Abstract. This paper presents WebRatio BPM, an Eclipse-based tool that supports the design and deployment of business processes. The tool applies Model Driven Engineering techniques to complex, multi-actor business processes, mixing tasks executed by humans and by machines. Business processes are described through the standard BPMN 1.2 notation, extended with information on task assignment and escalation policies, activity semantics, and typed dataflows, to enable a two-step generative approach: first the Process Model is automatically transformed into a Web Application Model in the WebML notation, which seamlessly express both human- and machine-executable tasks; secondly, the Application Model is fed to an automatic transformation capable of producing the code. The tool provides various features that increase the productivity and the quality of the resulting application: one-click generation of a running prototype of the process from the BPMN model; fine-grained refinement of the resulting application; support of continuous evolution of the application design after requirements changes (both at business process and at application levels).

1 Introduction

Business process languages, such as BPMN (Business Process Management Notation) [8], have become the de facto standard for enterprise-wide application specification, as they enable the implementation of complex, multi-party business processes, possibly spanning several users, roles, and heterogeneous distributed systems. Indeed, business process languages and execution environments ease the definition and enactment of the business constraints, by orchestrating the activities of the employees and the service executions.

This paper presents an integrated approach and a supporting toolsuite to the specification, design and implementation of complex, multi-party business processes, based on a Model-Driven Engineering (MDE) methodology and on code generation techniques capable of producing dynamic Web applications from platform independent models.

The proposed approach is a top down one: the (multi-actor, multi-site) business process is initially designed in an abstract manner, using the standard
BPMN notation for schematizing the process actors, tasks, and business constraints. The resulting BPMN model is an abstract representation of the business process and cannot be used directly for producing an executable application, because it lacks information on essential aspects of process enactment such as: task assignment to humans or to Web Services, data flows among tasks, service invocation and user interface logics. Therefore, the standard BPMN specification is manually annotated with the missing information, to obtain a detailed process model amenable to a two-step transformation:

- A first model-to-model transformation (Process to Application) translates the detailed process model into: 1) a platform-independent model of the Web user interface and of the Web Service orchestrations needed for enacting the process, expressed in a Domain Specific Language called WebML [2]; 2) a Process Metadata Model, representing the business constraints (e.g., BPMN precedence constraints, gateways, etc).

- A second model-to-text transformation (Application to Code) maps the Application Model and the Process Metadata Model into the running code of the application. The resulting application is runtime-free and runs on any standard Java Enterprise Edition platforms.

The original contributions of the paper are: (i) an MDE forward-engineering methodology that starts from a high-level process model and progressively transforms it into the running code of the application, comprising Web Service interaction logics and Web User Interface logics; (ii) a refinement of the BPMN notation for augmenting the semantics of the Process Model and enable effective model transformation; (iii) a two-step generative framework transforming the refined Process Model into an Application Model, and for generating the executable code from the Application Model; and (iv) an extended version of the WebRatio toolsuite [11], called WebRatio BPM, that fully implements all the steps of the presented methodology.

The paper is organized as follows: Section 2 discusses the background technologies and notations; Section 3 discusses the approach to application development; Section 4 and Section 5 illustrate the extended process model and the application model, respectively; Section 6 describes the implementation of the WebRatio BPM tool; Section 7 discusses the related work; and Section 8 draws the conclusions.

2 Background: BPMN, WebML, and WebRatio

This work builds upon existing methods and tools to cover the different design phases.

BPMN [8] supports the high level specification of service choreographies, and allows one to visually specify actors, tasks, and constraints involved in a business process. Precedence constraints are specified by arrows, representing the control flow of the application, and gateways, representing branching and merging points
of execution paths. Parallel executions, alternative branches, conditional executions, events, and message exchanges can be specified. BPMN allows analysts to describe complex orchestrations of activities, performed by both humans and machines. Figure 1 shows an example of BPMN, describing a simplified leasing process for houses and cars.

WebML [2] is a Domain Specific Language for data-, service-, and process-centric Web applications [1]. It allows specifying the conceptual model of Web applications built on top of a data schema and composed of one or more hypertexts used to publish or manipulate data. The data model can be specified through standard E-R or UML Class diagrams. Upon the same data model, different hypertext models (site views) can be defined (e.g., for different types of users or devices). A site view is a graph of pages, consisting of connected units, representing data publishing components. Units are related to each other through links, representing navigational paths and carrying parameters. WebML allows specifying also update operations on the underlying data (e.g., the creation, modification and deletion of instances of entities or relationships) or operations performing arbitrary actions (e.g. sending an e-mail, invoking a remote service [6], and so on). Figure 2 shows a simple site view containing two pages, respectively showing the list of houses and a form for searching cars available for leasing. Page Search Leasing Cars contains an entry unit for inputting the car model to be searched, a scroller unit, extracting the set of cars meeting the search condition and displaying a sequence of result blocks, and a multidata unit displaying the cars pertaining to a block of search results. Besides Web applications, WebML can be used to specify Web services, Web service orchestrations, and the consumption of Web services by Web applications.

