A Model Driven Approach for the Integration of External Functionality in Web Applications. The Travel Agency System

Victoria Torres, Vicente Pelechano, Marta Ruiz, Pedro Valderas
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera s/n, Valencia-46022, España
{vtorres, pele, mruiz, pvalderas }@dsic.upv.es

Abstract

Nowadays, it is getting more and more common to develop Web applications where part of the functionality is carried out by different systems. These systems provide functionality developed in different technologies that are integrated to build a complete web application. To deal with the integration issue that allows us to build this kind of Web applications, current Web Engineering methods should provide the mechanisms that facilitate this integration with third business parties during the modeling process. This paper presents a model driven method to achieve integration with external parties at a high level of abstraction. The method provided is an extension to the OOWS approach for the construction of this new kind of Web applications. The Travel Agency System has been taken as a case study to clearly understand how the whole method is applied.

1. Introduction

Web applications cannot be longer conceived as isolated applications. Moreover, the different possibilities in which business partners can provide their functionality (CORBA, J2EE or .NET) motivate us to propose a method that helps in the construction of more opened and collaborative Web Applications that integrate functionality from different sources.

There are several ways in which web applications can be built integrating functionality provided by external parties. For instance, a web application could require a concrete external functionality to accomplish a specific functional requirement or to provide some information that complements the data handled by our system. A more complex way could be when business process supported by the web application makes use of activities implemented by external business partners.

Web engineering methods are extending their solutions to provide support and/or integrate functionality and business processes into web conceptual models. In this context, we can distinguish approaches that deal with business process modelling and integration into navigational models like OOHDM [1], WSDM [2] or UWE [3] (that introduce process definitions into navigational models, causing a semantic overload of the navigational nodes because activities and processes are living together with nodes and links), other methods like WebML [5] and UML-Guide [6] model business processes as some kind of navigation; UML-Guide is based on the semantic web technology (OWL) to specify state machines that are used to express navigation and web service operation calls. Both approaches introduce some kind of syntactic mechanisms to include web service calls into the navigational model. Finally, OO-H [4] and WIED [7] (in the form of a companion notation to the WebML), model business processes and navigational models as separate concerns and notations. Only WebML and UML-Guide are worried about how to support integration of external functionality.

Our proposal introduces some contributions in this context because we think that the integration of business process in web application modelling should follow a concern oriented approach preserving the role and the notation of current business process modelling techniques (for instance, UML activity diagrams) and navigational modelling techniques as OO-H and WIED states, but also focusing on solving the integration to external parties problem in a model driven fashion (following the MDA principles). We also want to emphasize that we provide a methodological guide that helps web designers in the construction process of this new kind of web applications. Moreover, we think that Web Engineering methods also should face up the integration problem from two different points of view, which are the consumer and the provider perspective.

On one hand, as consumers we need mechanisms that help us to model Web Applications that use external artifacts. On the other hand, as producers we need mechanisms for generating artifacts that could be exported and used by other applications.

In this context it is necessary to provide a methodological guide that helps web developers in the construction process of this new kind of web applications. We think that the integration issue should be tackled following a model driven approach.

The rest of the paper is structured as follows. Section 2 provides an overview of the method, explaining the set of models use it and the existing dependencies between them. Moreover, we state how this proposal fits into the MDA approach. Section 3 presents briefly the Travel Agency System (TAS) case study. In the following sections, from 4 to 6 we present the method, by means of the TAS case study. Section 7 shows the strategy followed to build Web Services that make accessible the Broker Agent implemented in the TAS. In section 8 we outline how we generate the interaction with external functionality provided in different technologies. In section 9 we show the user interfaces generated from the specifications made in the Navigational Model. Finally, section 10 draws some conclusions and outlines further work.

2. An Overview of the Model-Driven Method

This method provides an extension to the OOWS [12] approach. This extension introduces the required expressivity to capture the integration requirements that are necessary to build distributed web applications.

Following the MDA guidelines, this method has been organized in three views: the Computation Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM). Our proposal, as can be seen in figure 1, introduces new models for: requirements elicitation (Task Definitions) and supporting integration (the Services and the Business Process Models). To support integration when modeling navigation we extend the already defined Navigational Model and reuse those existing models like the Dynamic, Structural, Functional and Presentation.

Due to the fact that our proposal is based on an existing method that provides a code generation tool (OlivaNova CASE tool[1]), in this extended version, we want to provide a solution for (1) producing functionality for third party consumption and (2) integrating functionality supplied by external providers. In particular, we focus on the integration of external services at the Business Process and Navigational level.

