WebRatio BPM: A Tool for Designing and Deploying Business Processes on the Web

Marco Brambilla[1], Stefano Butti[2], and Piero Fraternali[1]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione
P.za L. Da Vinci, 32. I-20133 Milano - Italy
{marco.brambilla,piero.fraternali}@polimi.it
[2] Web Models S.r.l., I-22100 Como - Italy
stefano.butti@webratio.com

Abstract. This paper presents WebRatio BPM, an Eclipse-based tool that supports the design and deployment of business processes as Web applications. The tool applies Model Driven Engineering techniques to complex, multi-actor business processes, mixing tasks executed by humans and by machines, and produces a Web application running prototype that implements the specified process. Business processes are described through the standard BPMN notation, extended with information on task assignment, 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 expresses both human- and machine-executable tasks; secondly, the Application Model is fed to an automatic transformation capable of producing the running 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) [13], 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 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 [4]; 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) a two-step generative framework comprising a first model transformation from a detailed Business Process Model to an Application Model and a second transformation for producing the executable code from the Application Model; and (ii) an extended version of the WebRatio toolsuite [17], called WebRatio BPM, that fully implements the proposed transformation steps. The tool is currently in beta version and will be released in the second quarter of 2010. However, a major European banking customer is already adopting WebRatio BPM for the development of a large, multi-country and multi-user business process based portal. Therefore, the validation of the approach is already ongoing and several lessons learned have been collected.

The advantages of having a two steps modeling process are multifold: (i) the BP model (BPMN) and the web model (WebML) allows the designer to separate the different concerns in the design, keeping the process issues separate from the hypertext and interface issues; (ii) the transformation of the BP model to a Web-specific model allows fine-grained description of the interfaces, while remaining at a modeling level; (iii) the Web modeling level allows seamless integration of resulting applications within non-BP-based application models (e.g., web portals, B2C e-commerce sites, and so on); (iv) having distinct models allows different user roles (i.e., business analysts at the BP level and Web engineers at the Web modeling level) to work together and independently at the same application design. These advantages, together with the one-click deployment option, make our proposal unique also considering the plethora of BPM tools existing on the market.

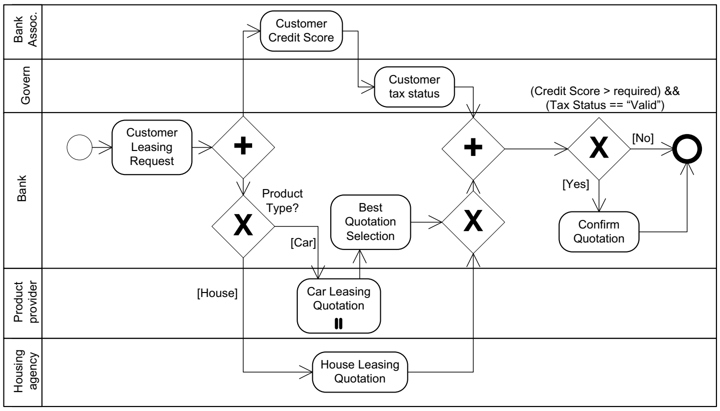

Fig. 1. Business process model of the leasing running example

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 [13] supports the specification of business processes, allowing one to visually specify actors, tasks, and constraints involved. 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 [4] is a Domain Specific Language for data-, service-, and process-centric Web applications [3]. 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

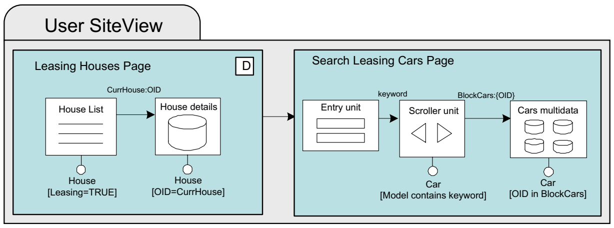

Fig. 2. WebML hypertext model example

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 [9], 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.

