Developing BP-driven Web Applications through the Use of MDE Techniques

VICTORIA TORRES, PAU GINER, VICENTE PELECHANO

Centro de Investigación en Métodos de Producción de Software

Abstract. Model Driven Engineering (MDE) is a suitable approach for performing the construction of software systems (in particular in the web application domain). There are different types of web applications depending on their purpose (i.e., document-centric, interactive, transactional, workflow/business process-based, collaborative, etc). This work focuses on business process-based web applications in order to be able to understand business processes in a broad sense, from the lightweight business processes already addressed by existing proposals to long-running asynchronous processes. This work presents a MDE method for the construction of systems of this type. The method has been designed in two steps following the MDE principles. In the first step, the system is represented by means of models in a technology-independent manner. These models capture the different aspects of web-based systems (these aspects refer to behaviour, structure, navigation, and presentation issues). In the second step, the model transformations (both model-to-model and model-to-text) are applied in order to obtain the final system in terms of a specific technology. In addition, a set of Eclipse-based tools has been developed to provide automation in the application of the proposed method in order to validate the proposal.

1. Introduction

Nowadays, most of our daily activities (communication by text or voice, listening to music, watching TV programs, etc.) can be performed on the web. The web provides applications that are almost always available and are easy to access (even resource-constrained devices provide a web browser and connectivity). The web development industry has grown very rapidly since its beginning in the mid-1990s. Multiple technologies have emerged to deal with both client (JavaScript, Flash or Microsoft Silverlight) and server (ASP, CGI, PHP, Python, etc.) side coding. In addition to these technologies, the availability of web application frameworks (i.e. Django[1], Ruby on Rails[2], Tapestry[3], etc.) and Web Engineering (WE) methods (OOWS [13], UWAT+ [9], WebML [7], OOHDM [29], UWE [21], OOH [17]) alleviate the common problems

[1] http://www.djangoproject.com/
[2] http://rubyonrails.org/
[3] http://tapestry.apache.org/

found when building applications of this type. Web application frameworks provide solutions to common activities such as database access or session management performed in web development. WE methods provide techniques and notations at a high level of abstraction to represent the knowledge required for the development of these systems. In addition to the infrastructure provided by web frameworks and WE methods, a significant improvement in the development of web applications is provided by the application of Model Driven Engineering (MDE) techniques [31]. These techniques propose the use of models and model transformations as the principal artefacts during application development. Models are used to specify web applications in a technology-independent fashion. Model transformations are defined to move technology-independent web specifications into a specific technology (i.e., to a particular web framework).

In the WE field, many of the existing proposals have been extended and adapted since their conception to satisfy the needs of new emerging types of web applications (i.e., semantic web applications [3] or RIA [11] among others). However, we have found some limitations and drawbacks in some of these extensions, particularly in those related to the integration of business processes (BPs) with navigational issues. In an attempt to solve these limitations, in this work, we present a WE method for the construction of BP-driven web applications. This method takes into account BP aspects that were not considered by the existing proposals. The main aspects considered in our approach are the support for: (1) multiple mechanisms for collaborative work within an organization; and (2) the ability to seamlessly handle processes that involve a different number of participants (from lightweight step-by-step wizard-like processes to a long asynchronous process that involves many partners). Thanks to these considerations and through the application of MDE techniques, our method generates web applications that support the execution of the BPs defined during modelling time. In addition, we have developed a set of tools based on the Eclipse platform (the Bizzy tool) to validate the proposal. This tool covers the proposal from the modelling to the generation phase and includes a set of modelling editors and model transformations that allow the specified web application to evolve into a specific technology.

The remainder of the paper is structured as follows. Section 2 presents the type of web applications that we deal with. Then, based on the observable characteristics of applications of this type, section 3 enunciates the set of requirements that WE methods should satisfy in order to properly build systems of this type. Section 4 presents the state of the art of the WE field and also in the Human-Computer Interaction (HCI) and BP fields regarding the issues considered in this work. Section 5 provides an overview of the entire MDE proposal developed in this work. Section 6 presents the new language primitives that have been defined in this work to properly specify BP-driven web applications at the modelling level. Based on the requirements identified in section 3 and the two case studies presented in section 2, section 7 explains the details of the proposal focusing only on the aspects related to BP specification. Section 8 provides some details of the Eclipse-based tools that we have developed to support the proposal. Section 9 provides the results obtained from the evaluation of our proposal. Finally, section 10 presents conclusions and further work.

2. What do we mean by BP-driven web applications?

BP-driven web applications are defined as applications that are accessed via a web browser over a network (i.e., the Internet or an intranet). These applications allow users to accomplish a set of key tasks or workflows. Therefore, applications of this type are considered to be task-based application and correspond to one of the web application categories identified by Kappel et al. in [20]. The aspects that characterize BP-driven web application are the following:

- **Well-defined process**. When users access BP-driven web application they have restricted freedom. They navigate in a controlled manner through the application only having access to the content and functionality that is required according to the business process definition.
- **Humans-system cooperation**. BPs tend to cross organization boundaries and integrate different computing resources and services [8]. In addition, a complete automation of a process is not always possible or desirable [34]. Thus, human participation cannot be overlooked and guidance must be provided to make human participation as easy as possible.

In line with the BP categorization[4] made by IBM, we have catalogued BPs into two types, which we have termed short-running (microflows in IBM nomenclature) and long-running BPs (the same in IBM nomenclature). Michael Havey also uses this categorization in the *Short and Long-Running Processes in SOA-part1*[5] article. Even though Havey extends this categorization with mid-running processes, these are not considered in the article since these can be handled similarly to short-running processes.

Short-running and long-running processes are explained in detail and exemplified with a case study in the following subsections. For the case study we use the Library4U web application that offers its users the common services that are usually found in on-line library systems. In this system, books and other materials can be borrowed and purchased. Of the services offered by these systems, we present the *checkout process* and the *book purchase request process,* which allow us to exemplify both short-running and long-running BPs, respectively. The notation that has been used in this work to specify BPs is the Business Process Modeling Notation [27] (BPMN), which is the notation adopted by the OMG to represent processes for different types of users, from business analysts to technical developers.

2.1. Short-running business processes

According to the IBM categorization, short-running BPs are defined as "*a process that is contained within a single transaction. This is ideal for situations where the user is expecting an immediate response*". Good examples of processes of this type are the *checkout process,* which is usually found in on-line stores, or the *booking service,* usually found in on-line travel agencies.

[4] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.bpel.ui.doc/concepts/clngmcro.html
[5] http://www.packtpub.com/article/short-and-long-running-processes-soa-1

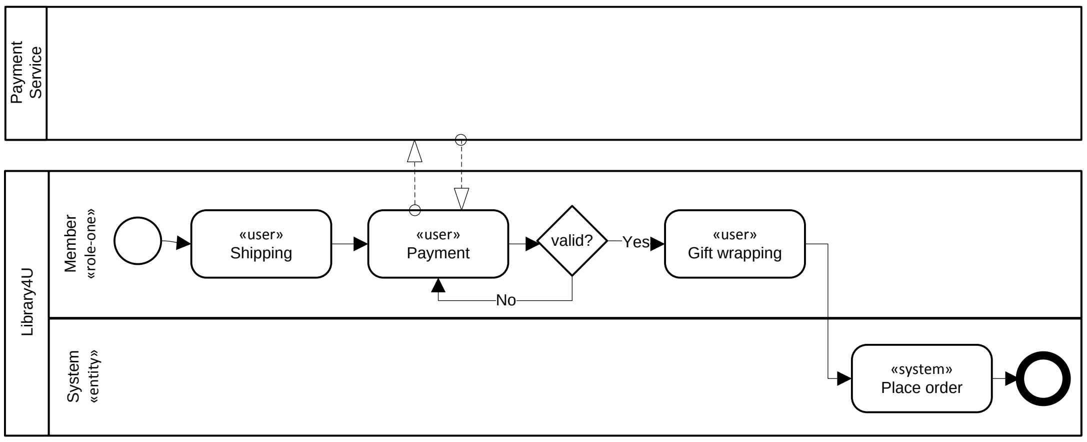


Figure 1 Example of a short-running process: the *checkout* process

Figure 1 shows the *checkout process*. This process involves the coordination of two systems that are represented in the diagram by the *library4U* and the *payment service* pools. The *library4U* pool represents the system being developed locally, while the *payment service* pool refers to an external partner from which the *library4U* uses some services. In addition, the tasks assigned to the *library4U* partner are distributed between two roles, a human being (lane labelled as *member*) and an automated system (lane labelled as *system*). The process defined in the diagram details the steps that are required to accomplish a virtual purchase. In this BP, it is the human participant (*member* role) who starts the *checkout process*. During the first stage of this process the user is asked to input some information about the shipping details such as address, city, country, shipping mode, etc. In the second stage, the user is asked to provide data about the payment (i.e., credit card number and expiration date). Once the required information is introduced, the process starts a payment validation step. If the validation step is completed successfully, the user is asked about wrapping options. Otherwise, the user is redirected again to the payment step to introduce the payment data. Finally, if the validation step succeeds the process terminates by creating and placing the order into the system. Unlike the previous tasks, this task is performed by the system (*system* role) automatically without any interaction with the user.

After analyzing several examples of short-running BPs, we have generalized the following features for them:

Feature 1 (Participants) They involve just one human participant who interacts with the system/process and one or more automated participants.

Feature 2 (Duration) They are completed in a very short period of time (intervals can range from seconds to a few hours). This is due to the fact that there is only one human user required by the process.

Feature 3 (Complexity) They are usually simple (in terms of control flow) and are not very large. Processes of this type normally take the form of a wizard, which is a sequential on-screen dialog that assists the user to achieve a specific goal.

Feature 4 (Initiator) They are always started by the user (human participant). Processes of this type usually correspond to services offered by the system to its users, who take the initiative to use them when necessary.

Feature 5 (Process instances) Since there is no need for coordination with other human participants, users normally complete the process step-by-step without parallel instances being started. As a result, the user only participates in one instance of the process at the same time.

Existing web frameworks such as Spring[6] or Seam[7] provide solutions (WebFlow[8] and Pageflow[9], respectively) for coping with these processes. Despite the fact that these solutions can be considered valid at the implementation level, they fail in using a standardized language to define BPs and in using techniques to accelerate and facilitate the construction of the Graphical User Interface (GUI) associated to the process. In our approach for dealing with these issues we make use of both a standardized BP language (the BPMN notation), which is specifically designed to define BPs and MDE techniques (model transformations) to produce the necessary GUI to execute the BP. This enables the use of existing standard-compliant technologies and reduces the need for manually creating and updating GUIs.

[6] http://www.springsource.com/
[7] http://www.seamframework.org/
[8] http://www.springsource.org/Webflow
[9] http://docs.jboss.com/seam/latest/reference/en-US/html/tutorial.html#numberguess

2.2. Long-running business processes

The second type of BP that we have categorized is represented by long-running BPs, which usually define the protocols that have to be followed within an organization to achieve a specific goal. Based on the IBM BP categorization, a long-running BP "*executes over an extended period of time, and is much more flexible and resilient than a microflow. Interruptible business processes and asynchronous business processes are examples of long running processes".* Examples of this sort of BPs are the *book loan* and *return service* offered by libraries.

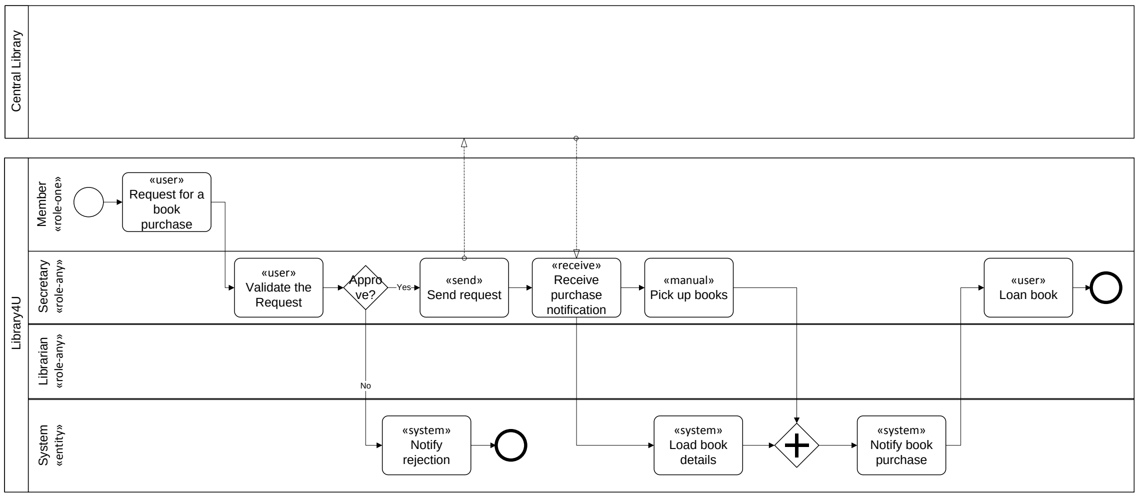

Central Library
Library4U
Member «role-one»
Secretary «role-any»
Librarian «role-any»
System «entity»
«user» Request for a book purchase
«user» Validate the Request
Appro ve?
Yes
No
«send» Send request
«receive» Receive purchase notification
«manual» Pick up books
«user» Loan book
«system» Notify rejection
«system» Load book details
«system» Notify book purchase

Figure 2 Example of a long-running process: the *book purchase request* process

In general, these processes involve not only the coordination of different systems but also the coordination of different people behaving with different roles (such as the *librarian*, *secretary,* or *member* roles represented by lanes in Figure 2). To exemplify these processes we are going to make use of the *book purchase request process* which is depicted in Figure 2. As this diagram shows, there exists coordination between two partners that refer to the local library (represented by the *library4U* pool) and the library warehouse (represented by the *central Llibrary* pool). This BP is started by a library user who wants to borrow a book that it is not available in the library. This user is represented by the *member* lane within the *library4U* pool. Therefore, to perform the loan, the user has to first provide the information of the book (i.e., book title or ISBN). Then, the book request has to be evaluated by any user belonging to the secretarial staff (represented in the diagram by the *secretary* lane) who decides to approve or reject the request based on certain criteria. If the request is rejected, the system notifies the member (usually by sending an e-mail). On the contrary, if the purchase request is approved, this request is redirected to the *central library*, which is responsible for the purchase of the requested books (among other services). At this point, the process has to wait for the purchase notification response sent back by the *central library*. When this notification arrives, the *library4U* system must load the book details into the system and, in parallel, any user belonging to the secretary group has to pick up (manual operation) the book from the *central library*. When these two tasks are completed, the system notifies the member about the acquisition of the requested book and, finally, one of the users belonging to the *librarian* user type finalizes the process by making the book loan to the solicitor member.