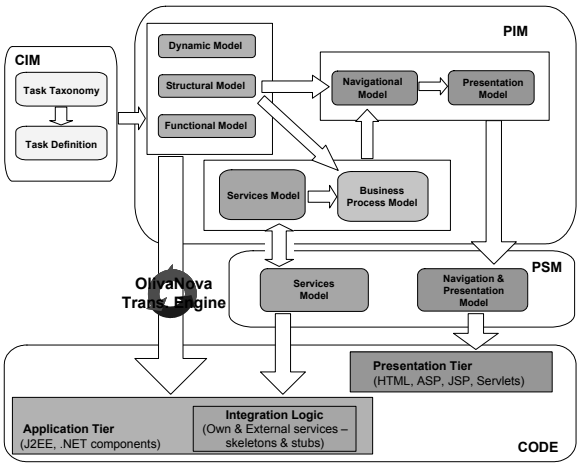

Fig. 1 Method Overview

In Fig. 1 we can see how the models proposed are organized in each different level (CIM, PIM, PSM and code).

3. Applying the Method to the TAS Case Study

The TAS is a Web Application that sells electronically travels to its customers. In particular, we only concentrate on providing transportation services (plains, trains, cars, boats or combinations of those) for a trip. This service can be either provided by external Broker Agents (implemented by other Travel Agencies) that work in conjunction with our TAS or implemented in our system. These Broker Agents use the services provided by Transportation Companies to supply an offer that matches with the customer requirements. In case a trip cannot be supplied by any Transportation Company, it is the Broker Agent which has to split the trip and try to compose the complete trip from split services. Once the customer has selected an offer that matches with his/her requirements, the TAS uses the services provided by the corresponding Financial Company to proceed with the payment of the selected offer. Finally, once a month, the TAS pays external Broker Agents for the services they have provided during the previous month.

In the following sections we are going to apply the TAS case study to the method roughly presented in the previous section. Moreover, we will also include how

[1] http://www.care-t.com

we build the Broker Agent business logic and the strategy followed to construct the Web Services that provide the functionality implemented by our Broker Agent.

4. Defining the CIM

The Computation Independent Model (CIM) proposed by MDA is built mainly to bridge the existing gap between those that are experts about the domain and those that are experts on how to build the artifacts that satisfy the requirements domain [12]. Then, according to MDA, a CIM must describe the requirements of the system.

We specify the early requirements of a Web application by means of a task model. This model is built from the tasks that users must be able to achieve when interacting with the application as well as from the tasks that the system must perform. The requirements specified when building the task model are used in following stages for systematically generating part of the PIM. This approach allows us to provide a higher degree of traceability than those requirement specification methods which transform models manually.

We propose two steps to define the task model:

(1) *Task identification*: we identify the set of tasks that the system together an actor must achieve to accomplish each requirement. An actor represents a user or any other system that interacts with the system under development [10]. The set of identified tasks are organized in a task taxonomy.

(2) *Task description*. To accomplish the goal defined by each leaf task included in the task taxonomy, we describe the set of actions that must be performed to succeed in achieving that goal. This description is made by using the UML activity diagrams [10].

4.1 Task Identification

To identify the set of tasks that represent the web application requirements we must detect, as a first step, which actors can interact with the system. Then, for each detected actor we must define a task taxonomy that represents the tasks that this actor can achieve when interacting with the system.

In the TAS example, we only detect an actor: the internet user. The task taxonomy associated to this actor is shown in Fig. 2. For the construction of the task taxonomy, we take as the starting point, a statement of purpose that describes the main goal of the web application. The statement of purpose is considered the most general task of the system. From this task, a progressive refinement is performed, obtaining as a result more specific tasks. Tasks are decomposed into subtasks by following structural or temporal refinements. The *Structural* refinement (represented by solid lines in Fig. 2) decomposes complex tasks into simpler subtasks. The *Temporal* refinement (represented by dashed lines in Fig. 2) provides order constraints for the children tasks according to the task logic. To define these temporal constraints we propose the use of the temporal relationships introduced by the ConcurTaskTree approach (CTT) [11]. In Fig. 2 we can see the temporal relationship *Enabling* (>>) which represents that after being finished the first task the second is activated. The rest of temporal relationships proposed by the CTT approach are not explained in this work due to space constraints.

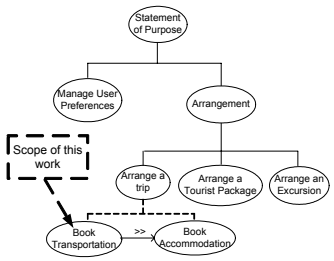

Fig. 2 A Task Taxonomy of the TAS

The statement of purpose of this system is decomposed into two tasks: *Manage User Preferences* and *Arrangement*. At the same time, the task Arrangement is divided into three tasks: *Arrange a Trip*, *Arrange a Tourist Package* and *Arrange an Excursion*. Finally, to arrange a trip the user must first Book Transportation and then (>> Enabling relationship) Book Accommodation. On the other hand, regarding to the Broker Agent (BA), Fig. 3 depicts the task taxonomy that represents the requirements of this system. In this case, there is only an actor that can interact with the BA system, which is the TAS system.