WebML is supported by the WebRatio CASE tool [17], 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 includes a unified workspace for projects and a version control and collaborative work environment.

3 Development Process

The development process supported by WebRatio BPM is structured in five main steps, represented in Figure 3 according to the SPEM notation [12].

Initially, business requirements are conceptualized in a *coarse Business Process Model* by the business analyst. 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 BPMN designer, who annotates it with parameters on the activities and data flows.

The resulting *refined Process Model* is subject to a first model transformation, which produces the WebML *Application Model* and *Process Metadata Model.*

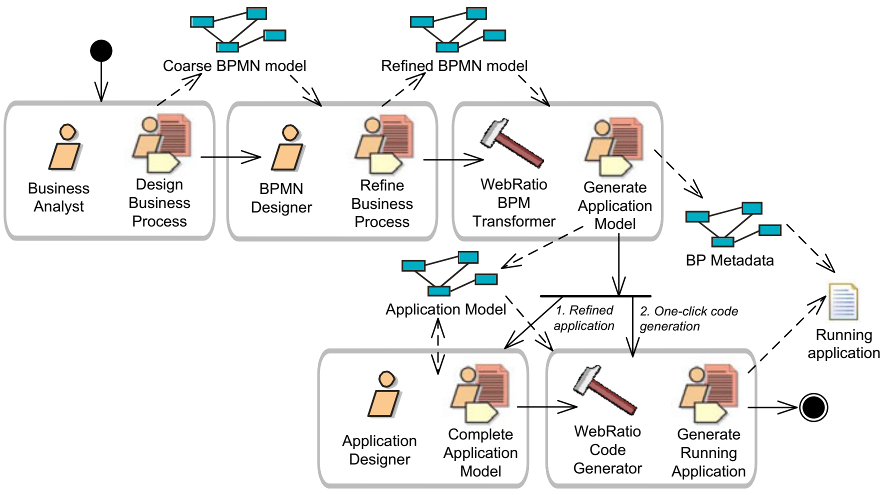

Fig. 3. Development process overview (SPEM notation)

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 relational data describing of the activities of the process and of the associated constraints, useful for encapsulating the process control logic. This transformation extends and refines the technique for model-driven design of Web applications from business process specification initially proposed in [3]. Subsequently, the generated Application Model can be either used as is by a one-click prototype generation transformation to get a first flavour of the application execution (option 2 in the branching point), or it can be refined manually by the application designer, to add domain-dependent information on the execution of activities (option 1 in the branching).

Finally, the 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.

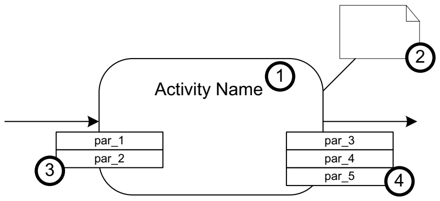

Fig. 4. Extended activity notation

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 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 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.

Notice that the diagram spans two modeling levels in the same data schema, namely process model and process instance information. The BPMN part is confined to the entities in the upper part of the figure, while the lower part regards execution data.

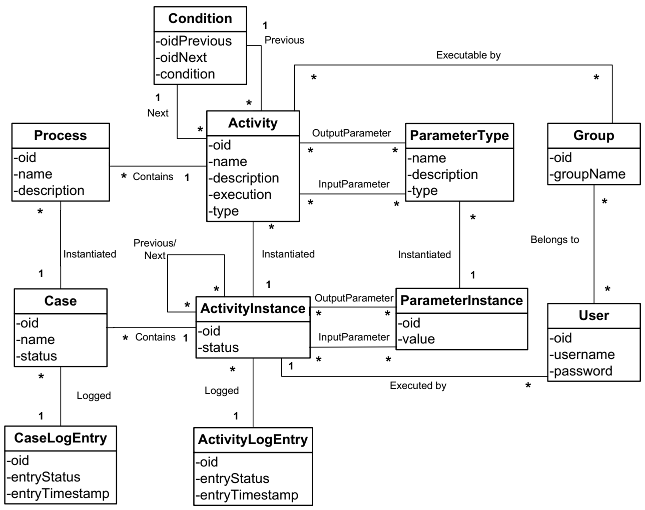

Fig. 5. Process Metadata describing the BPMN constraints

The transformation from the extended BPMN to the Process Metadata produces 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.