Again, after analyzing several examples of long-running BPs, we have generalized the following features for them:

Feature 1 (Participants) They usually involve more than one human participant and one or more automated systems. Collaboration among different roles and organizations is required to fulfil a business goal.

Feature 2 (Duration) They usually take a long time to be completed (intervals can range from days to years). This is due to the fact that there may be several human participants, which implies that processes have to wait for information from all of them.

Feature 3 (Complexity) They usually represent organizational protocols or procedures that have to be followed in order to accomplish a specific goal. In these processes, tasks are usually distributed not only among different roles within the organization but also among external partners. In addition, procedures can be very complex since they should consider all the possible ways to achieve a specific goal (taking into account the differences among partners in terms of internal policies, data formats, and possible exceptions). Therefore, all these factors contribute to not only having long process models but also having complex ones as well.

Feature 4 (Initiator) They can be started by any participant involved in the process. It can be started by either an automated task or by a human participant.

Feature 5 (Process instances) Due to the delay introduced by the participation of different human roles in long-running BPs, it makes sense to focus on specific tasks from the process instead of following a step-by-step approach. For example, in the *book purchase request process*, it is more practical for secretary members to validate all the pending requests before picking up the books. This proceeding allows several requests to be solved without having to performing multiple cycles of book validate-wait and reception-pick up.

The lack of specific primitives provided by web frameworks to support BPs forces developers to implement ad-hoc solutions. In fact, these frameworks follow a synchronous request-response pattern that, for instance, does not allow maintaining the state of the process. Therefore, these solutions are not originally prepared to support BPs of this type. The most suitable solution for supporting them is given by Business Process Management solutions. These solutions provide support to the entire life cycle of BPs, including the design and execution phases we are focused on. However, the design of the GUI required to support the specified BPs is carried out separately from the BP specification. This implies that changes in the definition of BPs have to be manually propagated to the associated GUI. Again, in our proposal, the application of MDE techniques allows us to keep both, the BP definitions and the GUI that supports them synchronized.

3. Requirements for dealing with BP-driven web applications

Based on the characteristics that are observable in the two types of BPs identified in section 2, we present the requirements that are particularly important for WE methods targeting the specification and construction of BP-driven web applications. These requirements concern not only the method and the development process but also the characteristics of the generated web applications.

Requirement 1 (BP data and functionality)
Description This requirement refers to the availability of mechanisms to specify which data and functionality from the system has to be associated with a specific task. Specifically, system functionality should only be associated with tasks that require some

system support (this does not happen with manual tasks that do not change the state of the system).

Rationale Performing this association is necessary in the context of a MDE method. This association allows us to achieve a level of automation during the system development process that could not be achieved in other situations. By explicitly identifying the data and functionality that is required to complete a specific task in the models, we can generate the code that will allow users to retrieve this data and executing this functionality.

Requirement 2 (BP definition)

Description This requirement refers to the availability of mechanisms to build BP diagrams that define the tasks, the roles responsible for these tasks, and the connections that define the different paths that can be executed to complete the BP.

Rationale In BP-driven web applications, BPs constitute a major artefact in the development process. Therefore, it is necessary to provide mechanisms (in terms of a language or notation) that allow their definition.

Requirement 3 (Work distribution)

Description This requirement refers to the mechanisms that allow BPs to be defined in a distributed context where some process tasks are delegated to external partners.

Rationale Real scenarios involve the cooperation of different systems to achieve a specific goal. This cooperation comes from the need for some external services that are provided by specific organizations or the need for internal services that are simply distributed among different systems.

Requirement 4 (Human participation)

Description This requirement refers to the available mechanisms that allow different types of human participation within a BP definition to be specified.

Rationale There are different ways in which human beings can participate in a specific process. Sometimes this participation requires some software assistance, but in other situations, humans complete their tasks without any software support. In addition, BP tasks can either be assigned to just one human participant or they can be assigned to a group of users. In order to properly handle the interaction in each case, it is important to clearly specify the type of behaviour/participation that is expected from the user in the corresponding model.

Requirement 5 (Separation of concerns)

Description During the system definition phase, this requirement refers to the mechanisms that are available to clearly differentiate between the navigation that occurs during BP execution from the navigation that occurs from traditional navigation (navigation driven by the user and aimed at content discovery or execution of atomic functionality). This differentiation will impact the generation of the corresponding web application, both enabling and disabling the links that should and should not be followed by the user.

Rationale The web application navigation required to execute a specified BP is related to the flow connections defined in the corresponding BP diagram. In this case, the navigation is clearly defined and the user does not really have the chance to decide the next link to follow.

Requirement 6 (Differentiating navigation types)

Description This requirement refers to the GUI mechanisms provided to the end user (the one executing the BP) to distinguish between a short-running and a long-running BP.

Rationale Differentiating these two types of BPs at the application level allows the end user to better understand the process and the tasks that she/he is involved in. As we have illustrated in section 2, users complete tasks in a different way depending on the nature of the process (short-running or long-running). For short-running processes where tasks are performed step-by-step, navigation must guide the user through all the steps of the process, providing mechanisms to cancel or to go back/forward (e.g., following the wizard pattern). Conversely, for long-running processes, mechanisms are required to access the different instances of the process and to complete several pending instances of a specific task. Task-based UIs [18] that are based on the to-do list metaphor for users to represent multiple instances of pending can be applied for the design of the UI in this case.

Requirement 7 (BP instance state)

Description This requirement refers to maintaining the state process instances in order to correctly take them up again after their suspension.
Rationale Users may need to suspend a process for a while for different reasons. Therefore, it is necessary to provide mechanisms that allow the process to be taken up again at the same point where it was suspended.

Some of these requirements (specifically requirements 1, 2, 3 and 7) have already been stated by Damiano et al. in [10]. However, we found that it was also important to make the difference between the two types of BPs identified in section 2, but only from the point of view of the web application. In fact, handling these two types similarly from the modelling point of view allows developers to focus on just the problem domain and not on the solutions that must be implemented to successfully support the BP. In this work, we have designed a MDE approach taking into account these requirements. During the system specification phase, the analyst/developer is provided with modelling mechanisms that specify the particularities that BPs introduce (req. 1 to req. 4). All these primitives are provided in different models according to the type of element being modelled. In addition to these mechanisms, the method has been designed taking into account the separation of concerns regarding navigational issues (req. 5), since navigation has a very important role in web applications. Other requirements such as generating a specific GUI according to the type of BP being considered (req. 6) are also taken into account during the system generation process. This is achieved by the application of the corresponding model transformations. Finally, to properly handle the state of the BP instances, an extension over the traditional three-tier architecture has been designed (req. 7). The details about how all these requirements are captured in the MDE approach are explained in section 6.

4. Related work

Based on the requirements that we have identified in section 3, in this section we present an overview of the existing literature dealing with the construction of BP-driven web applications. Most of this literature has been developed in the context of the WE field. However, there are also some works developed in the HCI and BP fields that deal with the alignment of GUI and BPs which address how human participation should be addressed in process modelling. Thus, this section has been organized in two parts. The first part briefly describes the most well-known WE methods that provide support for dealing with the construction of BP-driven web applications. The second part includes related works that have been developed in the HCI and BP fields.

It was in 2003, during the third International Workshop on Web-Oriented Software Technologies (IWWOST)[10], when the WE community started to consider how WE methods should deal with the construction of BP-driven web applications. As a result of this workshop, the OOHDM [30], UWE [22], OO-H [22] and WSDM [36] proposals presented a solution to deal with these applications. After this event, other proposals such as UWAT+ [10], WebML [4], HERA [2] or MIDAS [26] have also provided a solution for dealing with them. The most evolved one corresponds to WebML which places emphasis on the following: (1) improving the usability of the generated UIs [5] (by means of the RUX approach [24] to generate RIAs); and (2) developing WebRatio [6], a commercial tool that puts into practice the approach. However, these WebML advances, and others proposals fail in a solution that considers the particularities of the two types of BPs that we have identified in this work (short-running and long-running BPs). This drawback refers mainly to requirement 6 (Differentiating navigation types). The OOHDM, UWE, OO-H, WSDM and MIDAS methods only consider dealing with short-running BPs. This does not involve many changes to their current proposals since the interaction of several human beings or the existence of multiple process instances (not satisfying requirement 7 (BP instance state)) are not involved. In fact, the proposed solutions simply introduce modelling mechanisms to specify BPs (such as BPMN, UML Activity Diagrams or Concur Task Tree), which are finally mapped to navigational structures to allow users to execute the tasks involved in the BP (these modelling mechanisms refer to requirement 2 (BP definition)). In addition to these mechanisms, the OOHDM proposal also defines special navigational links that differentiate between navigation during BP execution and traditional navigation. UWAT+, WebML and HERA focus on providing support for long-running BPs. Dealing with these BPs involves taking into account that some processes are completed by several human beings and, therefore, the navigation defined to achieve a specific goal involves the coordination of different users. Also users may be involved in several

[10] http://users.dsic.upv.es/~west/iwwost03/articles.htm

instances of the same process, which can also be in different stages. In this case, it is necessary to maintain the state of each process instance in order to allow users to complete their pending tasks whenever they are required to do so, thereby satisfying requirement 7 (BP instance state)).

From the modelling point of view, and taking into account that most of the WE methods rely on MDE, all the revised proposals mix the navigation that occurs during BP execution and the navigation that occurs during traditional navigation at the modelling level. This mix hinders model legibility and understandability (note that BP navigation requires control mechanisms to redirect navigation to one path or another, which makes model legibility and understandability of the model even more difficult. Taking into account the important role played by models in a MDE approach, it is necessary to ensure the quality of the models in order to complete software developments successfully.

This consideration refers to requirement 5 (separation of concerns). Taking into account that BP models already define flows that connect BP tasks (these flows represent the navigational links between the elements that form the navigational structure), WE methods should consider keeping these connections in the BP model only, and not bringing them to the navigational model.

WE methods define a set of models to specify different aspects of web applications. For instance, the system data and functionality is usually specified by means of UML class diagrams, entity-relationship models, UML use cases or the like. By associating these models with the model that captures the navigational structure of the system, these proposals specify which data and functionality is going to be provided to the user in each different interaction unit (these units will render to web pages). This association refers to requirement 1 (BP data and functionality), which allows users to complete BP tasks by accessing the proper data and executing the proper system functionality.

Related to requirement 3 (Work distribution), only the HERA, MIDAS and WebML methods consider the collaboration with external parties in order to support some of the activities included in the BPs. In contrast, OOHDM, OO-H, UWE, WSDM and

UWAT+ conceive web applications as isolated systems where all the activities of the BPs are supported by the web application itself.

Finally, users can participate in different ways in a specific BP. For instance, in some cases, the user is the only one responsible for specific tasks. However, there may be situations in which the responsibility of completing a specific task can be shared by a set of users. This refers to requirement 4 (Human participation) and from the revised proposals, none of which provide mechanisms to specify the different behaviour in which a user can participate in a BP.

Table 1 summarizes the support provided by each of the previously analyzed proposals to deal with the integration of BPs in web applications.