WebML is supported by the WebRatio CASE tool [11], which allows the visual specification of data models and site views and the automatic generation
of J2EE code. The tool consists of a set of Eclipse plug-ins and takes advantage of all the features of this IDE framework. It also supports customized extensions to the models and code generators, model checking, testing support, project documentation, and requirements specifications.

The main features of WebRatio are the following: it provides an integrated MDE approach for the development of Web applications and Web services, empowered by model transformations able to produce the complete running code; it unifies all the design and development activities through a common interface based on Eclipse, which includes the visual editing of models, the definition of presentation aspects, and the extension of the IDE with new business components and code generation rules; it stores all the design artifacts into a common area (the Eclipse workspace) and manages them in a version control and collaborative work system; it can easily be extended with any existing Eclipse plugin (e.g., for Java component development, debugging, and so on).

3 Development Process

The development process supported by WebRatio BPM is structured in four main steps, represented in Figure 3.

Initially, business requirements are conceptualized in a high-level Business Process Model, which summarizes the needs of the stakeholders in a coarse BPMN schema. Figure 1 is an example of BPM that can be obtained as a requirement specification of the leasing application. Subsequently, the BPMN schema is refined by a BPM expert, who annotates it with parameters on the activities and data flows.

The resulting extended Process Model is subject to a first transformation, which produces the Application Model and Process Metadata Model. The Application Model (discussed in Section 5.2) specifies the details of the executable application according to the WebML notation, representing the hypertext interface for human-directed activities. The Process Metadata Model (discussed in Section 5.1) consists of a conceptual view (represented as a UML class diagram) of the activities of the process and of the associated constraints, useful for encapsulating the process control logic. This transformation extends and refines

Fig. 2. WebML hypertext model example.
the technique for model-driven design of Web applications from business process specification initially proposed in [1]. Subsequently, the generated Application Model can be either executed as-is to get a first flavour of the application execution, or it can be refined manually by the designer, to add domain-dependent information on the execution of activities. Finally, the refined Application Model is the input of a second transformation, which produces the code of the application for a specific technological platform (in our case, J2EE): this step is completely automated thanks to the code generation facilities included in WebRatio.

4 Refined Process Model

The high-level BPMN process model designed in the requirement specification phase is not detailed enough to allow the generation of the application code. Its refinement is done using an extended BPMN notation, which enables a more precise model transformation into a WebML Application Model and then into the implementation code. In particular, the process model is enriched with information about the data produced, updated and consumed by activities, which is expressed by typed activity parameters and typed data flows among activities. Furthermore, activities are annotated to express their implicit semantics, and gateways (i.e., choice points) that require human decisions are distinguished.

Figure 4 shows the graphical notation of the extended BPMN activity. An activity is associated with a Name (1), which is a textual description of its semantics, and possibly an Annotation (2), which describes the activity behaviour using an informal textual description. An activity is parametric, and has a (possibly empty) set of input (3) and output (4) parameters. The actual values of

Fig. 3. Development process overview.

Fig. 4. Extended activity notation.
input parameters can be assigned from preceding activities; the output parameters are produced or modified by the activity. Analogous extensions are defined for gateways; these are further refined by specifying whether they are implemented as manual or as automatic branching/merging points. Manual gateways (tagged as Type “M”) involve user interaction in the choice, while Automatic gateways (tagged as Type “A”) automatically evaluate some condition and decide how to proceed with the process flow without human intervention. The output flow of gateways can be associated to a guard condition, which is an OCL Boolean expression over the values of the parameters of the gateway; the semantics is that the activity target of the flow link with the guard condition can be executed only if the condition evaluates to true.

5 Application Model

Starting from the Detailed Process Model presented above, an automatic transformation produces: (1) Process Metadata Model, describing the process constraints in a declarative way as a set of relations; (2) the Domain Model, specifying the application-specific entities; (3) and the Application Model, including both the site views for the user interaction and the service views for Web service orchestration.

Hence, the transformation consists of two sub-transformations:

- Detailed Process Model to Process Metadata: the BPMN precedence constraints and gateways are transformed into instances of a relational representation compliant to the Process Metamodel shown in Figure 5, for enabling runtime control of the process advancement;
- Detailed Process Model to Application Model: the BPMN process model is mapped into a first-cut Application Model, which can be automatically transformed into a prototype of the process enactment application or subsequently refined by the designer to incorporate further domain specific aspects.

