected from the navigational context *Product* (see Fig. 1-C) are: `retrieveProduct` and `retrieveSimilarProducts`. (2) The `getIndexed`*`IndexName`*` (id_sesion, attributes)` operation gives support for the *index* mechanisms defined in a navigational context. The operation `getIndexedProductIndex` is identified from the index of the context *Product*. (3) The `search`*`FilterName`*`(id_sesion, attribute, value)` operation gives support to the filter mechanisms defined in a navigational context. The operation `searchProduct` is detected from the filter defined in the context *Product*.

3.3 Application Logic Group

The Application Logic (AL) group provides operations to implement functional requirements of a Web application.

The operations that constitute this group are obtained from both the *task diagram* and the *class diagram*: (1) The *task diagram* is used to determine the public operations that must be offered. For each leaf task we define an operation. In the task diagram of the Amazon example (see Fig. 1-A), we define the following public operations: *SelectProduct*, *AddProductShoppingCart*, *ConsultShoppingCart*, *Modify-Item*, *DeleteItem* and *HandlePayment*. The *Login* operation is not offered in this group because it is an operation of the *UM* group. (2) The *class diagram* is used to obtain the arguments of each operation. We detect each class that participates in an operation achievement and then its/their attributes define the operation arguments.

3.4 Navigation Support Group

The Navigation Support (NS) group provides operations to implement the navigation defined in the navigational model. The NS moves the navigational logic to the *interaction tier* facilitating both the implementation of adaptation and personalization mechanisms of web applications. This group has three operations: (1) The `explorationLink(id_sesion)` operation gives support to the implementation of the exploration links. (2) The `sequenceLink(id_sesion, context)` gives support to the implementation of the sequence links. (3) The `operationLink-(id_sesion, service)` operation gives support to the implementation of the operation links.

Fig. 2 shows the implementation (Web page) of the *Product* context (see Fig. 1-C). In this figure we can see the use of some operations shown in this work.

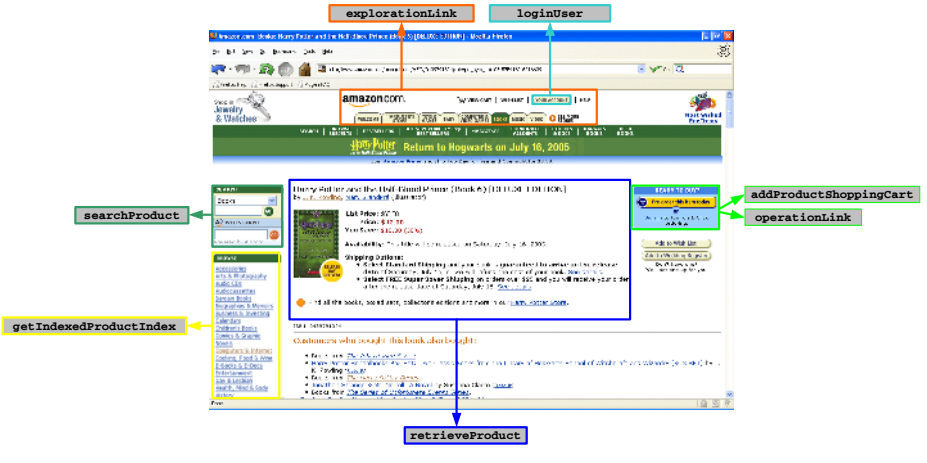

Fig. 2. Web page of the *Product* context

4 Evaluation of Our Proposal

In this section, we compare the operations that are obtained following our strategy to those published in Amazon[1]. Our intention is to identify some weak points of our proposal in order to improve our method.

Amazon web service offers 18 operations while we offer 27 operations (14 from the UM group, 4 from the IR, 6 from the AL and 3 from the NS group). Next, we show the comparative between the Amazon and our operations:

(1) `BrowseNodeLookUp`: this operation is supported by `retrieveCategory` detected from a view defined in the *Product Category* context.
(2) `Help`: it is not supported because we have not captured this requirement in our web conceptual model. This operation just provides a user manual.
(3) `CustomerContentLookup`: we implement it with `retreiveClientCo-mentaries` detected from a view defined in the *Client Commentaries* context.
(4) `CustomerContentSearch`: this operation is supported by `retreiveClient` detected from a view defined in the *Clients* context.
(5) `ItemLookup`: we support it with `retrieveProduct` detected from the *Product* view defined in the *Product* context.
(6) `ItemSearch`: This operation is supported by `searchProduct` detected from the filter defined in the *Product* context.
(7) `SimilarityLookup`: it is implemented by `retrieveSimilarProducts` detected from the *Similar Products* view defined in the *Product* context.
(8) `ListLookup`: this operation is supported by `getIndexedProductIndex` detected from the index defined in the *Product* context
(9) `ListSearch`: it is implemented by `searchWhishList` detected from a filter defined in the *Whish List* context
(10) `CartAdd`: we support it with `addProductShoppingCart` detected from the *task diagram*.
(11) `CartClear`: is not supported because in the Amazon web conceptual model, this functionality has been indirectly modelled through the task *Delete Item*. we have considered that if the user wants to clear the cart he/she must delete all the items.
(12) `CartCreate`: this operation is implicitly implemented in our `addProductShoppingCart` operation.
(13) `CartGet`: we implement it with `consultShoppingCart` detected from the *task diagram*.
(14) `CartModify`: this operation is implemented by two of our operations: `modifyItem` and `deleteItem` (detected from the *task diagram*).
(15) `SellerLookup`: this operation is supported by `retrieveSeller` detected from a view defined in the *Sellers* context.
(16) `SellerListingLookup` and (17) `SellerListingSearch`: these operations are related to the integration of Amazon with Third Party systems which is out of the scope of this work. Information about this can be found in [7].

[1] http://www.amazon.com/gp/browse.html/102-0679965-?%5Fencoding=UTF8&node=3435361

(18) `TransactionLookup`: it is not supported because we have not considered information about financial operation in the web conceptual model specification.

Furthermore, our approach provides additional operations that do not exist in the Amazon web service. These operations are those presented in the UM and NS groups, in addition to `handlePayment` of the AL group.

As a conclusion, we can see that our proposal provides a good enough solution that is closer to the functionality provided by Amazon and it also includes additional functionality that can be used to provide user authentication mechanisms, giving an extra control of the navigation requirements and allows to support adaptation and personalization mechanisms of web applications.

5 Conclusions and Further Work

In this work, we have presented an approach to introduce SOA and the Web services technology in the OOWS method. We have presented a methodological guide to obtain the operations that define the Web service from the OOWS models. This methodological guide can be generalized to other Web Engineering Methods, because the OOWS method shares with them the most common models and primitives taken as source to obtain the Web services.

We are working on providing mechanisms that facilitate the integration of Web applications with Third party systems (tps) at the conceptual level [7]. When tps supply us their functionality as Web services, we apply Web services composition to achieve integration.

References

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

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

[3] P. Dolog, "Model-Driven Navigation Design for Semantic Web Applications with the UML-Guide". In Maristella Matera and Sara Comai (eds.), Engineering Advanced Web Applications. 2004

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

[6] ANSI. Incits 359 2004. American National Standard for Information technology. Role-Based Access Control, 2004.

[7] V. Torres, V. Pelechano, M. Ruiz, P. Valderas, "A Model Driven Approach for the Integration of External Functionality in Web Applications. The Travel Agency System". In Workshop on Model-driven Web Engineering (MDWE 2005) at ICWE July 2005, Sydney, Australia. Accepted for publication.