	UWAT+	WebML	OOHDM	UWE	OO-H	WSDM	HERA	MIDAS
BP data and functionality	+	+	+	+	+	+	+	+
BP definition	+	+	+	+	+	+	+	+
Work distribution	-	+	-	-	-	-	+	+
Human participation	-	-	-	-	-	-	-	-
Separation of concerns	-	-	-	-	-	-	-	-
Differentiating navigation types	-	-	-	-	-	-	-	-
BP instance state	+	+	-	-	-	-	+	-

Table 1 WE methods summary (+ fully supported, - not supported)

Besides the works developed in the WE field, there are some works ([19], [25] or [28]) that focus on how human participation should be addressed in process modelling. The goal of these works is not to develop web applications supporting BPs, but they relate to our work since they take into account the user interaction required to support BP tasks. All these works make use of MDE techniques to produce BP executable definitions in terms of BPEL4People[11] and WS-HumanTask[12] standards from models or views that represent human participation. MDE techniques allow them to automatically derive BP

[11] http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf

[12] http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf

definitions in terms of a specific language. Part of our approach is similar to these since we also make use of MDE techniques to derive the executable BP definition from a set of models (BP, structural, services, navigational and presentation model). However, in our case, we make use of the WS-BPEL [1] standard. To cope with human participation, we have developed the *Task Manager Service* which is presented in section 7.8.

In the HCI field, we can also find some works aimed at aligning BP definitions with User Interfaces (UI). In this field, there are works such as the one conducted by Sousa et al. in [32] or Sukaviriya et al. in [33]. In both works, the main purpose is to achieve traceability between BP and UI. However, these differ from the perspective that they take. In [32], traceability is focused on the end-user perspective. BPs are connected with GUIs in order to easily identify the effect of changes in both BP and GUIs. In this case, BP models are linked to UI through the mapping between the elements from the Task model and the User Interface model (both models are based on UsiXML [23]). However, in [31], traceability is focused on the business requirements in order to help UI designers to react to business changes appropriately. Both approaches propose the use of MDE techniques to achieve the alignment between BPs and GUIs. However, neither of them applies these techniques to the development of the system, and they are simply used to speed up the construction of the UI associated to BP tasks.

5. Proposal overview

The Business Process Management (BPM) initiative[13] promotes the use of models to describe business processes from a high abstraction level (e.g., using BPMN). Then, these models can be translated into executable representations of the process (e.g., using WS-BPEL . In line with this intensive use of models, we find MDE. Putting MDE into practice involves three steps. First of all, it is necessary to define a language (metamodel) that allows specific kinds of systems to be expressed. Then, in a second step, this language can be used to create models that represent different system instances. Finally, in order to execute these models, it is necessary to transform them into an executable representation. This last step is achieved by means of model transformations that allow the system represented in the models to evolve into a specific implementation technology. Metamodels, models, and model transformations constitute

[13] http://www.bpmi.org/

the building blocks that make the application of MDE approaches possible. However, to make a proper use of an approach like this, the steps that define the development process as well as the artefacts resulting from each step must be defined.

Independently of the software development model used in the definition of a software development process (waterfall or iterative), a complete process involves several steps from requirements gathering to software maintenance. Nevertheless, in this work, the development process is focused on just two of these steps which relate to the *design* and *implementation* of the web system.

The first step of our proposal is the *design* of the web system. The system is represented in terms of the models defined by the method (see section 5.2). Then, the second step is to transform all of these models into a specific implementation technology, which allows the execution of the modelled system (see section 5.3). During this second step, the *implementation* of the web system is performed by the application of model transformations, this allows moving from the problem space (real world concepts) to the solution space (system implementation).

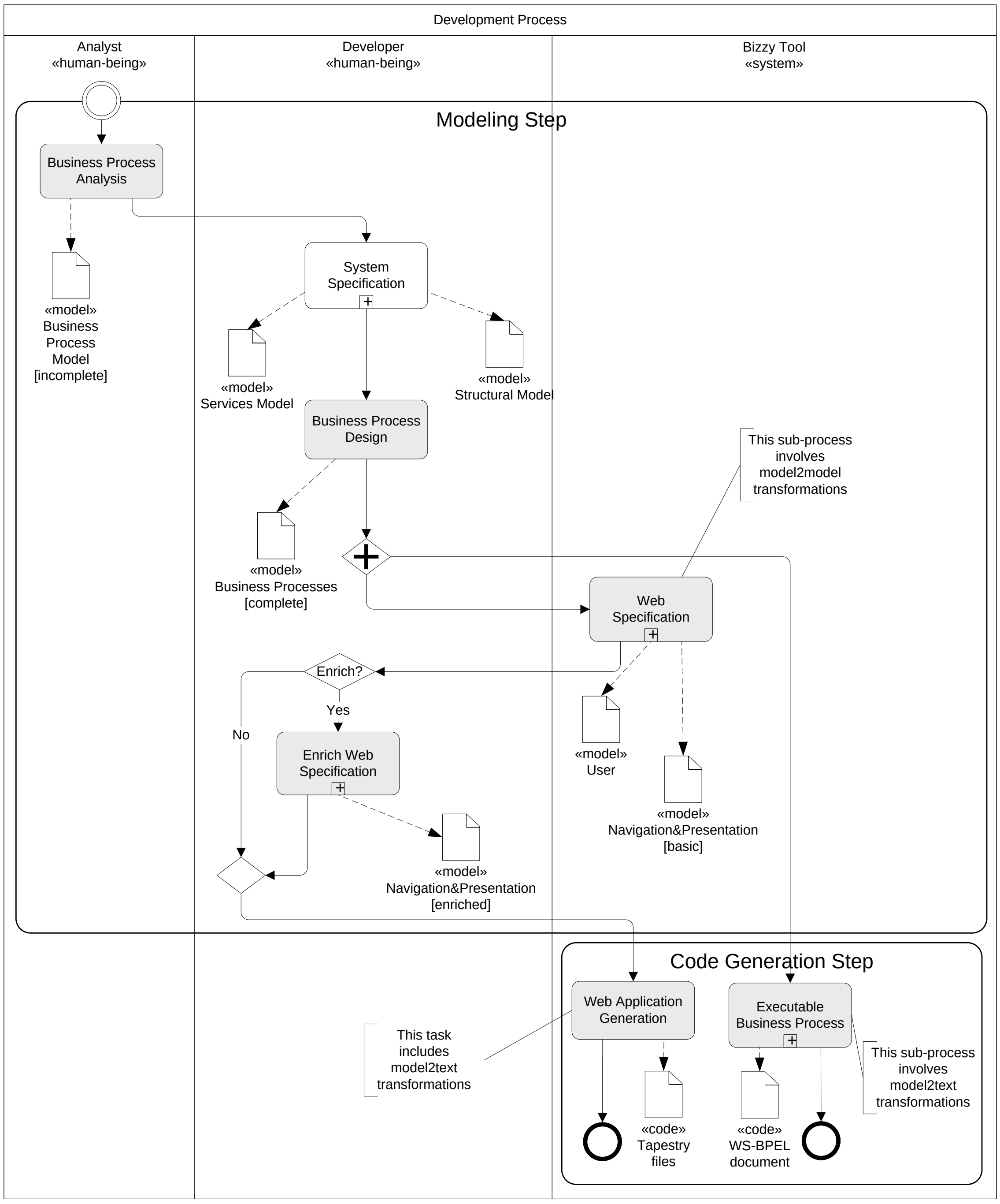
5.1. The big picture

Figure 3 shows the development process defined to build BP-based web applications (note that all the models referenced in this figure are briefly introduced in subsection 5.2). The process involves the participation of three different roles, two of which relate to human-beings (the *analyst* and the *developer*) and one to a system (the *Bizzy tool*). This differentiation is denoted graphically by the corresponding stereotypes. In addition, we have divided the tasks into two steps according the type of work performed: the modelling step and the code generation step.

The process is started by the *analyst* who defines the set of processes that should be supported by the system using BPMN. This model does not include any details about the real work that is going to be performed during the process tasks. Therefore, we have termed the generated model as an incomplete[14] BP model (Figure 3 shows the artefact obtained in the *business process analysis* task). The main reasons for using BPMN in

[14] in terms of system executability

our work to specify BPs are that: (1) it constitutes the standard for BP modelling; (2) there is an open source editor for creating BPMN models that is extensible and based on the Eclipse platform; and (3) there is some support for transforming BPMN models into executable WS-BPEL processes.



Development Process
Analyst «human-being»
Developer «human-being»
Bizzy Tool «system»
Modeling Step
Business Process Analysis
«model» Business Process Model [incomplete]
System Specification
«model» Services Model
«model» Structural Model
Business Process Design
«model» Business Processes [complete]
This sub-process involves model2model transformations
Web Specification
Enrich?
Yes
No
Enrich Web Specification
«model» User
«model» Navigation&Presentation [basic]
«model» Navigation&Presentation [enriched]
Code Generation Step
Web Application Generation
Executable Business Process
This task includes model2text transformations
This sub-process involves model2text transformations
«code» Tapestry files
«code» WS-BPEL document

Figure 3 Development process for the construction of BP-driven web applications

Then, the *developer* starts the development process. Based on the previously generated incomplete BPs, the *developer* complements these BPs with new models (Figure 3 shows the *system specification* sub-process task), these are used to specify: (1) the structure of the system in terms of classes, attributes, and operations; and, (2) the external functionality that is going to be consumed by the system. As a result we obtain the *structural model* and the *services model*, respectively, which gather all this information (a brief description of these models is provided in section 5.2). These models allow the structure and functionality of the application to be handled independently of the technology employed to implement them.

Although the development process proposes performing first the *business process analysis* and then the *system specification*, these tasks can be performed the other way round. The order in which these tasks are going to be performed is determined by the way in which the system requirements are discovered. In some cases, requirements are provided in terms of a well-known process. In this case, it is more appropriate to start with the *business process analysis* task. However, when requirements are provided in terms of domain concepts and uni-granular functionalities, it is more appropriate to start with the construction of the *structural* and *services models*.

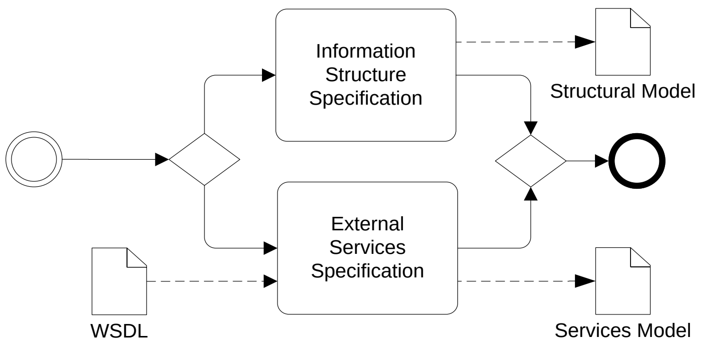

Information Structure Specification
Structural Model
External Services Specification
WSDL
Services Model

Figure 4 Expanded sub-process system specification

The following step in the process is still performed by the *developer*. At this point, the *developer* completes the BPs that were defined by the *analyst* in order to generate an equivalent representation, in a later step; however, it will be generated in terms of the WS-BPEL executable language. This is achieved by associating the operations defined either in the *structural model* or in the *services model* with the tasks included in the BPs. In some cases, the *developer* has to refine tasks (i.e., by splitting them into several connected tasks) in order to fit their granularity to the operations defined in the *structural* and *services models*.

Once the system has been shaped in the above-mentioned models we can bring them into the *Bizzy tool* and use them for the generation of new artefacts (new models and executable code). This tool automates the model transformations that generate: (1) the *navigational* and *presentation models* (*web specification* task in Figure 5) that represent the interaction between users and the system; (2) the executable BPs in terms of WS-BPEL; and (3) the Java code that implements the systems in terms of the Tapestry web framework.

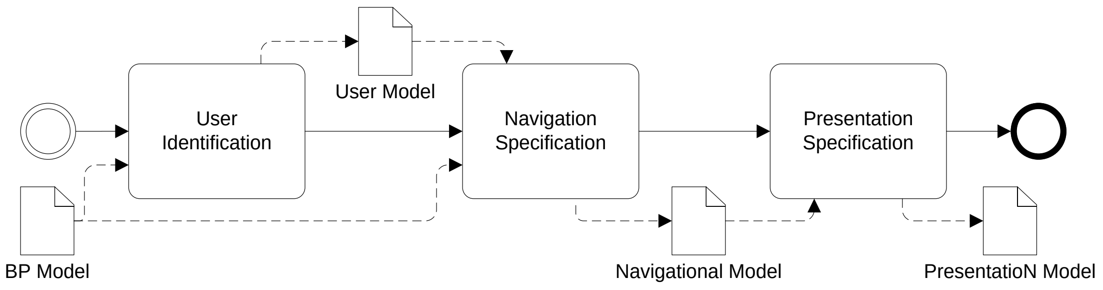


Figure 5 Expanded sub-process web specification

The following subsections provide some details about the artefacts that are obtained in the tasks that make up both the modelling and the code generation steps of the process.