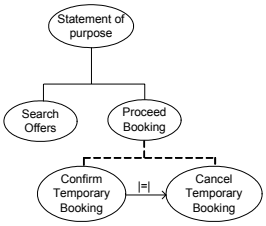

Fig. 3 A Task Taxonomy of the Broker Agent

In the following section we introduce a strategy to describe each identified tasks. To better understand this strategy we show the description of one task of each presented task taxonomy: Book Reservation (from the TAS) and Search Offers (from the BA).

4.2 Task Description

In order to describe the set of tasks detected in the previous stage we extend the traditional descriptions which specify the system actions that are needed to achieve each task. We introduce information about the *interaction between actors and the system*, indicating explicitly when (at which exact moment) it is performed. To do this, we introduce the concept of interaction point (IP). An IP can define two different types of interaction:

(1) *Output Interaction*: the system provides actors with information and/or access to operations which are related to an entity[2]. Actors can perform several actions with both data and operations: they can select information (as a result the system provides them with new information) or activate an operation (as a result the system carries out with an action).

(2) *Input Interaction*: the system is waiting for the user to introduce some data about an entity. The system uses this information to correctly perform a specific action (for instance, to carry out with an on-line purchase with the provided data client). In this case, the only action that the user must perform is to introduce the required data.

In order to perform descriptions based on IPs we propose the use of UML Activity Diagrams [10] as can be seen in Fig. 4 where:

- Each node (activity) represents an IP (solid line) or a system action (dashed line). In addition, IPs are stereotyped with the *Output* or the *Input* keyword to indicate the interaction type.
- In the *Output IPs,* the number of information instances[3] that the IP includes (cardinality) is depicted as a small circle in the top right side of the primitive.
- As far as the *Input IPs,* we said that the data introduced by the user is taken by the system to correctly perform a specific action. To capture that this kind of IPs exclusively depends on a system action (it does not take part in the general process of the task), nodes that represent both elements (input IP and system action) are encapsulated into dashed squares.

[2] Any object of the real world that belongs to the system domain (e.g. customer, product, invoice, etc)

[3] Given a system entity (e.g. customer), an information instance is considered to be the set of data related to each element of this entity (Name: Joseph, Surname: Elmer, Telephone Number: 9658789).

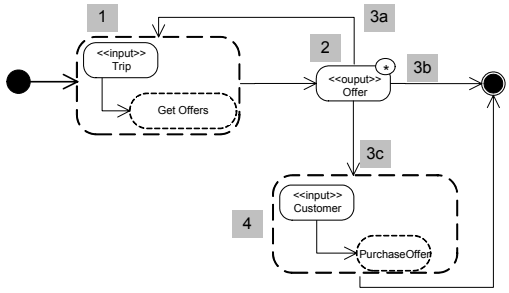

Fig. 4 Book Transportation Business Process

Fig. 4 shows the description of the task *Book Transportation***.** This task begins with the system action *Get Offers* (1). This action temporally books the offers that match with the trip description introduced by the actor (in this case, the internet user). This description is introduced by means of the Input IP defined previously to the system action. Once this action is finished, the task continues with an Output IP, where the system provides the internet user with the list of matched offers (2). From this IP the internet user can either start the process again to refine his/her trip description (3a), reject all the supplied offers and quit (3b) or select one offer (3c). When the internet user selects an offer the system performs the action *Purchase Offer* (4). To perform this action the user must introduce its client information by means of an Input IP. Then the task finishes. Fig. 5 shows the description of the *Search Offers* task. This task starts with the system action *Search Transportation*. This action searches transportation that matches with the trip information provided by the actor (the TAS system) through an Input IP. Next, the system (the Broker Agent) temporally books the transportation to finally conclude the task.

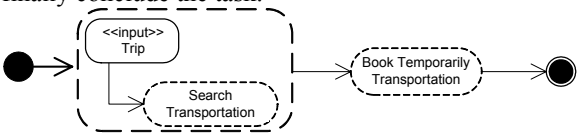

Fig. 5 Search Offers Business Process

In order to make task descriptions easy, details about the information exchanged (in each IP) between the user and the system are not described (we just indicate the entity which the information is related to). This information is specified by means of a technique based on information templates that is next introduced.