Process Metadata generation has been formalized as an ATL transformation from the BPDM metamodel to the Process Model of Figure 5.

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 6 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 process 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.

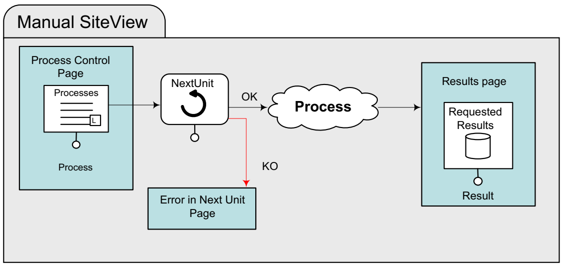

Fig. 6. Excerpt of a WebML application model generated from a BPMN model

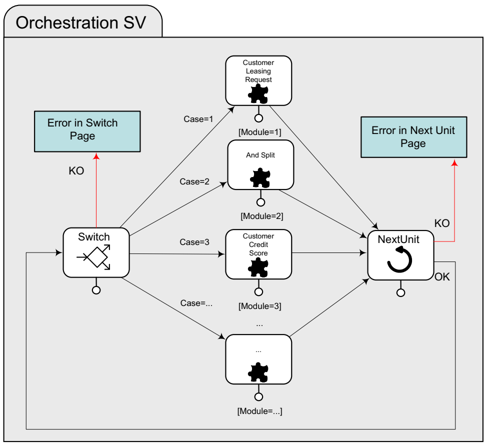

Fig. 7. WebML Orchestration Siteview

Figure 7 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. At the beginning, the initiation logic (shown in Figure 6) invokes the first module in the loop. The invoked module, be it a Web service call or a Web interface for the user to perform the activity, upon termination returns the control to the Next unit, which determines the modules to be executed next.

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 8 shows two simplified examples of generated modules: the *XOR (ProductType)* module (Figure 8.a) implements the automatic evaluation of the XOR

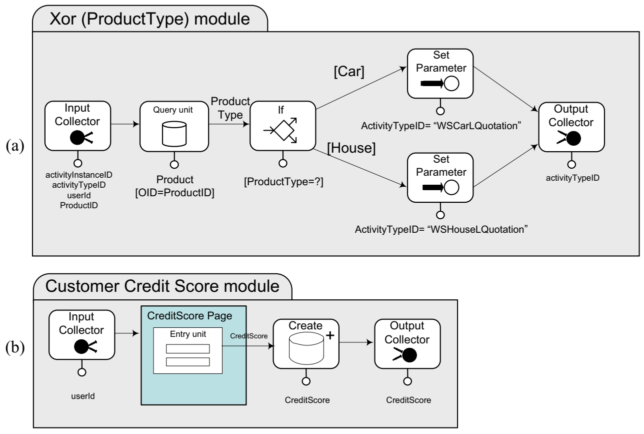

Fig. 8. WebML Modules for XOR gateway and Customer Credit Score

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 8.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.

The whole approach is specified by an ATL transformation organized into the three above specified rules: a *Process transformation rule* generates the process actions and then invokes the *Activity rule* that manages untyped activities. A set of *type-specific Activity rules* inherit from the general transformation and refine it.

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 9 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 that 500 different activities.