5.2. The modelling step

Following the separation of concerns promoted by MDE, this step gathers all the process tasks that relate to system specification (independent of any technological details) and that generate the models that capture the different aspects of the system. These models are:

- **The structural model**: This model defines the system structure by means of a UML class diagram. In this diagram, classes, attributes, operations, and relationships between classes are defined. In this model, we are going to define all the data and functionality developed and maintained in our local system.
- **The services model**: This model defines the external functionality (functionality that is provided by external systems) that is going to be consumed by the system. The objective of this model is to bring up external services (services offered as web services) to the modelling level in order to manage them more easily.

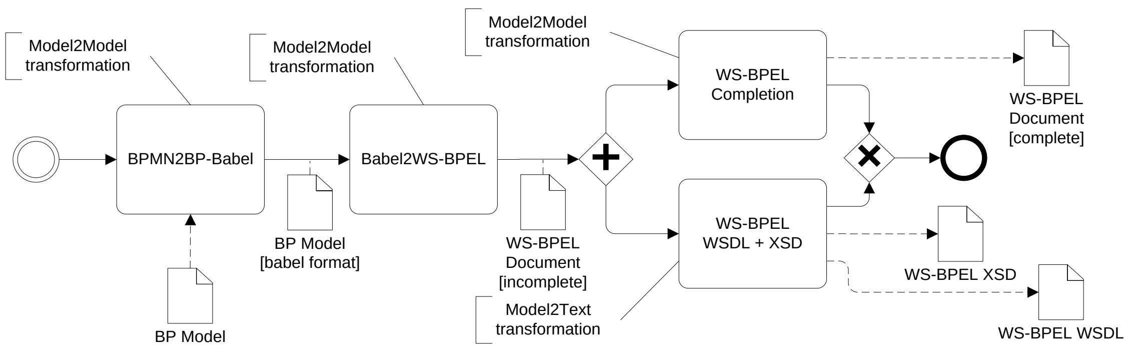
- **The business process model**: This model is used to specify the set of BPs that have to be supported by the system. Its specification is performed by means of the BPMN notation, which has been extended in this work (see section 6.1) in order to derive not only WS-BPEL code but also the corresponding web application.
- **The user model**: This model defines the kind of users that are going to interact with the web application. In this model, we can also define hierarchical relationships between different types of users, thus allowing them to share their navigational privileges.
- **The navigational model**: This model captures the navigational structure of web applications. This structure is defined as a view over the system, that is, over the *structural* and *services models*. The *navigational model* includes a view for each type of user identified in the *user model*.
- **The presentation model**: This model defines the presentation properties of the web application content (data and functionality). These properties are related to information paging, layout, and ordering criteria and are applied to the views defined in the *navigational model*.

5.3. The code generation step

In the second step in the development process, the domain specified in the set of models presented above is transformed into code artefacts that can be executed. This phase of the process has two tasks: one dedicated to the code generation of service orchestration and another dedicated to the generation of the interface that will allow users to interact with the BPs supported in the web application.

5.3.1. WS-BPEL code generation step

To obtain an executable version of the BPs specified at the modelling level, we have defined the process depicted in Figure 6. In this process, we make use of the BABEL Java tool (*Babel2WS-BPEL* task) to translate the BPMN diagrams into WS-BPEL code. However, before performing this activity, the BP models must be prepared according to the format accepted by the BABEL tool. This is performed in the *BPMN2BP-Babel* task by means of a model-to-model transformation that we have implemented for this purpose.


Model2Model transformation
Model2Model transformation
Model2Model transformation
BPMN2BP-Babel
Babel2WS-BPEL
WS-BPEL Completion
WS-BPEL WSDL + XSD
BP Model [babel format]
WS-BPEL Document [incomplete]
WS-BPEL Document [complete]
WS-BPEL XSD
WS-BPEL WSDL
Model2Text transformation
BP Model
Figure 6 Expanded Sub-process WS-BPEL generation

In spite of the fact that the BABEL tool generates a WS-BPEL document from a BPMN model, the generated WS-BPEL document is not complete since this tool does not consider the extension performed in our proposal to BPMN (see section 6.1). Therefore, to obtain a complete and ready-to-run WS-BPEL code we have implemented two new transformations that can be performed in parallel. The *WS-BPEL completion* task completes the WS-BPEL definition including the *Partner Link*, *Variables*, and *Correlation Sets* sections of the document. Since WS-BPEL processes are considered web services, it is necessary to generate their interfaces to define their operations and data types. This is performed in the *WS-BPEL WSDL + XSD* task.

5.3.2. User interface code generation step

The generation process to obtain the modelled system into a web framework is performed in just one step (see Figure 7). In this step (*web application generation* task in Figure 3), a set of model-to-text transformations is executed to obtain the modelled system in terms of the Tapestry web framework. The main reasons for adopting this framework were the component-based model and the MVC architecture in which the framework is based on. This allows a clear separation of different technologies used to be distinguished during the development of the web application.

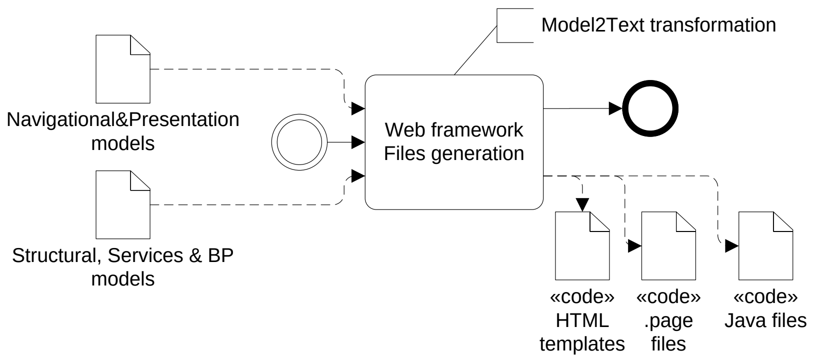

Model2Text transformation
Navigational&Presentation models
Web framework Files generation
Structural, Services & BP models
«code» HTML templates
«code» .page files
«code» Java files

Figure 7 Expanded sub-process tapestry web framework generation

As Figure 7 shows, the *web framework files generation* task requires the complete specification of the system to produce a web application that is organized in three types of code files:

- **Java files**, which define the Java classes that implement the logic of the application. It corresponds to the *Controller* aspect from the MVC architecture on which the framework relies.
- **HTML files**, which define the templates that correspond to the *View* aspect from the MVC architecture.
- **Page files**, which are XML documents that include the declaration of the Tapestry components used in the HTML files. Although page files are optional, they help to obtain a more readable code by allowing the application of the dependency injection pattern [14].

6. Language extensions to support BP requirements

In order to satisfy the requirements identified in section 3, we have extended the languages/notations used in the current proposal. These extensions refer to the BPMN notation and to the *navigational model* of the OOWS web modelling language [12].

6.1 BPMN extension

According to the type of information that is gathered in the BPM, the BPMN notation provides graphical elements to specify: (1) the systems (external and internal) that are involved in the BP; (2) the activities that conform the internal (private BP) system; (3) the participants (within the private BP) that are responsible for performing these activities; (4) the conditions that control the BP flow; and (5) the interaction that occurs between the private BP and the external partner(s). Within the set of graphical objects defined by the BPMN notation, we have found almost all of the elements necessary for defining the kinds of processes that we are interested in. Nevertheless, we have found some limitations in the notation. In accordance with the requirements identified in section 3, we have extended the BPMN metamodel. The limitations and their extensions are the following:

- *Differentiating the scenarios where human participants behave as individuals or as members of a particular group*. In order to differentiate the behaviour of a process role, we have extended the BPMN *lane* element with a new attribute, the *type* attribute. The values accepted by this new attribute are *role-one* and *role-any*. The *role-one* value is used when the user behaves as an individual. Therefore, the human being performing the first task of the lane has to be the same for the rest of tasks defined within the same lane. The *role-any* value is used when the user behaves as a member of a group. In this case, any human being belonging to the group specified in the lane can perform any of the tasks included in it.
- *Specifying the functionality that is going to support each task/activity included in the BP definition*. A necessary step for obtaining an executable definition of the process is to specify the functionality that is going to be performed in each task. For this purpose, we have extended the BPMN *task* element with a new attribute, the *operation* attribute, which allows us to link tasks with functionality that has been defined either in the *structural* or in the *services model*. This extension only applies to tasks defined as *service*, *receive*, *send*, and *user* in the BPMN notation (these are some of the types in which a task process can be defined). Process tasks defined as *script* or *manual* are not susceptible to this extension.

Specifically, the proposed extensions have been defined over the BPMN Metamodel included in the BPMN modeller[15] from the STP (SOA Tools Platform) project. The main goal of this modeller is to provide a graphical notation that allows BPs to be defined according to the BPMN specification. This tool has been developed based on

[15] http://www.eclipse.org/stp/bpmn/

GMF (Graphical Modeling Framework) and reuses and extends the GEF (Graphical Editing Framework) and the EMF (Eclipse Modeling Framework) projects.

6.2 Navigational extension

The interaction that takes place between the user and the system during the execution of a BP is different from the one occurring during traditional navigation. An important difference is that, during the execution of a BP, users are guided through a set of steps to accomplish a specific objective. This *controlled* navigation does not occur during traditional navigation where it is the user who decides the links to follow. Therefore, in order to specify this *controlled* navigation, in this section, we present the extensions designed over the OOWS *navigational model* to clearly capture the particularities of BP navigation. These are the limitations found in the original notation and the extensions defined to cope with them:

- *Specify the distributed data and functionality that is necessary to complete a specific task*. The original notation (the OOWS approach) provides us with primitives (such as the *class-view* primitive) to build views over data and functionality defined in the *structural model*, that is, from the local system. However, since we are interested in defining BPs whose data and functionality can also be provided by external systems, we have defined two new primitives (*service-data-view* and *service-functional-view* primitives) that allow us to define (data and functional) views over the services defined in the *services model* and that are required to perform a specific BP activity.
- *Specify access points to reach BPs*. To allow users to reach the BPs supported by the web application, we have included two new primitives, the *process-context* and the *process-link* primitives. Similarly to traditional navigation, BPs can be accessed directly from every part of the web application (what is known in the OOWS proposal as *exploration context*), or they can be accessed only from restricted web locations (what is known in the OOWS proposal as *sequence context*). Therefore, the *process-context* primitive is introduced to represent the navigation required by a BP. In turn, a *process-context* can be defined as *exploration* or *sequence* depending on the access type required. If the *process-context* has been defined as *sequence*, it is necessary to associate a

process-link to the *process-context* in order to restrict the parts from which users can reach the corresponding BP.

- *Specify complementary information to help users during the execution of a specific BP task*. During the execution of some BP tasks, users need to check the data that has some relationship with the current task. To do this, we have introduced the *complementary-AIU* primitive, which complements the GUI that is provided to the user to complete a specific BP task with some extra data. By providing users with information related to the BP activities we: (1) help them complete the activity; and (2) prevent them from switching from BP execution to traditional navigation to reach that information.
- *Showing different GUI according to the type of activity being executed*. According to the different types of activities that can be defined in a BP we need to make this differentiation explicit at the navigational level. This differentiation allows us to generate a more appropriate GUI according to the type of activity being performed. To do this, we have introduced the *main-AIU* and *human-AIU* primitives. The *main-AIU* primitive refers to activities whose execution modifies the state of the underlying information system. The *human-AIU* primitive refers to activities that are not automated in the system and that are fully performed without the assistance of any system.

7. Dealing with BP requirements through case studies

In this section we explain which extensions have been applied to the requirements that were identified in section 3 and the strategy followed to satisfy these requirements.

7.1 BP data and functionality

During the BP definition, we have to specify the system data and functionality that supports each task. This information is specified in the *BP* and in the *navigational* models. An extension to the notation used to build the *BP* model has been defined. This extension specifies which operation (from those defined either in the *structural* or *services models*) is going to be executed to support each task. For instance, as Figure 8 indicates, in the *book request purchase BP* case study, the two first tasks labelled as *request for a book purchase* and *validate the request* are associated with the *createRequest()* and *validateRequest()* operations defined in the *structural model*.