4.2.1 Describing the system data

The information that might be stored in the system is represented by means of a template technique that is

based on data techniques such as the CRC Card [12]. We propose the definition of an information template (see Fig. 6) for each entity identified in the description of a task. In each template we indicate an identifier, the entity name and a *specific data* section. In this section, we describe the information in detail by means of a list of specific properties associated to the entity. For each property we provide a name, a description and a data type. In addition, we use these templates to indicate the information shown in each IP. For each property we indicate the IPs where it is shown (if there is any). To identify an IP we use the next notation: *Output (Entity, Cardinality)* for Output IPs and *Input (Entity, System Action)* for Inputs IPs.

Identifier:	T1			
Entity:	Customer			
Specific Data:	*Name*	*Description*	*Type*	*IPs*
	Name	Name of the Customer	String	
	Address	Postal Address	String	
	Email	Address of the E-mail	String	
	Birthday Date	Date of the Customer's Birthday	Date	
	Credit Card	Number of the Customer's Credit Card	String	Input(Trip,Pre-Book Offers)
	Phone Number	CD cover frontal	Number	

Fig. 6 Information Template for the Customer entity

According to the template showed in Fig. 6, the information that the system must store about a Customer is (see the specific data section): his/her name, address, email, birthday date, credit card number (which is requested in the IP *Input(Trip, Pre-Book Offers)* and the phone number.

5. Building PIM models from the CIM

Five models should be specified in order to describe the web application at the OOWS PIM level:

(1) the structural view of the system
(2) the external functionality that our system will consume
(3) the business process of the application
(4) the navigational view of the system
(5) the presentation view of the system.

Although there is not an explicit order in which these models might be built, the existing dependencies between them introduce some constraints about the sequence in which some of these views should be built.

Model transformation is applied at this point. From the requirements gathered previously, we proceed to transform them into more detailed models of the web app.

We want to note that CIM models did not specify what part of the system was going to be provided by an external business partner, if any. Nevertheless, we should state at this stage (PIM modeling) which functionality is going to be provided by external parties.

In the following subsections we proceed to model the structure, external functionality, business processes, navigation and presentation views of the TAS case study.

5.1 Structural Modeling

The Structural Model specifies by means of a *UML Class Diagram* the system structure (its classes, operations and attributes) and relationships between classes (specialization, association and aggregation).

This model can be partly obtained (just classes and attributes) from the *Templates* built at the requirements step (see section 4.2.1). As Fig. 7 shows, we have obtained for the TAS case study four classes (Customer, Trip, Broker Agent and Line Trip) that describe the part of the domain that needs to be fully managed by our system.

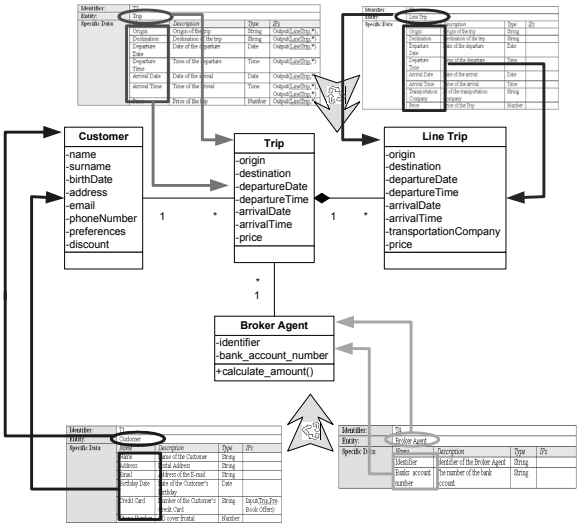

Fig. 7 TAS Structural Model

For instance, the TAS requires keeping customer data (such as his/her name, surname, preferences for travel searches and the kind of discount associated to him/her) among others.

5.2 Services Modeling

To build Web applications that make use of external artifacts (such as components, class libraries, Web services, etc.), we need to represent them at the modeling level. Following this approach we can work with external functionality as if they were native elements of our system. Therefore, we model those external artifacts that are going to interact with our

system in a specific model called the *Services Model*. This model has been conceived to specify in a technology independent fashion the external providers based on the functionality that they supply. This specification helps us to easily handle external functionality as if they were part of the native system at high level of abstraction.

In this model we define the services supplied by external providers as well as the set of operations (their interfaces) that they offer. This definition allows us to have a generic description of functionality that is provided at the same time by different partners and in different technologies. There are two main benefits of having a generic representation of external functionality. The former is that the modeling process gets easier because we work with a generic specification (note that we define adaptors to match real external operations with those generic modeled in this model). The latter is that adding or/and removing providers do not have a collateral effect in the rest of models that depend on the services model.