Thanks to the former transformation, the BPMN constraints, stored in the Process Metadata Model, are exploited by the components of the Application Model for executing the service invocation chains and enacting the precedences among human-executed tasks.

5.1 Process Metadata Model Generation

Figure 5 shows, as a UML class diagram, the schema of the metadata needed for executing an BPMN process at runtime.

A Process represents a whole BPMN diagram, and includes a list of Activities, each described by a set of input and output ParameterTypes. A Case is the instantiation of a process, and is related to the executed Activity Instances, with the respective actual Parameter Instances. The evolution of the status history
is registered through CaseLogEntry and ActivityLogEntry. Users are the actors that perform a task and are clustered into Groups, representing their roles.

The transformation from the extended BPMN to the Process Metadata is a relational encoding of the BPMN concepts: each process model is transformed to a Process instance; each activity is transformed into an Activity instance; each flow arrow is transformed into a nextActivity/previousActivity relationship instance; each guard condition is transformed into a Condition instance.

5.2 Application Model Generation

The transformation from Refined Process Models to WebML coarse models of services and hypertext interfaces considers the type (human or automatic) of the gateways and the information on the data flows. The application models produced by the transformation still need manual refinement, to add domain-specific elements that cannot be expressed even in the enriched BPMN notation. However, by exploiting information about the activity type, a first-cut application model can be generated, which needs reduced effort for manual refinement.

The computation of the next enabled activities given the current state of the workflow is encapsulated within a specific WebML component, called Next unit, which factors out the process control logic from the site view or service orchestration diagram: the Next unit exploits the information stored in the Process Metadata to determine the current process status and the enabled state transitions. It needs the following input parameters: caseID (the currently executed process instance ID), activityInstanceId (the current activity instance ID), and the conditionParameters (the values required by the conditions to be evaluated). Given the activityInstanceId of the last concluded activity, the Next
unit queries the Process Metadata objects to find all the process constraints that determine the next activity instances that are ready for activation. Based on the conditions that hold, the unit determines which of its output links to navigate, which triggers the start of the proper subsequent activities.

The Process to Application Model Transformation from BPMN to WebML consists of two main rules: the Process transformation rule, addressing the structure of the process in-the-large; and the Activity transformation rule, managing the aspects of individual activities: parameter passing, starting and closing, and behavior. For modularity and reusability, the piece of WebML specification generated for each activity is enclosed into a WebML module, a container construct analogous to UML packages.

Figure ?? shows an overview of the outcome of the Process transformation rule: the hypertext page for manually selecting the process to be started and for accessing the objects resulting from process termination. This WebML fragment models the process wrapping logic, generated from the Start Process and End Process BPMN events.

The generated WebML model further comprises: (1) the orchestration site view, that contains the logic for the process execution; (2) a site view or service view for each BPMN pool; (3) a set of hypertext pages for each human-driven BPMN activity; (4) one service invocation (triggering the suitable actions for application data updates) for each automatic activity.

Figure 6 shows the model of the orchestration site view. The enactment of the process is performed through a loop of WebML module invocations, each representing the implementation of one of the activities.

The Activity transformation rule is based on the BPMN activity and gateway specifications, taking into account aspects like the actor enacting the activity (e.g., a human user or the system). For each BPMN activity and gateway, a
WebML module implementing the required behavior is generated. Each generated module has a standard structure: an input collector gathers the parameters coming from previous activities; the activity business logic part comprises a form with fields corresponding to the output of the activity and a Create unit that stores the information produced by the activity persistently, for subsequent use. For gateways, the transformation rule behaves according to the BPMN semantics and to the kind of executor assigned to the gateway (human or automatic): if the gateway is tagged as human-driven, a hypertext is generated for allowing the user to choose how to proceed; if the gateway is tagged as automatic, the choice condition is embedded in the application logic. The transformation of BPMN gateways is conducted as follows:

- **AND-splits** allow a single thread to split into two or more parallel threads, which proceed autonomously. The WebML model for AND-split automatic execution generates a set of separate threads that launch the respective subsequent activity modules in parallel, while manual execution allows the user to select and activate all the possible branches.

- **XOR-splits** represent a decision point among several mutually exclusive branches. Automatic XOR-splits comprise a condition that is automatically evaluated for activating one branch, while manual XOR-splits allow the user to choose one and only one branch.

- **OR-splits** represent a decision for executing one or more branches. Automatic OR-splits comprise a condition that is automatically evaluated for activating one or more branches, while the manual version allows the user to choose the branches to activate.

- **AND-joins** specify that an activity can start if and only if all the incoming branches are completed. This behavior is usually implemented as automatic.