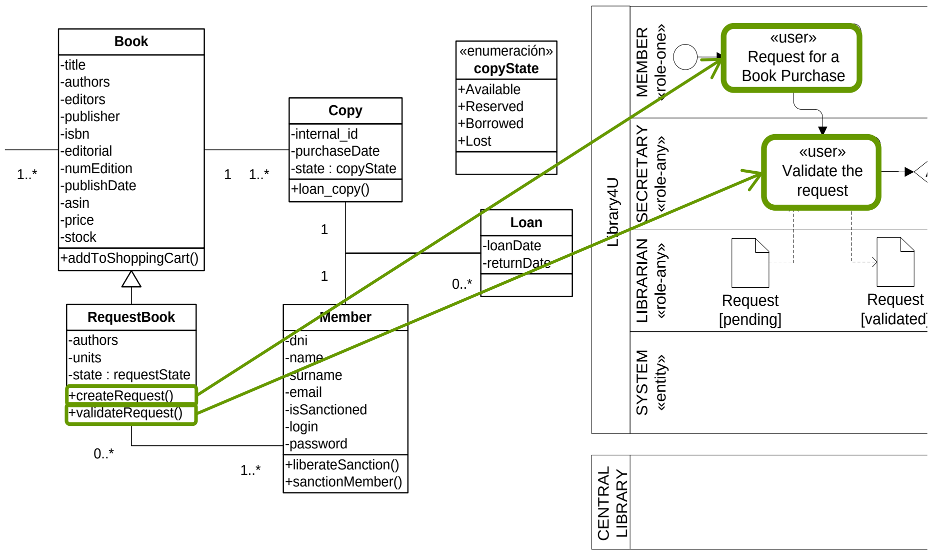


Figure 8 Association between operations defined in the structural model (left) and tasks defined in the BP model (right)

To allow users to complete a specific task, the GUI has to provide them with some data and functionality. This data and functionality is going to be shown to the user as views over the *structural* and the *services model*, with these views being defined in the *navigational model*. Therefore, the following primitives have been defined in the *navigational model*:

- **Class-view**: This primitive is used to define views over the classes defined in the *structural model*. These views filter the set attributes and operations of the class making visible only those required to complete a specific task.
- **Service-data-view**: This primitive is used to define data views over the services provided by external partners. It specifies the data types (from those returned by the service) that are required by the associated task.
- **Service-functional-view**: This primitive is used to define functional views over the services provided by external partners. It specifies the operation that is necessary to support the associated task (from the set provided by the service).

In addition to these views, users can be provided with some extra data that can help them complete a specific task. To differentiate the mandatory from the optional elements, we have defined the following primitives:

- **Main-AIU (Abstract Interaction Unit)**: This primitive gathers both data and functionality that *mandatorily* has to be provided to the user to complete a specific task.
- **Complementary-AIU (Abstract Interaction Unit)**: This primitive gathers data that is *optionally* provided to users to help them complete a specific task.

These two new primitives behave as mere containers of elements (data and functionality) but allow the navigational content to be better organized, thereby improving its legibility and maintainability.

7.2. BP definition

To support this requirement, we rely on the BPMN notation used to build the BPM. This notation provides modeling mechanisms (such as *lanes*) that allow BP tasks to be assigned to different roles according to role responsibilities. In addition, these roles are used to identify the different types of users of the web applications (types that are gathered in the *user model*). Note, however, that during BP definition, no relationship between different roles is specified. Therefore, we have to manually define (if necessary) the inheritance relationships between these types of users in the *user model*. The BP flow is defined during the BP specification, that is, during the construction of the BP model. The expressivity provided by the BPMN notation (i.e., *fork* and *merge gateways* or *loop activities*) allows us to represent the potential paths that can be taken during BP execution. In this work, the flow need only be specified in the BP model, and not bringing in the *navigational model* as other proposals require (current proposals from the WE field use navigational links to represent the flow that was already defined in the BP diagram). This is possible since we make use of a process engine (see section 7.8) that determines the set of activities that are ready to be executed in a specific process instance.

7.3. Work distribution

Some BPs make use of services/functionality that are provided by different distributed systems. To deal with this distribution, we make use of the BPMN mechanism provided for it, specifically the *Pool* and the *Message Flow* elements. These mechanisms allow us to specify: (1) different organizations (systems) involved in the same BP; and (2) the way they cooperate to accomplish the BP goal (through the interchange of messages).

7.4. Human participation

Manual tasks represent activities that are completely performed by human beings without any support regarding the web application (i.e., a task whose work implies organizing a meeting). To specify this fact, the *human-AIU* primitive has been defined in the *navigational model*. Similarly to the *main-AIU* and *complementary-AIU* primitives, this primitive behaves as a container of elements. However, in this case, these elements are limited just to data. This is because no system functionality is required to support the current task. Figure 9 shows the web page corresponding to the *PickUpBook* human-AIU container, which provides the user with the necessary data from the system to complete this task (In this case, the data provided is the book title, book isbn and number of units of the book to pick up).

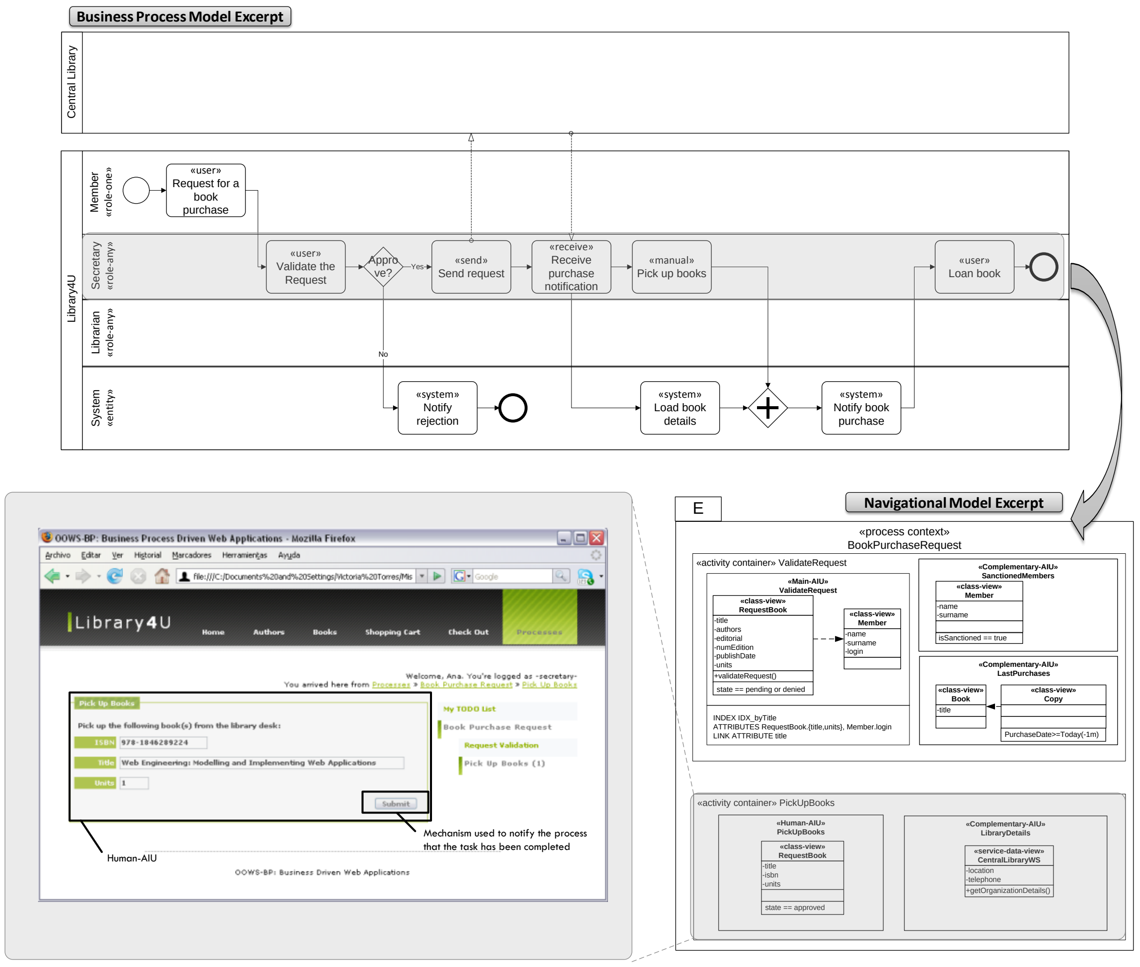

Business Process Model Excerpt
Central Library
Library4U
Member «role-one»
Secretary «role-any»
Librarian «role-any»
System «entity»
«user» Request for a book purchase
«user» Validate the Request
Approve?
Yes
No
«send» Send request
«receive» Receive purchase notification
«manual» Pick up books
«user» Loan book
«system» Notify rejection
«system» Load book details
«system» Notify book purchase
OOWS-BP: Business Process Driven Web Applications - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
file:///C:/Documents%20and%20Settings/Victoria%20Torres/Mis
Library4U
Home
Authors
Books
Shopping Cart
Check Out
Processes
Welcome, Ana. You're logged as -secretary-
You arrived here from Processes » Book Purchase Request » Pick Up Books
Pick Up Books
Pick up the following book(s) from the library desk:
ISBN 978-1846289224
Title Web Engineering: Modelling and Implementing Web Applications
Units 1
Submit
My TODO List
Book Purchase Request
Request Validation
Pick Up Books (1)
Mechanism used to notify the process that the task has been completed
Human-AIU
OOWS-BP: Business Driven Web Applications
E
Navigational Model Excerpt
«process context» BookPurchaseRequest
«activity container» ValidateRequest
«Main-AIU» ValidateRequest
«class-view» RequestBook
-title
-authors
-editorial
-numEdition
-publishDate
-units
+validateRequest()
state == pending or denied
«class-view» Member
-name
-surname
-login
INDEX IDX_byTitle
ATTRIBUTES RequestBook.{title,units}, Member.login
LINK ATTRIBUTE title
«Complementary-AIU» SanctionedMembers
«class-view» Member
-name
-surname
isSanctioned == true
«Complementary-AIU» LastPurchases
«class-view» Book
-title
«class-view» Copy
PurchaseDate>=Today(-1m)
«activity container» PickUpBooks
«Human-AIU» PickUpBooks
«class-view» RequestBook
-title
-isbn
-units
state == approved
«Complementary-AIU» LibraryDetails
«service-data-view» CentralLibraryWS
-location
-telephone
+getOrganizationDetails()

Figure 9 Web page implementing the *pick up books* human-AIU

7.5. Separation of concerns

The separation of concerns principle promoted by the MDE approach allows a specific domain to be efficiently represented. In this case, the concern involved refers to navigation, where traditional navigation should be specified separately from the navigation that occurs during BP execution. To do this, we have included the following primitives in the *navigational model*:

- **Process-context**: This primitive has been introduced into the *navigational model* to distinguish traditional navigation from the navigation that occurs during BP execution. We can also specify the way users can access these contexts. If these contexts are going to be accessible from any part of the web application, then these are defined as *exploration* (as depicted in Figure 10). On the contrary, if these are going to be accessed through a predefined path (as depicted in Figure 10), then these are defined as *sequence*.

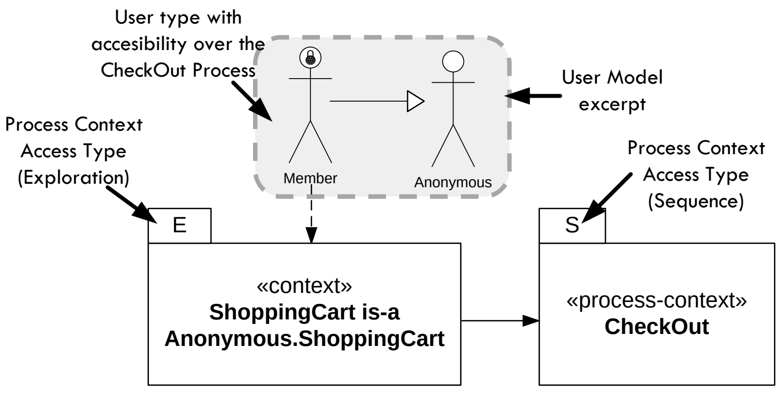

User type with accesibility over the CheckOut Process
User Model excerpt
Process Context Access Type (Exploration)
Process Context Access Type (Sequence)
Member
Anonymous
E
S
«context» ShoppingCart is-a Anonymous.ShoppingCart
«process-context» CheckOut

Figure 10 Navigational map for the member user type (Authoring-in-the-large)

Figure 10 corresponds to the navigational map for the member user type where no details about its content are provided (what is called the *Authoring-in-the-large* view). The details of these contexts are provided in the *Authoring-in-the small* view of the context as Figure 11 shows for the *book purchase request BP*. Figure 11 also includes most of the primitives defined in the proposal (*process-context, activity container, main-AIU, complementary-AIU, human-AIU, class-view, service-data-view,* etc.), which are explained throughout this section.