As can be seen in Fig. 8 we have modeled each external system type (the Broker Agent, The Financial Company and the Transportation Company) with the set of operations that they provide.

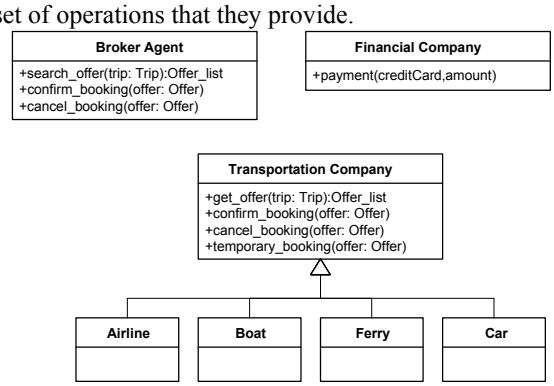

Fig. 8 TAS Services Model

5.3 Business Process Modeling

The Business Process Model defines the set of business processes (BP) that characterize the business of the application. The BPs that are defined in this model correspond to processes that describe the flow and operations that made up the system actions (nodes in the task description) detected at the CIM level. In the case of distributed web systems, as it is the case with the TAS, these BPs can be formed not only of activities performed by our system but also from activities carried out by external partners. Therefore, these processes can be made up by internal (implemented by our system) or/and external activities (provided by business partners). To differentiate in this diagram the external activities we mark them with the external stereotype.

For the TAS we have specified two task taxonomies, one to specify the interaction between the user and the TAS system (see Fig. 2) and another to specify the tasks that should perform the Broker Agent that implements our system.

Associated to the first task taxonomy, and taking the task description specified for the *Book Transportation* leaf in the task tree, we proceed to specify/refine the description of the process that defines the system actions (*Get Offers* and *Purchase Offer*) included in the Book Transportation Business Process.

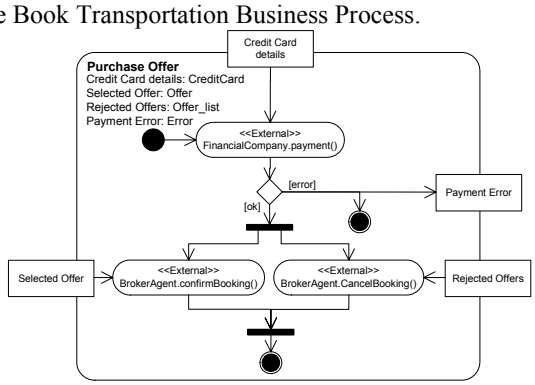

Fig. 9 Purchase Offer Process

In particular, Fig. 9 depicts the definition of the process for the Purchase Order system action. This process accepts as input the credit card details, the offer to be purchased and the list with the rejected offers. With this information it will proceed with the payment to the corresponding Financial Company. If this activity is correctly performed, the process continues by confirming the selected offer and canceling the temporary booking of discarded offers.

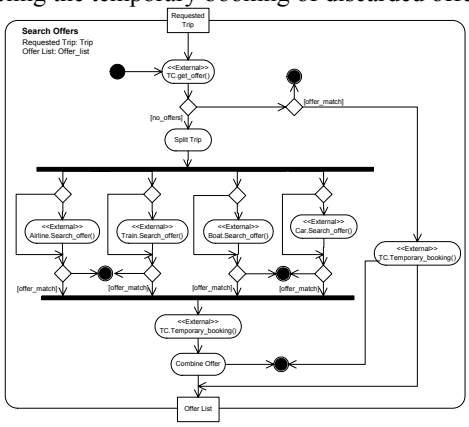

Fig. 10 Search Offers Process

For the Broker Agent task taxonomy we have defined two tasks, which are *Search Offer* and *Proceed Booking*. Fig. 10 depicts the activity diagram that defines the *Search Offer* process. This process accepts

as input a requested trip (including preferences and constraints) and manages to get a set of offers that match with the specified trip.

Fig. 11 defines the process in charge of paying external Broker Agents for the services supplied to our TAS. In this figure, the once-a-month accept time event action generates an output (signal) once a month that is received by the Pay Broker Agent process. At this moment, the process is performed.

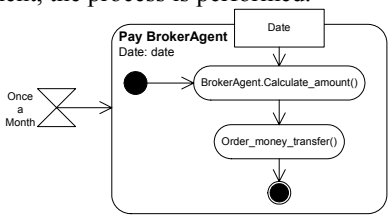

Fig. 11 Pay Broker Agent Process

5.4 Navigational Modeling

Until this point we have already specified the back-end of the system (structure, application logic and external functionality). Nevertheless, as our goal is to build Web applications, it is necessary to specify the front-end of the system. This is done through the Navigational Model.