- **XOR-joins** specify that the execution of a subsequent activity can start as soon as one activity among the incoming branches has been terminated. This behavior is usually implemented as automatic.

- **OR-joins** specify that the execution of the subsequent activity can start as soon as all the started incoming branches have been terminated. This behavior is usually implemented as automatic, possibly through custom conditions on the outcome of the incoming branches.

Figure 7 shows two simplified examples of generated modules: the **XOR (ProductType)** module (Figure 7.a) implements the automatic evaluation of the XOR gateway in the BPMN model of Figure 1: given the ProductID, it extracts its type and checks whether it is a car or a house. The next activity to be performed is set accordingly, and this information is passed to the Next unit in the orchestration site view. The **Customer Credit Score** module in Figure 7.b shows the generated hypertext page that allows the user to enter and store the credit score value for the customer, which is the output parameter of the **Customer Credit Score** activity of Figure 1.
6 Implementation of WebRatio BPM

The illustrated method has been implemented as a new major release of WebRatio, called WebRatio BPM. To achieve this result, all three major components of the tool suite have been extended: the model editing GUI, the code generator, and the runtime libraries. The **model editing GUI** has been extended by: 1) creating an Eclipse-based workflow editor supporting the definition of the refined BPMN Process Model; and 2) adding the Next unit as a new component available in the WebML Application Model editor. The **code generator** has been extended in two directions: 1) the BPMN to WebML transformation has been integrated within the toolsuite, thus allowing automatic generation of the WebML Application Models and of the Process Metadata. 2) the code generation from WebML has been augmented to produce the instances of the Process Metadata and to integrate the novel components (e.g., the Next unit) into the existing J2EE code generation rules.

Moreover, a one-click publishing function has been added to the BPMN editor, thus allowing the immediate generation of a rapid prototype of the BPMN process. The prototype is a J2EE dynamic, multi-actor application with a default look & feel, produced from the WebML Application Models automatically derived from the BPMN diagrams, according to the previously described techniques. The process prototype comprises a few exemplary users for each BPMN actor, and allows the analyst to impersonate each role in the process, start a process and make it progress by enacting activities and both manual and automatic gateways. Figure 8 shows a snapshot of the user interface of the WebRatio BPMN editor.

The WebRatio BPM tool is being tested in a real industrial scenario of a major European bank, that needs to reshape its entire software architecture according to a BPM paradigm with a viable and sustainable design approach.
The first set of developed applications addressed the leasing department. The running case presented in this paper is inspired by the leasing application that is under development. The real application involves more than 80 business processes, which orchestrate more than 500 different activities.

7 Related Work

A plethora of tools exist for business process modeling and execution, produced by major software vendors, open source projects, or small companies. In our review of existing tools, we identified more than fifty relevant tools in the field. A report from Gartner [4] describes the magic quadrant of the field and selects the most promising solutions. Among them, we can mention Lombardi Teamworks, Intalio, webMethods BPMS, Tibco iProcess, Savvion BusinessManager, Adobe Livecycle ES, Oracle BPM Suite, IBM WebSphere Dynamic Process Edition. Most of them rely on Eclipse as IDE environment and include a visual designer of business models and a generator of configurations for associated workflow engines. The main innovations of our approach with respect to the competitors are: (1) quick generation of a running prototype; (2) possibility of refinement of the prototype at the modeling level; (3) support of the final application generation through MDD.

In the scientific community, some other works have addressed the challenge of binding the business processing modeling techniques with MDD approaches addressing Web applications development. Koch et al. [5] approach the integration of process and navigation modeling in the context of UWE and OO-H. Also Araneus [7] has been extended with a workflow conceptual model, allowing
the interaction between the hypertext and an underlying workflow management system. In OOHDM [9], the content and navigation models are extended with activity entities and activity nodes respectively, represented by UML primitives. Torres and Pelechano [10] combine BPM and OOWS [3] to model process-centric applications; model-to-model transformations are used to generate the Navigational Model from the BPM definition and model-to-text transformations can produce an executable process definition in WS-BPEL.

8 Conclusion

This paper presented a methodology and a tool called WebRatio BPM for supporting top-down, model-driven design of business-process based Web applications. The tool is now available for testing purposes and will be commercially distributed starting from October 2009. Ongoing and future works include evaluation of productivity of the developers and of quality of the implemented applications, and coverage of further aspects of BPMN semantics (i.e., customized events and exceptions). The tool, albeit still in a pre-beta status, is being used in a large banking application and therefore we expect to collect useful user feedbacks, new requirements, and (quantitative) quality and productivity metrics for the proposed approach in the immediate future.

References