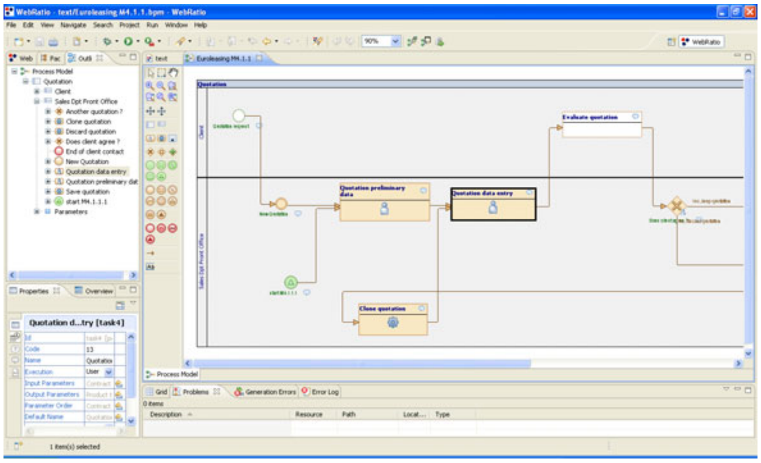

Fig. 9. WebRatio BPM user interface

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 [6] 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. In our analysis, we considered more than 50 tools: each of them exposes different strengths and weaknesses. Some 50% of them adopt BPMN as modeling notation; a few provide quick prototyping features (e.g., Oracle, Tibco, BizAgi), while only one provides fine grained simulation capabilities, i.e., the prossibility of visualizing hypothetical executions over the visual model, considering stochastic properties of the executions themselves (IBM); some of them are also very strong on BAM features (business analysis and monitoring), such as Oracle and BizAgi; owever, the ones that provide a good level of personalization of the user interfaces allow to do so only at the code level. The main innovations of our approach with respect to the competitors are: (1) quick generation of a running prototype with no coding required; (2) possibility of refinement of the prototype at the web modeling level; (3) clear separation of concerns and assignment to design tasks to roles; (4) model transformation and code generation of the final Web application 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.

In this work we focus on process- and data-centric application design, a field where several MDD-based approaches has proven valid. The challenge, though, is to define methods for effectively binding the business processing modeling techniques with MDD approaches addressing, for instance, Web applications development. The Process Modeling language (PML) [11], for instance, is an early proposal for the automatic generation of simple Web-based applications starting from imperative syntax, that allows users to enact their participation to the process .

Koch et al. [7] approach the integration of process and navigation modeling in the context of UWE and OO-H. The convergence between the two models is limited to the requirement analysis phase, while the design of the application model, is separated. In our work, both aspects are considered: like in UWE, we preserve the *process model* as an additional domain model in the application data; as in OO-H, we provide semi-automatic generation of WebML navigational model skeletons directly from the process model. Among the other existing models, we can mention Araneus [10], that has been extended with a workflow conceptual model, allowing the interaction between the hypertext and an underlying workflow management system. In OOHDM [14], the content and navigation models are extended with activity entities and activity nodes respectively, represented by UML primitives. In WSDM [16], the process design is driven by the user requirements and is based on the ConcurTaskTrees notation. An alternative approach is proposed by Torres and Pelechano [15], where BPM and OOWS [5] combines 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. Liew at al. [8] presents a set of transformations for automatically generating a set of UML artifacts from BPM.

With respect to our previous work, this paper extends and refines the technique initially proposed in [3] with several major aspects. The main innovation point is that the BP model and the application model are now treated at the same level and can be evolved separately, thanks to the topology of the generated application models, which insulates the process control logic from the interface and navigation logic of the front-end. Specifically, while in our previous proposal the BPM constructs were transformed into control flows in the application model (e.g., as links in the WebML hypertexts), practical use demonstrated that this approach led to severe difficulties during process and application maintenance and evolution; therefore, we had to better encapsulate process with the help of the process metadata.

Another related research area is the specification and deployment of service orchestrations [1] (e.g., as WS-BPEL specifications). These approaches lack management of the user interactions and of results presentation.