In the Navigational view we build two models:

(1) the **user diagram**, This model expresses what kind of users can interact with the web application and the system visibility that they should have.

(2) the **navigational model**. This model defines the system visibility for each kind of user in terms of navigational constructs.

5.4.1 User Diagram

To define what kind of users can interact in the system we build the *User Diagram*. This diagram provides mechanisms to properly cope with additional user management capabilities, such as the user specialization that allows defining user taxonomies to improve navigational specification reuse.

As can be seen in Fig. 12, for the TAS case study we have specified two user types, an anonymous user (*Anonymous user*) that do not need to provide information about his/her identity to the system (depicted with a question mark symbol) and a registered user (*Registered customer*), who needs to be identified to connect the system (depicted with a padlock).

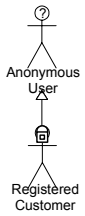

Fig. 12 TAS User Diagram

Once users have been identified, a structured and organized system view for each type must be specified. These specifications are shown next in the Navigational Model.

5.4.2 Navigational Model

The Navigational requirements of the system are defined in the *Navigational model.* In this model we provide a structured and organized view of the system for each user type defined previously in the User Diagram. Navigation requirements are captured in two steps: the "Authoring-in-the-large (global view) and the "Authoring-in-the-small" (detailed view).

For the definition of the navigational global view we take as reference the task taxonomy specified in the requirements modeling step (see Fig. 2). Only that leafs from the task tree whose associated interaction-actor is the user are transformed into Navigational Contexts4 in the Navigational map. In particular, those leafs targeted with *Enabling* relationships are defined in the Navigational Map as *Sequence Contexts* (contexts that can only be accessed via a predefined navigational path by selecting a sequence link). The rest of navigational contexts are defined as *Exploration Contexts*. Exploration contexts are accessible from any node of the application. Fig. 13 shows the global view of the system for the registered customer user type.

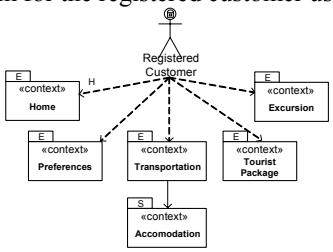

Fig. 13 TAS Navigational Map

Once the global view has been defined we should provide a navigational description for each navigational node (detailed view). Each navigational context is defined as a view over one of the three

[4] Navigational Contexts represent user interaction units that provide a set of cohesive data and operations to perform certain activity.

models presented in the previous sections: *the class diagram, the services model and the business logic model.* On one hand, a view over the class diagram (class view) is defined in terms of the visibility of class attributes, operations and relationships (class views defined by OOWS). On the other hand, when these views are built from operations (defined in the services model) or processes (defined in the business logic model) they are defined in terms of the data returned by these operations/processes (we call it Functional Views). These Functional Views are organized in three sections as follows:

(1) One section to specify the operation/process name, including its input and output parameters.
(2) A second section to specify which data from the data returned by the operation/process is going to be shown in the context.
(3) Finally, in a last section we include the Operations/Processes that can be performed with the data contained in the context.

An example of a view defined over the business process model is depicted in Fig. 14. The way in which a Functional View will be provided graphically is explained next:

(1) If the operation that defines the Functional view requires a set of input parameters these will be asked to the user by means of a input form. If not, the operation is directly executed without providing any data. In Fig. 14 the user should provide data about the trip as well as some constraints and preferences related to it.
(2) As a result of this invocation, this context would filter the data returned by the operation, showing only the data specified at the central section of the Functional view (which are origin, destination, dateDeparture, etc).
(3) Finally, the operations that can be invoked using the data contained in this context are located in the bottom section of the Functional view. In the exemplified Transportation context, the *ProceedBooking* activity is made accessible to proceed with the booking of the selected offer as well as the cancellation of those rejected offers that were temporarily booked. To accomplish this operation an input form is provided to the user in order to fill in the credit card details required for this operation.

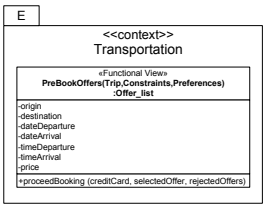

Fig. 14 Transportation Navigational Context

5.5 Presentation Modeling

Once the navigational model has been built, we specify presentational requirements using the *Presentation Model*. Presentation requirements are specified by means of properties that are associated to the primitives of the navigational context. This specification is strongly based on the Navigational Model. It allows us to specify the organization of data included in the Navigational Model.