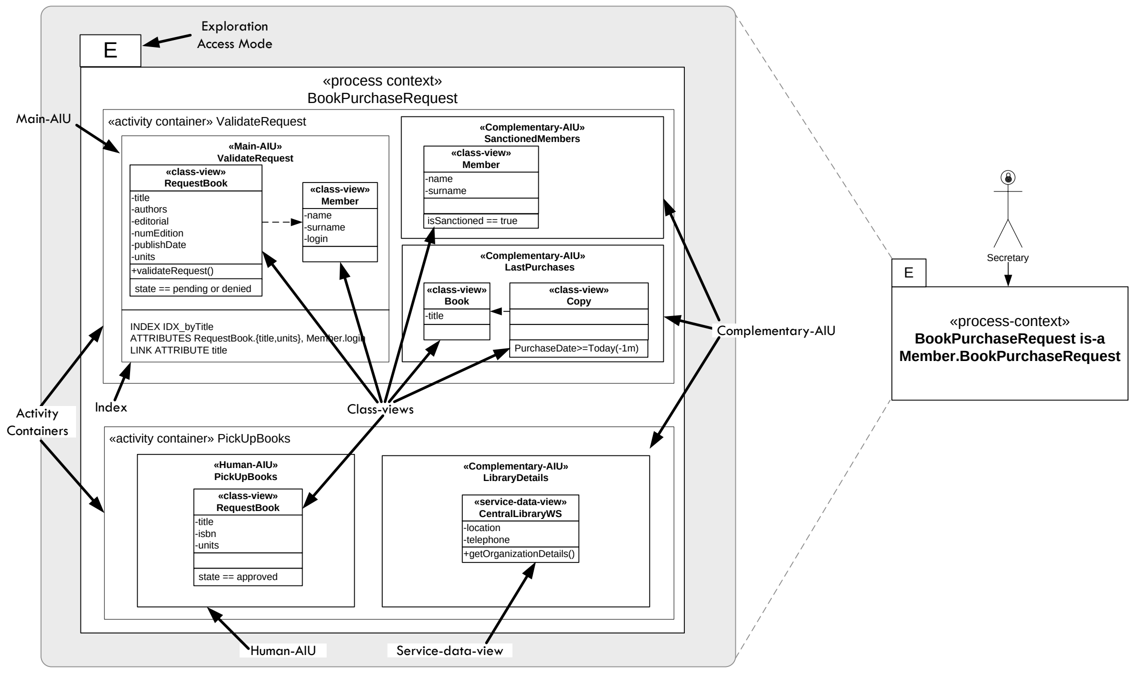
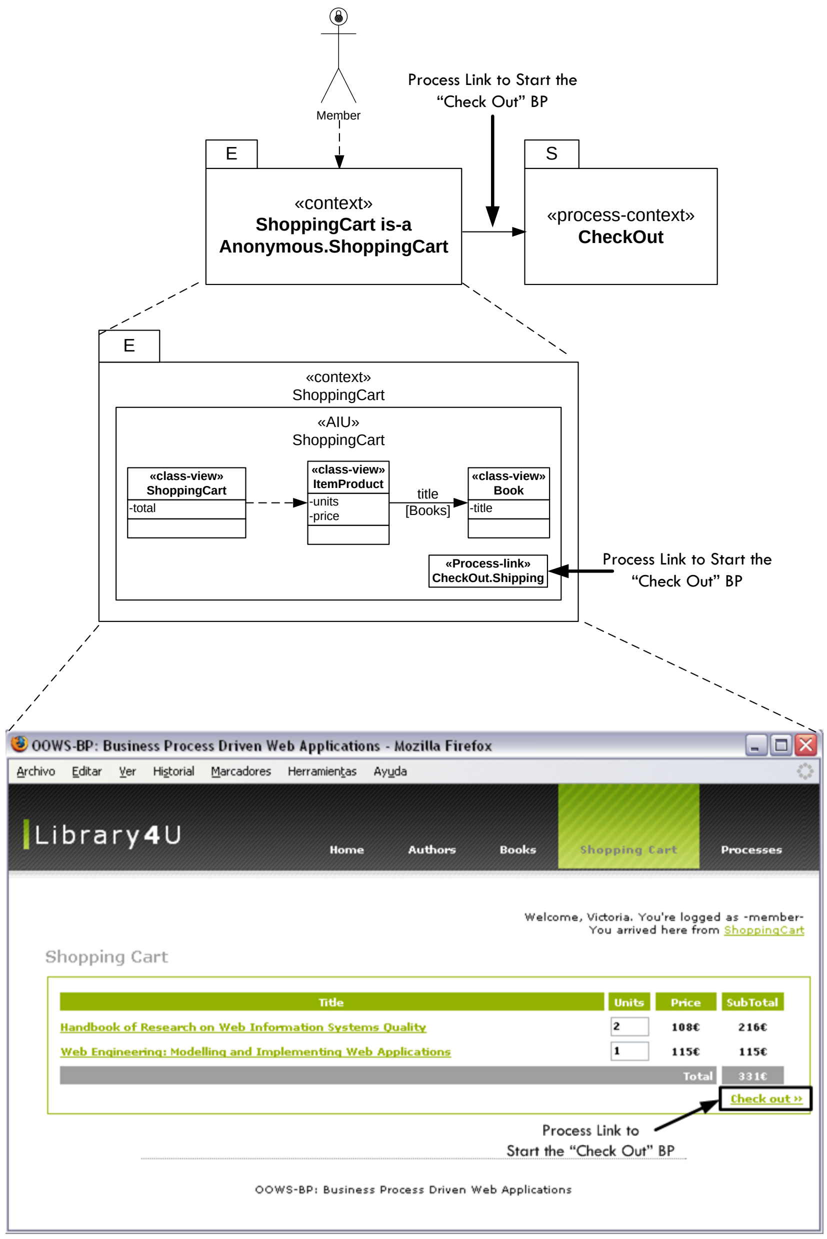


Figure 11 Book purchase request process-context (Authoring-in-the-small)

- **Activity-container:** Similarly to the *process-context* primitive, this primitive has been defined to behave as a container; however, in this case, its granularity is reduced to a business process task. It gathers all the data and functionality that is going to be provided to the user to complete a specific task.
- **Process-link**: This primitive is used to specify the anchor that allows users to start BPs when the associated *process-context* has been defined as *sequence* (see Figure 12). The access to the *process-contexts* defined as *exploration* is provided to users in the *processes* menu item (see Figure 12). In this section, the user is provided with: (1) a link to all BPs where the user starts the process; and (2) a to-do list with where the user can find easily the pending tasks.



Member
Process Link to Start the "Check Out" BP
E
«context»
ShoppingCart is-a Anonymous.ShoppingCart
S
«process-context»
CheckOut
E
«context»
ShoppingCart
«AIU»
ShoppingCart
«class-view»
ShoppingCart
-total
«class-view»
ItemProduct
-units
-price
title
[Books]
«class-view»
Book
-title
«Process-link»
CheckOut.Shipping
Process Link to Start the "Check Out" BP
OOWS-BP: Business Process Driven Web Applications - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
Library4U
Home
Authors
Books
Shopping Cart
Processes
Welcome, Victoria. You're logged as -member-
You arrived here from ShoppingCart
Shopping Cart
Check out »
Process Link to Start the "Check Out" BP
OOWS-BP: Business Process Driven Web Applications

Title	Units	Price	SubTotal
Handbook of Research on Web Information Systems Quality	2	108€	216€
Web Engineering: Modelling and Implementing Web Applications	1	115€	115€
		Total	331€

Figure 12 Web page corresponding to the *shopping cart* navigational context

7.6 Differentiating navigation types

Web applications should provide users with a navigational structure that assists them throughout the entire process, bringing them to the next step. However, the navigational structure of the application depends on the type of BP being executed. When the BP corresponds to a short-running process, a navigational structure that follows the wizard pattern[16] should be provided. Figure 13 shows the web page that implements how a short-running BP is presented to the user and how it is specified in the *BP* and *navigational models*.

[16] http://ui-patterns.com/pattern/Wizard

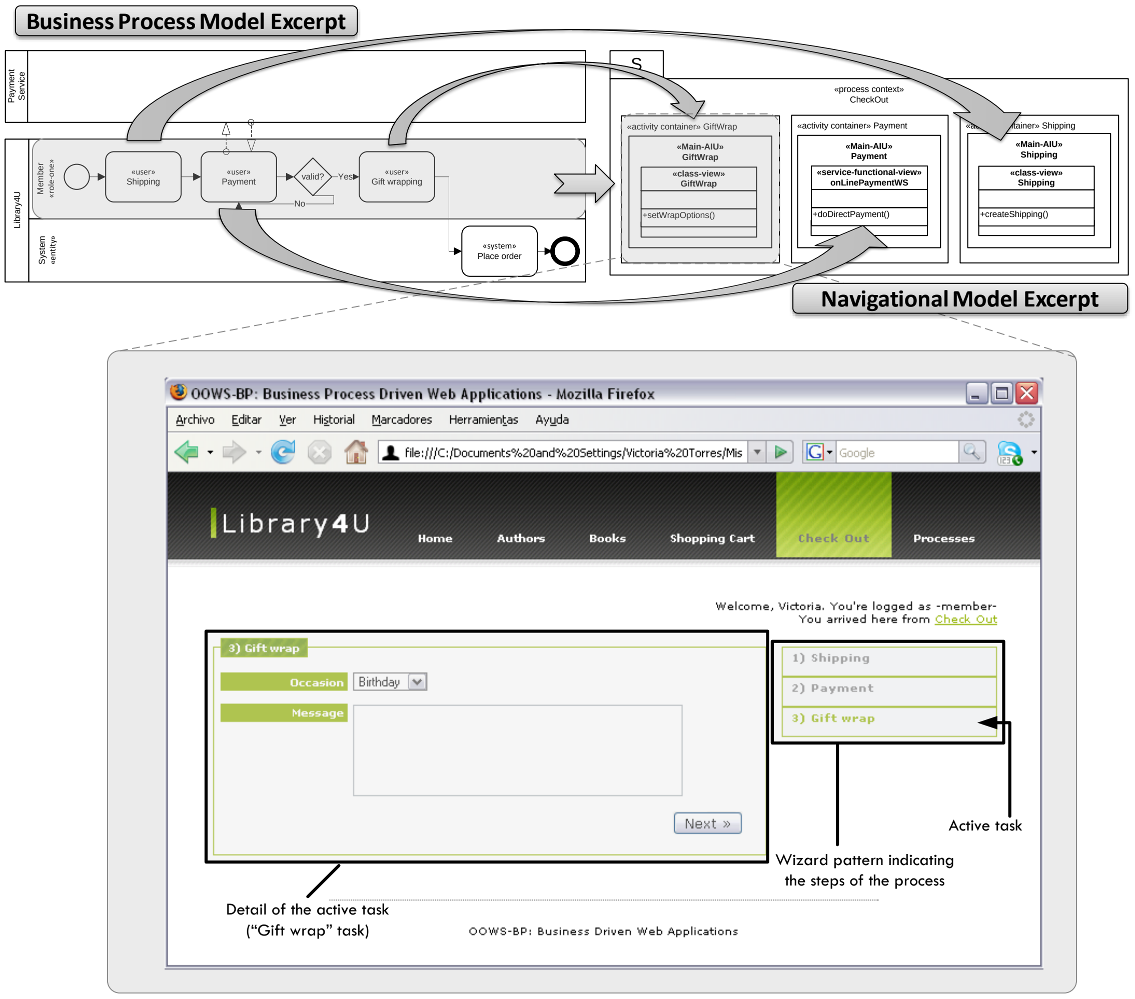

Business Process Model Excerpt
Payment Service
Library4U
Member «role-one»
System «entity»
«user» Shipping
«user» Payment
valid?
Yes
No
«user» Gift wrapping
«system» Place order
«process context» CheckOut
«activity container» GiftWrap
«Main-AIU» GiftWrap
«class-view» GiftWrap
+setWrapOptions()
«activity container» Payment
«Main-AIU» Payment
«service-functional-view» onLinePaymentWS
+doDirectPayment()
«activity container» Shipping
«Main-AIU» Shipping
«class-view» Shipping
+createShipping()
Navigational Model Excerpt
OOWS-BP: Business Process Driven Web Applications - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
file:///C:/Documents%20and%20Settings/Victoria%20Torres/Mis
Google
Library4U
Home
Authors
Books
Shopping Cart
Check Out
Processes
Welcome, Victoria. You're logged as -member-
You arrived here from Check Out
3) Gift wrap
Occasion
Birthday
Message
Next »
1) Shipping
2) Payment
3) Gift wrap
Active task
Wizard pattern indicating the steps of the process
Detail of the active task ("Gift wrap" task)
OOWS-BP: Business Driven Web Applications

Figure 13 Web page implementing the wizard pattern to navigate through the checkout process

Figure 13 shows the implementation of the wizard pattern, which shows the steps that have to be followed by the user and the current that is being performed step (indicated as *active task*). In this case, these three steps correspond to the three tasks assigned to the *member* user type (see business process model from Figure 13).

In contrast, Figure 14 shows the web page that implements how a long-running BP is presented to the user. When executing long-running process, a navigational structure with a to-do list should be provided (similarly to the way Business Process Management Systems do). To do this, different transformation rules have been defined to be executed during the generation step.