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. Our approach is based on model transformations and allows designers to produce applications (both as early prototypes and refined products) without coding. Thanks to the two different modeling levels (business model and Web hypertext model), the needs and skills of different design roles can be accommodated, thus allowing easy joint work between business analysts and web engineers. The high-level BPM perspective provides easy specifications of business models and quick generation of running prototypes, while the hypertext model covers the need of refined design of the final application, thus provide separation of concerns.

The tool, albeit still in a beta status, is have been used in a large banking application for 6 months now. In this period, we collected useful user feedbacks and new requirements, that were considered in the refinement of the system. The experiment was applied on large scale on a real industrial application:

– three different user roles worked together on the same large project: 3 business analysts, 6 application developers, and a few key customers interacted in the definition of the problems and of the solutions;
– the users were spread across Europe and across 4 different companies (the business consultancy company, the application development company, the banking customer, and WebRatio itself);
– the size and volume of the subprojects was so big that it challenged the code generation performances, bringing to applications that included more than 100,000 software components and XML configuration descriptors.

Although the size and the complexity of the project was so large, the need raised only for refinements and small fixes to the approach, which therefore proved valid. Two big requirements were collected on the field: the need for a BAM (Business Analysis and Monitoring) console associated to the approach and for a refined support to requirements and process changes. The BAM console design task will be simplified by the possibility of building the console itself with the model-driven WebML approach;being specified through models, the console will be configurable and customizable at will depending on the customer needs. The support to requirements and process changes is crucial in process-based applications, since the evolution of processes must be supported even when the application is in use, and therefore several process instances can be ongoing while the process change is applied. This requires to preserve all the versions of process models (and of associated applications) at the same time, to grant correct execution of both ongoing and new processes.

Further tasks will include quantitative 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).

References

1. Benatallah, B., Sheng, Q.Z.: Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. Distrib. Parallel Databases 17(1), 5–37 (2005)
2. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. ACM TOSEM 15(4), 360–409 (2006)
3. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. ACM TOSEM 15(4), 360–409 (2006)
4. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc., San Francisco (2002)
5. Fons, J., Pelechano, V., Albert, M., Pastor, O.: Development of web applications from web enhanced conceptual schemas. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 232–245. Springer, Heidelberg (2003)
6. Gartner. Magic quadrant for business process management suites. Technical report, Gartner (February 2009)
7. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of business processes in web application models. J. Web Eng. 3(1), 22–49 (2004)
8. Liew, P., Kontogiannis, K., Tong, T.: A framework for business model driven development. In: STEP '04: Software Tech. and Engineering Practice, pp. 47–56. IEEE, Los Alamitos (2004)
9. Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications. ACM Transactions on Internet Technologies (TOIT) 5(3), 439–479 (2005)
10. Merialdo, P., Atzeni, P., Mecca, G.: Design and development of data-intensive web sites: The Araneus approach. ACM Trans. Internet Techn. 3(1), 49–92 (2003)
11. Noll, J., Scacchi, W.: Specifying process-oriented hypertext for organizational computing. J. Netw. Comput. Appl. 24(1), 39–61 (2001)
12. OMG. Spem - software process engineering meta-model, version 2.0. Technical report (2008), http://www.omg.org/technology/documents/formal/spem.htm
13. OMG, BPMI. BPMN 1.2: Final Specification. Technical report (2009), http://www.bpmn.org/
14. Schmid, H.A., Rossi, G.: Modeling and designing processes in e-commerce applications. IEEE Internet Computing 8(1), 19–27 (2004)
15. Torres, V., Pelechano, V.: Building business process driven web applications. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 322–337. Springer, Heidelberg (2006)
16. De Troyer, O., Casteleyn, S.: Modeling complex processes for web applications using wsdm. In: Ws. on Web Oriented Software Technology (IWWOST), pp. 1–12. Oviedo (2003)
17. Webratio, http://www.webratio.com