To define this model we make use of the basic presentation patterns defined by the OOWS approach, which are *Information paging*, *Ordering criteria* and *Information Layout*. Fig. 15 shows the presentation defined for the Transportation context. It defines by means of the Ordering criteria that the data contained in the context must be presented ordered ascendant by price. It also defines the layout in which data must be organized. We decided to show the offer list following the register pattern.

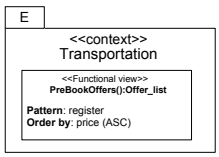

Fig. 15 Transportation Presentation Context

6. Building the PSM for the Services Model

At the PSM level Services Models should be defined as many as different technologies our application is going to interact with. For instance, we should have a Services Model for *Web Services* in case our application needs to interact with partners that provide their functionality by means of this technology.

The PSM for Services Models represent the specific technological aspects of the different technologies. To build these models we take the specific interface specifications that they provide (WSDL for WS or IDL for CORBA).

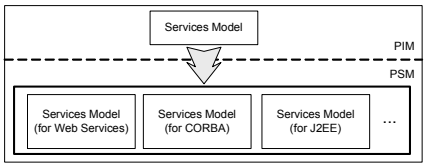

Fig. 16 Services Model at PIM and PSM levels

Once the PSMs are built, in order to solve the differences that arise when integrating various applications (different interfaces, different protocols, different data formats, etc.) we should make use of adapters. These adapters should define the mappings between the operations defined at the PIM (abstract representation) and the ones included at the PSM (specific representation defined by providers). In Fig. 17 we show two Web Services imported from two different travel agencies (BalearicTripsWS and OceanicTripsWS). Both provide a set of operations that fulfil the ones modelled in our PIM Services Model. However, we still need to link the operation/s from each service with the generic ones. For instance, the `search_offer` operation defined for the Broker Agent at the PIM Services Model is fulfilled by the `searchOffers` operation from the OceanicTripsWS and by the `getOffers` and `temporaryBooking` operations from the `BalearicTripWS`.

Fig. 17 An excerpt of two Imported Web Services

7. Generating a Web Service for the Broker Agent

In order to make available to external parties the Broker Agent implemented in our system, we are going to generate a Web Service (WS) that provides access to its functionality.

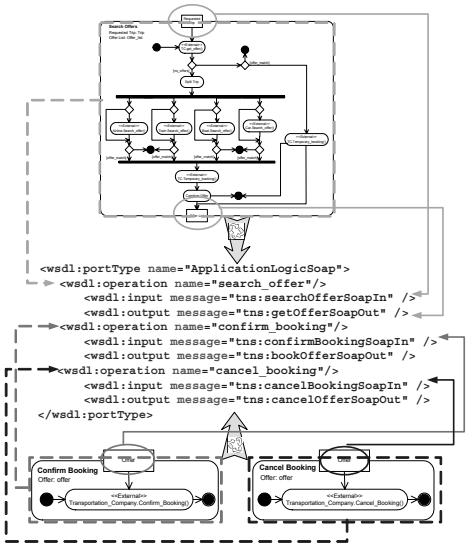

Fig. 18 Application Logic WSDL for the Broker Agent

For the construction of the WSDL definition of the WS we take the business processes defined for the internal Broker Agent in the Business Processes Model. This model was made up of three processes which are going to be included in the WS. These are `search_offer`, `cancel_booking` and `confirm_booking`. These processes definitions include the necessary information for building the WSDL definition (including the input and output parameters of each process).

A graphical schema that defines how to obtain the WSDL definition is provided in Fig. 18.

8. Generating the Interaction with External Parties

Depending on the technology that a provider uses to export its functionality, we should provide an appropriate solution for each case. As Fig. 19 shows, for each instance from each PSM Services Model we should build the required client artifact in charge of interacting with the corresponding service provider.

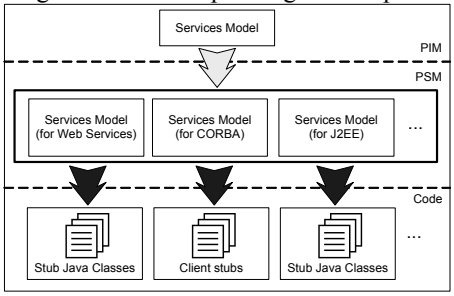

Fig. 19 Generating the appropriate Interaction

Based on the interfaces supplied by their respective providers we can generate the corresponding clients to fulfill the interaction with the external partner.

For instance, when functionality is provided following the Web Services model we generate from the associated WSDL definition the necessary client stubs to consume the supplied operations via SOAP, REST or XML-RPC depending on the characteristics of the available WS.