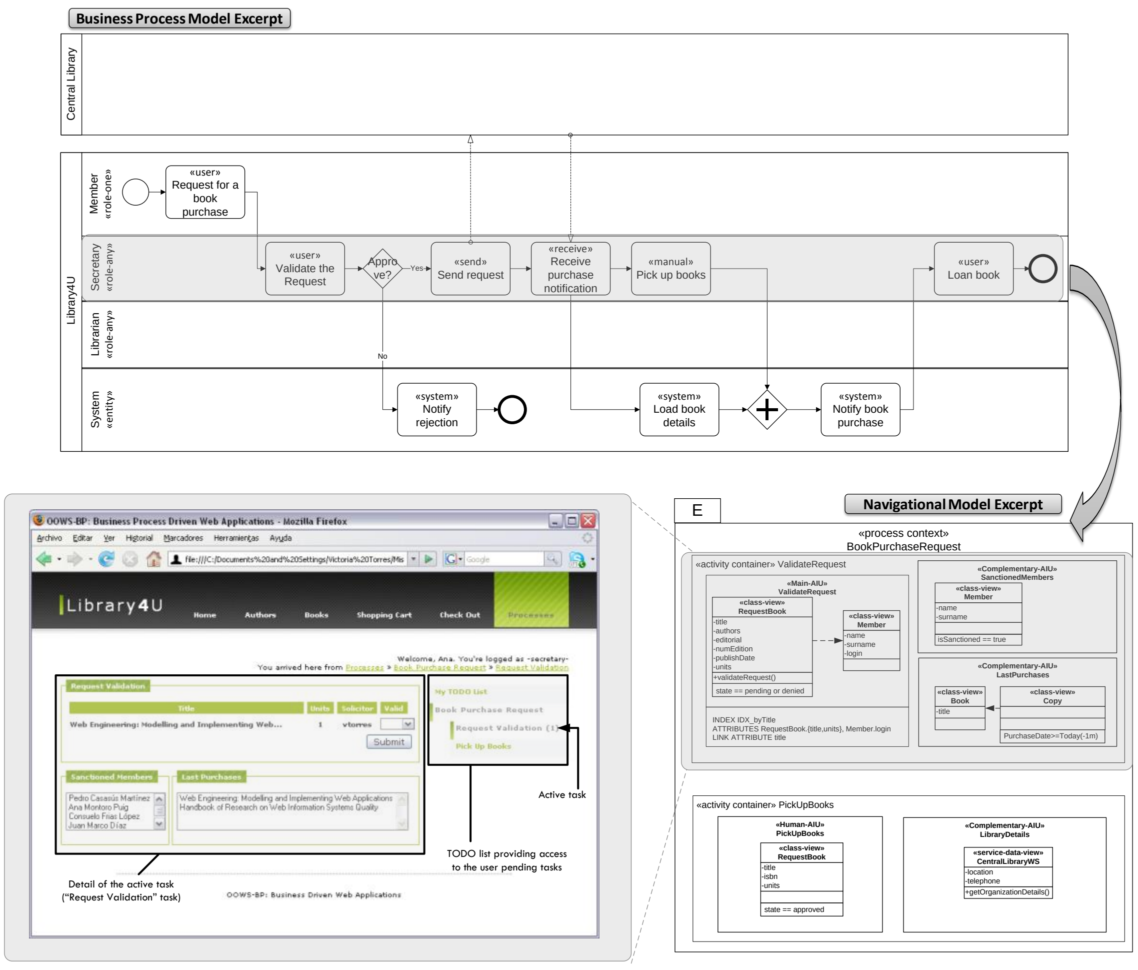

Business Process Model Excerpt
Central Library
Library4U
Member «role-one»
Secretary «role-any»
Librarian «role-any»
System «entity»
«user» Request for a book purchase
«user» Validate the Request
Approve?
Yes
No
«send» Send request
«receive» Receive purchase notification
«manual» Pick up books
«user» Loan book
«system» Notify rejection
«system» Load book details
«system» Notify book purchase
OOWS-BP: Business Process Driven Web Applications - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
file:///C:/Documents%20and%20Settings/Victoria%20Torres/Mis
Library4U
Home Authors Books Shopping Cart Check Out Processes
Welcome, Ana. You're logged as -secretary-
You arrived here from Processes » Book Purchase Request » Request Validation
Request Validation
Title Units Solicitor Valid
Web Engineering: Modelling and Implementing Web... 1 vtorres
Submit
Sanctioned Members
Pedro Casasús Martínez
Ana Montoro Puig
Consuelo Frías López
Juan Marco Díaz
Last Purchases
Web Engineering: Modelling and Implementing Web Applications
Handbook of Research on Web Information Systems Quality
My TODO List
Book Purchase Request
Request Validation (1)
Pick Up Books
Active task
TODO list providing access to the user pending tasks
Detail of the active task ("Request Validation" task)
OOWS-BP: Business Driven Web Applications
E
Navigational Model Excerpt
«process context» BookPurchaseRequest
«activity container» ValidateRequest
«Main-AIU» ValidateRequest
«class-view» RequestBook
-title
-authors
-editorial
-numEdition
-publishDate
-units
+validateRequest()
state == pending or denied
«class-view» Member
-name
-surname
-login
INDEX IDX_byTitle
ATTRIBUTES RequestBook.{title,units}, Member.login
LINK ATTRIBUTE title
«Complementary-AIU» SanctionedMembers
«class-view» Member
-name
-surname
isSanctioned == true
«Complementary-AIU» LastPurchases
«class-view» Book
-title
«class-view» Copy
PurchaseDate>=Today(-1m)
«activity container» PickUpBooks
«Human-AIU» PickUpBooks
«class-view» RequestBook
-title
-isbn
-units
state == approved
«Complementary-AIU» LibraryDetails
«service-data-view» CentralLibraryWS
-location
-telephone
+getOrganizationDetails()

Figure 14 Web page corresponding to the *processes* section

Similarly to the short-running process, we generate the *navigational model* that is necessary to support BP tasks. Then, we generate the web page that implements this part of the *navigational model*.

7.8. BP instance state

Similar to the introduction of database management systems into the architecture of software systems, dealing with systems that support the execution of BPs requires the introduction of solutions that allow BPs to be properly managed.

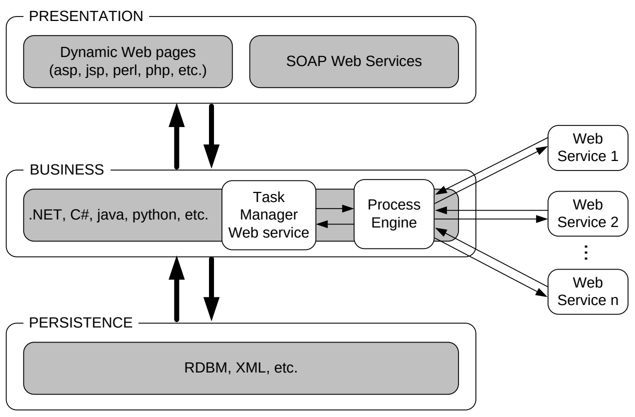


Figure 15 Three-layer architecture for BP-driven web applications

Based on the decision of adopting WS-BPEL as the executable language of processes, we have introduced a process engine that implements this specification into the architecture of the generated web applications. Specifically, we have made use of the ActiveBPEL[17] process engine. However, since WS-BPEL is based on web services, when we want to model workflows (processes that include human participants), we have to make use of some mechanisms on top of the original specification to allow us to handle the asynchrony introduced by process participants of this kind.

As Figure 15 shows, the proposed architecture follows the classical three-layer architecture. From these three layers, we are going to focus on the business layer, which is the layer where the extension has been introduced. This extension includes two new elements, the *process engine* and the *task manager web service*. The function of the *process engine* is to create and run new process instances from input WS-BPEL processes when an incoming message triggers the start process activity. Moreover, since WS-BPEL is based on web services, and these can be hosted on different servers, the business layer can be distributed with their components being linked by the process engine.

[17] http://www.active-endpoints.com/active-bpel-engine-overview.htm

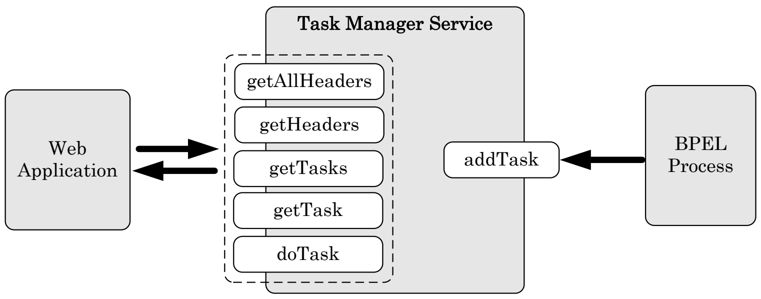
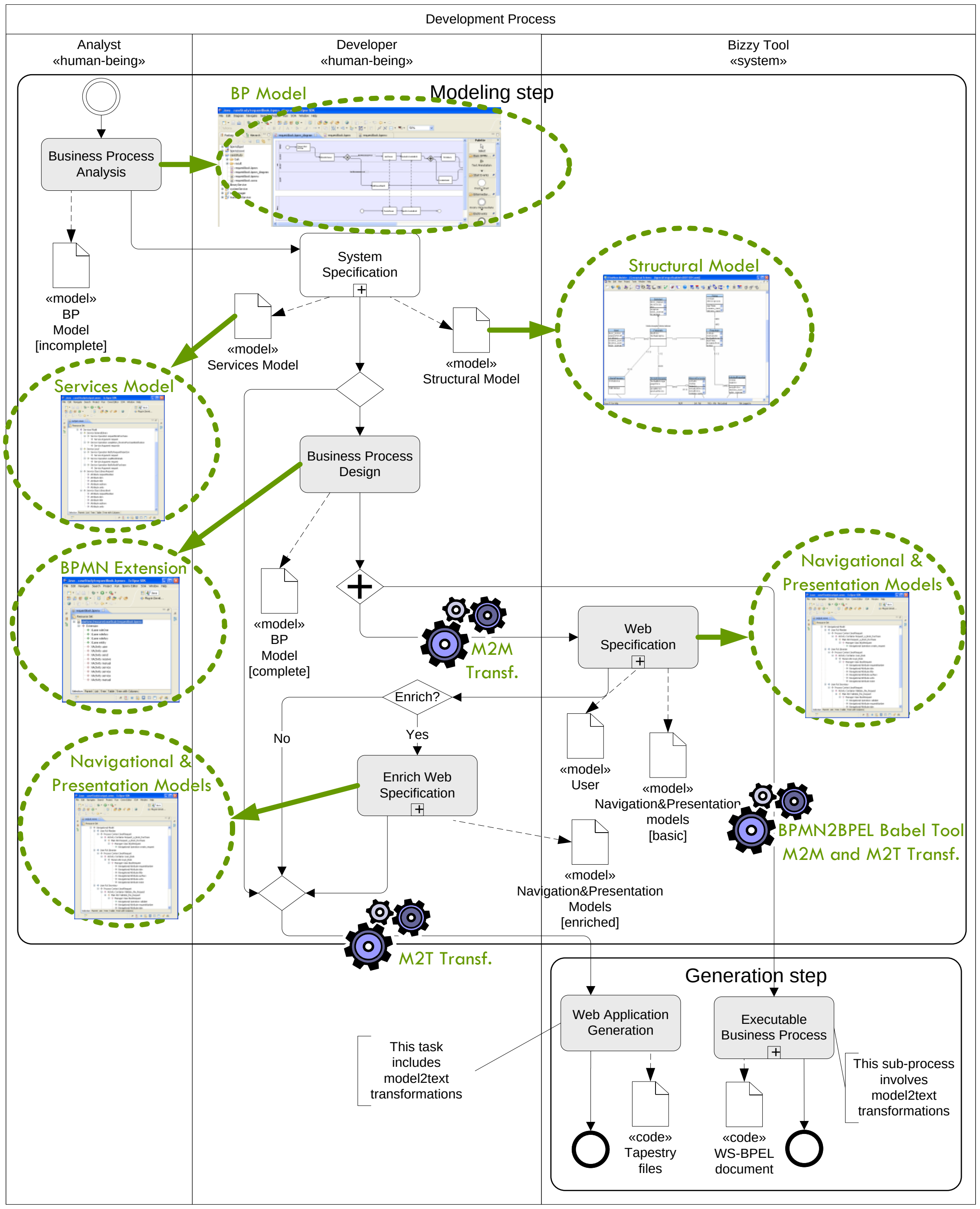


Figure 16 Task manager service interface

The *task manager web service* (Figure 16) is the element that comes into play when activities related to humans are invoked. This service takes responsibility for handling activities of this type and communicates with the process engine and the application logic. As Figure 16 shows, the *task manager service* has six operations to interact with both the WS-BPEL process and the web application.

8. Tool support

The generation of BP-driven web applications is performed by using different model editors and model transformations throughout the different steps defined in the development process. According to the development process presented in section 4, this process is divided into two main steps that relate to *system specification* and *system generation*. The coordinated use of these model editors and model transformations allows us to generate the web application according to the models specified at the problem space. Figure 17 shows the tool support provided for each of the steps defined in the process. In this figure, when a model transformation is performed, this has been indicated by means of a gear image, which also includes the type of transformation performed (model-to-model, model-to-text or both).



Development Process
Analyst «human-being»
Developer «human-being»
Bizzy Tool «system»
BP Model
Modeling step
Business Process Analysis
System Specification
«model» BP Model [incomplete]
«model» Services Model
«model» Structural Model
Structural Model
Services Model
Business Process Design
BPMN Extension
«model» BP Model [complete]
M2M Transf.
Web Specification
Navigational & Presentation Models
Enrich?
No
Yes
Enrich Web Specification
«model» User
«model» Navigation&Presentation models [basic]
BPMN2BPEL Babel Tool M2M and M2T Transf.
«model» Navigation&Presentation Models [enriched]
Navigational & Presentation Models
M2T Transf.
Generation step
Web Application Generation
Executable Business Process
This task includes model2text transformations
This sub-process involves model2text transformations
«code» Tapestry files
«code» WS-BPEL document

Figure 17 Tool support for the BP-driven web application development process

The model editors and transformations used throughout the development process have been built from a set of tools included in the Eclipse project, specifically from the Eclipse Modeling Framework[18] (EMF). EMF includes tools for the generation, edition, and serialization of models conforming to Ecore metamodels (an implementation of the OMG's Essential MOF to represent metamodels). In the *Bizzy tool*, all the metamodels

[18] http://www.eclipse.org/modeling/emf/

used (structural, navigational, presentation, BPMN extended and WS-BPEL) are represented as Ecore metamodels. In some cases, the Ecore metamodels were built from the corresponding XML Schemas (EMF permits the generation of Ecore metamodels from XML Schemas). In other cases, a new Ecore metamodel had to be built.

Most of the model editors included in the *Bizzy tool* are provided as a tree-based EMF editor. The only graphical editor included in the tool is the BPMN Modeller[19], which was developed in the SOA Tools Platform (STP).

Two different languages have been used for model transformations. The Atlas Transformation Language[20] (ATL) was used to deal with model-to-model transformations. Specifically, the transformations that have been implemented in this language allow transforming: (1) the BP model into the corresponding *navigational model* and (2) the *BP model* into the BP format accepted by the BPMN2BPEL BABEL tool. The MOFScript language was used to deal with model-to-text transformations. Specifically, the transformations that have been implemented in this language allow generating: (1) the interface (WSDL file) and the data types (XSD file) used by the web service represented by the WS-BPEL BP; and (2) the web applications in terms of the Tapestry web framework.

In addition to these Eclipse-based model editors and transformations, a Java tool has been used to perform the step that partially builds the executable WS-BPEL document. This tool is the BABEL BPMN2BPEL[21] tool. Its role is to perform the transformation between BPMN diagrams into WS-BPEL definitions.

9. Validation of the proposal

Since its conception, the methodology proposed in this work is being validated in different settings. Initially, we made some internal developments at the Department of Computer Science at the Technical University of Valencia. Then, when the methodology and the tool supporting it proved to be adequate for the development of BP-driven web applications we initiated collaboration with the Valencian Regional

[19] http://www.eclipse.org/stp/bpmn/
[20] http://www.eclipse.org/m2m/atl/
[21] http://www.bpm.fit.qut.edu.au/projects/babel/tools/

Ministry of Infrastructure and Transport and also with the Technical University of Valencia for different purposes. The details of these settings are the following:

1. Department of Computer Science at the Technical University of Valencia[22]. In this case, we developed several case studies whose main goal was the management of daily activities of the department. Examples of the developed studies are the *checkout process* and the *book purchase request process* presented in section 2. The people involved in the development of these case studies were people with an academic profile (teachers, researchers and university students) with some background in WE and specifically in the OOWS approach. After the success obtained in these developments we began a project within the university to carry out the development of the web applications of all the departments of the university. At this moment we are still working on the specification of the system. However, we have already perceived a greater acceptance by all the stakeholders involved in the project.
2. Valencian Regional Ministry of Infrastructure and Transport[23]. In this context, the ideas of the developed methodology were applied to put into practice some parts of gvMétrica, an adaptation of METRICA III to satisfy the needs of the regional ministry. Some of the results of this work can be found in [34]. The working team involved in this project was made up by business analysts (personnel from the Organization Department) and computer science engineers (personnel from the IT Department). Currently, some of the extensions defined in our methodology are already implemented in the Modelling Software Kit (MOSKitt) tool, which is an open source CASE tool built on the Eclipse [4] platform to give support to the gvMétrica methodology.

Besides these settings, the methodology has been taken as basis for the development of applications for the Internet of Things [15]. In this case, the methodology was extended to deal with the particularities introduced by physical mobile workflows [16]. These particularities mainly refer to the integration of real-world objects in business processes, which involves handling the broad heterogeneity of technologies to bridge physical and digital worlds.

[22] http://www.dsic.upv.es
[23] http://www.moskitt.org

On the basis of these settings, the main conclusion must be that the presented methodology, with all its contributions, appears to have a positive effect not only during the development process but also with the generated applications. For instance, in the context of the regional ministry where several human participants are involved in the processes, the implementation given for long-running processes allows the participants to handle multiple instances of the same task at the same time (by means of the to-do metaphor). On the contrary, in the context of mobile workflows, human participants are involved in more occasional and dynamic processes where a step-by-step strategy is better suited. In addition, the development of the *Bizzy tool* proved the importance of having a tool to support most of the methodology. In fact, some of the aspects that were out of the scope of the tool such as the automatic development of adaptors for services had a bad influence on the application of the methodology.

10. Conclusions and further work

In this work, we have presented a complete approach to carry out the development of BP-driven web applications. This approach extends from the modelling phase (the phase where the system is represented in terms of a set of models) through the generation phase (the phase that applies a set of transformation rules to obtain the executable artefacts). At the modelling level, we have defined a set of abstractions that represent navigational properties found during the business process (BP) execution. We have also modified the architecture of the generated web applications in order to properly handle BPs. As a result, we have introduced a process engine into the architecture of these systems. This process engine allows the construction of more lightweight *navigational models*, where the process flow is maintained inside the process definition. Based on the MDE, we have defined a set of model transformations to obtain: (1) executable process definitions expressed in the WS-BPEL language; and (2) the set of files (.java, .html and .page) necessary to deploy an application in terms of a Web framework, specifically the Tapestry web framework.

Finally, we have developed a tool (the Bizzy tool) that implements the ideas presented in this work. The tool has been developed applying the latest trends in the MDE field. It has been built using tools that are included in the Eclipse development environment (the Eclipse Modelling project and the SOA Tool Platform project) and the BPMN2BPEL

Java tool. Tools such as ATL and MOFScript have also been used to implement the transformations defined in the proposal, and EMF has been used to manipulate the models defined in the method. Finally, the BPMN editor from the STP Project has been used to model the BPs defined in the BP model that included in the proposal.

We have applied our proposal to several case studies which require the support of business processes in the context of the management of the following: a university library, a department incident and even to pervasive environments. These case studies have allowed us to validate the viability of our proposal.

As further work, we are interested in considering how to deal with the variability that is observable in BPs. BPs primarily define the set of tasks that have to be completed in order to achieve a specific goal. However, the way in which tasks are performed depends on the context in which they are executed. In some cases, the number of alternatives can get so high that BP specifications become illegible and difficult to maintain. For this reason, we want to explore mechanisms to properly handle BP variability at both the modelling level and the execution level.

References

1. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business process execution language for web services version 1.1 (May 2003)
2. Barna, P., Frasincar, F., Houben, G.J.: A workow-driven design of web information systems. In Wolber, D., Calder, N., Brooks, C., Ginige, A., eds.: ICWE, ACM (2006) 321-328
3. Bakshi, K., Karger, D.R.: Semantic web applications. Proceedings of the ISWC 2005 Workshop on End User Semantic Web Interaction (November 2005)
4. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process modeling in web applications. ACM Trans. Softw. Eng. Methodol. **15**(4) (2006) 360-409
5. Brambilla, M., Preciado, J.C., Trigueros, M.L., Sánchez-Figueroa, F.: Business process-based conceptual design of rich internet applications. In: ICWE. (2008) 155-161
6. Brambilla, M., Butti, S., Fraternali, P.: Webratio bpm: A tool for designing and deploying business processes on the web. In: ICWE. (2010) 415-429
7. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (webml): a modeling language for designing web sites. In: Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking, Amsterdam, The Netherlands, The Netherlands, North-Holland Publishing Co. (2000) 137-157
8. Davis, J.: Open Source SOA. Manning Publications Co. (2009)
9. Distante, D.: Reengineering legacy applications and web transactions: an extended version of the UWA transaction design model. PhD thesis, University of Lecce, Italy (2004)
10. Distante, D., Rossi, G., Canfora, G., Tilley, S.R.: A comprehensive design model for integrating business processes in web applications. Int. J. Web Eng. Technol. **3**(1) (2007) 43-72
11. Duhl, J.: Rich internet applications. Technical report, Technical report, IDC (November 2003)
12. Fons, J.: OOWS: A Model Driven Method for the Development of Web Applications. PhD thesis, Universidad Politécnica de Valencia(2008)

13. Fons, J., Pelechano, V., Pastor, O., Valderas, P., Torres, V.: Applying the OOWS Model-Driven Approach for Developing Web Applications. The Internet Movie Database Case Study. Human-Computer Interaction Series. In: Web Engineering: Modelling and Implementing Web Applications. Springer London (2008) 65-108
14. Fowler, M.: Inversion of control containers and the dependency injection pattern. http://martinfowler.com/articles/injection.html (January 2004)
15. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American **291**(4) (October 2004) 76-81
16. Giner, P., Cetina, C., Fons, J., Pelechano, V.: Developing mobile business processes for the internet of things. IEEE Pervasive Computing **9** (2010) 18-26
17. Gómez, J., Cachero, C., Pastor, O.: Extending a conceptual modelling approach to web application design. In Wangler, B., Bergman, L., eds.: CAiSE. Volume 1789 of Lecture Notes in Computer Science., Springer (2000) 79-93
18. Goth, G.: The task-based interface: Not your father's desktop. IEEE Software **26**(6) (2009) 88-91
19. Holmes, T., Tran, H., Zdun, U., Dustdar, S.: Modeling human aspects of business processes - a view-based, model-driven approach. In: ECMDA-FA. (2008) 246-261
20. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W., eds.: Web Engineering – The Discipline of Systematic Development of Web Applications. John Wiley & Sons Ltd., England (2006)
21. Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. PhD thesis, Ludwig-Maximilians-University Munich, Germany (2001)
22. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of business processes in web application models. J. Web Eng. **3**(1) (2004) 22-49
23. Limbourg, Q., Vanderdonckt, J.: Usixml: A user interface description language supporting multiple levels of independence. In: ICWE Workshops. (2004) 325-338
24. Linaje, M., Preciado, J.C., Sánchez-Figueroa, F.: Engineering rich internet application user interfaces over legacy web models. IEEE Internet Computing **11**(6) (2007) 53-59
25. Link, S., Hoyer, P., Schuster, T., Abeck, S.: Model-driven development of human tasks for workflows. In: ICSEA '08: Proceedings of the 2008 Third International Conference on Software Engineering Advances, Washington, DC, USA, IEEE Computer Society (2008) 329-335
26. Marcos, E., Cáceres, P., Castro, V. D.: An approach for navigation model construction from the use cases model. CAiSE Forum. Held in conjunction with the 16th Conference On Advanced Information Systems Engineering (June 2004)
27. Business process modeling notation (BPMN). OMG final adopted specification. dtc/06-02-01 (February 2006)
28. Pietschmann, S., Voigt, M., Meissner, K.: Adaptive rich user interfaces for human interaction in business processes. In: Proceedings of the 10th International Conference on Web Information Systems Engineering (WISE 2009), WISE, Springer LNCS (October 2009) 351-364
29. Schwabe, D., Rossi, G.: An object oriented approach to web-based applications design. Theor. Pract. Object Syst. **4**(4) (October 1998) 207-225
30. Schmid, H.A., Rossi, G.: Modeling and designing processes in e-commerce applications. IEEE Internet Computing **8**(1) (2004) 19-27
31. Schwinger, W., Retschitzegger, W., Schauerhuber, A., Kappel, G., Wimmer, M., Pröll, B., Cachero, C., Casteleyn, S., Troyer, O.D., Fraternali, P., Garrig_os, I., Garzotto, F., Ginige, A., Houben, G.J., Koch, N., Moreno, N., Pastor, O., Paolini, P., Pelechano, V., Rossi, G., Schwabe, D., Tisi, M., Vallecillo, A., van der Sluijs, K., Zhang, G.: A survey on web modeling approaches for ubiquitous web applications. IJWIS **4**(3) (2008) 234-305
32. Sousa, K.S., Mendona, H., Vanderdonckt, J.: A model-driven approach to align business processes with user interfaces. J. UCS 14(19) (2008) 3236-3249
33. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S.: Model-driven approach for managing human interface design life cycle. In: MoDELS. (2007) 226-240
34. Tedre, M.: What should be automated? Interactions **15**(5) (2008) 47-49
35. Torres, V., Giner, P., Bonet, B., Pelechano, V.: Adapting BPMN to Public Administration. To appear in: Proceedings BPMN 2010 Springer's Lecture Notes in Business Information Processing (LNBIP). Postdam, Germany.
36. Troyer, O.D., Casteleyn, S.: Modeling complex processes for web applications using wsdm. In: Proceedings of the Third International Workshop on Web-Oriented Software Technologies (held in conjunction with ICWE2003), IWWOST2003. (2003)