9. Generating the Web Interface

Web interfaces are generated taking as input the information modeled in the Navigational and Presentation Model. For the running example, we have modeled in section 5.4 the Transportation context as a functional view for the PreBookOffer operation.

The web interface that implements the Transportation context includes (as the top of Fig. 20 shows) direct access to those Navigational Contexts that we have defined as Exploration Contexts (Preferences, Transportation, Tourist Packages and Excursions).

The set of input parameters of this operation define the information that is required to the user in order to provide the corresponding data (an offer list provided by external broker agents). In Fig. 20 we can see how the input parameters (trip details, constraints and preferred transportation) are included in the Transportation context to allow the user to specify the parameters for the PreBookOffer operation.

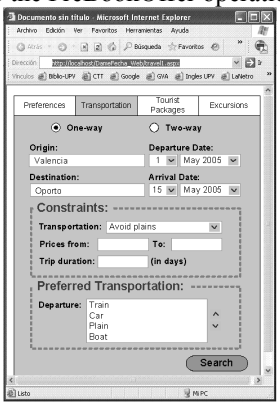

Fig. 20 Web Form for the Transportation Context

When this operation is executed, the context is shown as a Web page (see Fig. 21) that includes the offer list gathered from the Broker Agents. In Fig. 21 we can identify the register pattern applied to the retrieved offer list from the Broker Agents that work with our system.

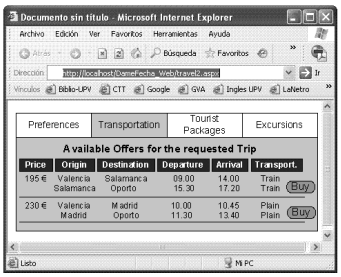

Fig. 21 Web Page with the Offer list

10. Conclusions and Further Work

In this work we have presented through the TAS case study the set of models that need to be built in order to develop a Web application that integrates functionality from external parties.

This approach is being incorporated to the OO-Method CASE tool (the software automatic production environment that gives support to the OO-Method [13]).

As further work we have planned to define the transformations that generate automatically the Web Services not only to provide the business logic of the application, moreover we plan to provide the information gathered also in the Navigational and presentation models.

In order to define transformations between models defined at CIM and PIM level we follow a strategy based on graph transformations. After completing the preliminary PIM models obtained from this first transformation, we plan to use the OlivaNova tool to obtain the application code for a specific platform.

10. References

[1] D. Schwabe, and G. Rossi, "An Object Oriented Approach to Web-Based Application Design", *Theory and Practice of Object System 4(4),* Wiley and Sons, New York, 1998, ISSN 1074-3224.

[2] O. De Troyer and C. Leune, "WSDM: A user-centered design method for Web sites", *In Proc. of the 7th International World Wide Web Conference*, 1998.

[3] N. Koch and M. Wirsing, "Software Engineering for Adaptive Hypermedia Applications", *In 3rd Workshop on Adaptive Hypertext and Hypermedia*, 2001.

[4] N. Koch, A. Kraus, C. Cachero and S. Meliá, "*Integration of Business Processes in Web Application Models*". Journal of Web Engineering. Vol. 3, No. 1 (2004)

[5] M. Brambilla, S. Ceri,, S. Comai, P. Fraternali and I. Manolescu, "*Model-driven Development of Web Services*

and Hypertext Applications", SCI2003, Orlando, Florida, July 2003

[6] 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

[7] R. Tongrungrojana and David Lowe, "*WIED: A Web Modelling Language for Modelling Architectural-Level Information Flows*". Journal of Digital Information, Vol 5 Issue2.

[8] J. Fons, V. Pelechano, M. Albert and O. Pastor, "Development of Web Applications from Web Enhanced Conceptual Schemas", *Proc. Of the International Conference on Conceptual Modelling, 22nd Edition, ER'03*, Chicago, EEUU, 2003, pp. 232-245.

[9] MDA Guide Version 1.0.1.

[10] Object Management Group. Unified Modeling Language (UML) Specification Version 2.0 Final Adopted Specification. www.omg.org, 2003.

[11] F. Paternò, C. Mancini and S.Meniconi, "ConcurTaskTrees: a Diagrammatic Notation for Specifying Task Models", INTERACT'97, Chapman & Hall, 1997, pp. 362-369.

[12] Wirfs-Brock, B. Wilkerson, and L. Wiener, "*Designing Object–Oriented Software*.", *Prentice–Hall*, 1990.

[13] O. Pastor, J. Gómez, E. Insfrán and V. Pelechano, "The OO-Method Approach for Information Modelling: From Object-Oriented Conceptual Modeling to Automated Programming", *Information Systems Elsevier Science*, 2001, Vol. 26, Number 7, pp. 507-534