# Business Process Modeling and Quick Prototyping with WebRatio BPM

Marco Brambilla[1], Stefano Butti[2], 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.** We describe a software tool called WebRatio BPM that helps close the gap between the modeling of business processes and the design and implementation of the software applications that support their enactment. The main idea is to enhance the degree of automation in the conversion of business process models into application models, defined as abstract, platform-independent representations of the application structure and behavior. Application models are themselves amenable to the semi-automatic transformation into application code, resulting in extremely rapid prototyping and shorter time-to-market. Thanks to the proposed chain of model transformations it is also possible to fine tune the final application in several ways, e.g., by integrating the visual identity of the organization or connecting the business process to legacy applications via Web Services.

## 1 Introduction

Business process modeling has become the recognized best practice for enterprise-wide application specification. Business process languages and execution environments ease the definition and enactment of the business constraints, by orchestrating the activities of employees and of computer-supported services.

However, turning a business process model into the specification, design and implementation of a software solution for process enactment is a non trivial task: the specified processes can be a mix on new functionality to be developed and interactions with pre-existing systems and the user's activities must be supported through effective and usable interfaces, possibly compliant with the visual identity and interaction style of other corporate applications. Furthermore, the business requirements embodied in the process models, as well as the technical context in which the underlying applications are deployed, are subject to evolution. This may cause severe alignment problems when trying to keep the business process and the application in sync.

The gap between process modeling and application development can be alleviated by increasing the degree of automation in the design, implementation and maintenance of applications derived from process models. The automation

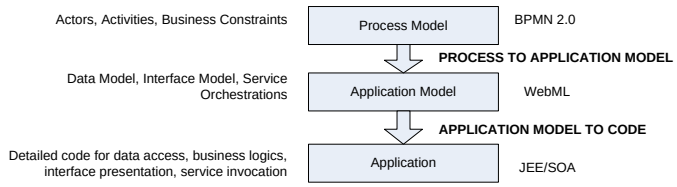



Fig. 1: Overview of the models and transformations

framework should support the semi-automatic translation of the process model into running applications, be flexible enough to incorporate different architectural and interaction requirements, and apply also to application evolution and maintenance. An outstanding difficulty is the semantic distance between the process model and the the running application: the former dictates roles, activities and business constraints at a very abstract level, irrespective of how these are supported by computing tools; the latter embodies very low-level details, where business, architecture, and interaction aspects are blended together and hard to customize and evolve separately. As an example, the same user's activity specified in the process model could be enacted in a variety of ways: by means of a wizard, by form editing, by using a legacy application interface, and so on. This observation motivates the approach applied in this paper, which is based on the three level conceptual architecture illustrated in Figure 1.

Following the approach of Model Driven Engineering, the business requirements and the application are represented using models, organized at three levels:

The models managed by our approach are organized in three levels: (i) the business model (specified with BPMN [6]); (ii) the structure and behavior of the application (expressed in WebML [3]); and (iii) the executable application running code. Application development is then seen as two consecutive transformations: the *Process Model to Application Model transformation*, and the *Application Model to Running Code transformation*.

The introduction of the application modeling layer increases the complexity of the conceptual architecture, but brings fundamental advantages: there is one place (the application model), where it is possible to reason about the distinct aspects of the application separately; the BPM to Application transformation can be supplied with transformation rules capable of producing alternative ways of encoding an activity, by using different patterns; automatically generated application models can be fine tuned, to introduce usability patterns, without breaking the application compliance to the process model; application evolution can be performed independently of the technical platform, by updating the application model and then regenerating the application code.

In the scientific community, several works have addressed the binding of BPM and Model Driven Development of Web applications: PML [5], YAWL [4], OOHDM [7], WSDM [8] and others. Our previous work [2] established the theoretical basis of the implementation described in here; with respect to that early

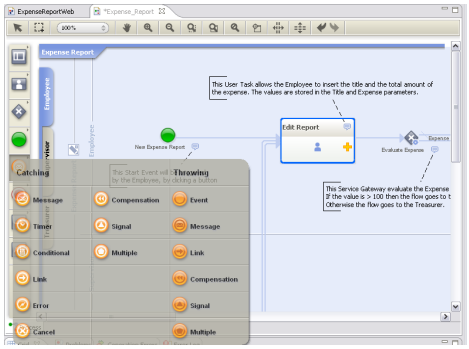



Fig. 2: The new WebRatio BPMN editor interface.

idea, now the BP model and the application model are treated as orthogonal and independent models.

Based on these premises, our original contributions are: (i) a model-driven perspective to business process-based software application development, which leverages the integrated use of two orthogonal models (BP and application models); (ii) a generative framework for producing the executable code from the process and application model, and a one-click, zero-coding generation of a running prototype on an enterprise class standard architecture; (iii) the implementation of the model editor (Figure 2) and transformations in a commercial tool suite called WebRatio [1], which supports all the steps of the proposed approach.

## 2 Models and Transformations

For describing the BP model, we adopt BPMN 1.2., plus some features of BPMN 2.0, whose Beta specification is currently available. Some features of version 2.0 of the language are of extreme importance for the generation of application models, namely: *DataInputs*, *DataOutputs*, and *DataInputAssociations*, which explicitly specify the inputs and outputs of executable tasks; the classification of *Tasks* in *UserTasks* and *ServiceTasks*; and others.

As a second level of modeling, we propose the application model, aiming at describing all the user interaction, service calls, and navigational aspects of the designed application. We focus on the Web as the platform of choice for the implementation of the software application, which is in line with the current trends in enterprise application development. Therefore, we adopt the *WebML* notation [3], a visual language for designing data- and service-centric Web applications [2], that allows specifying the conceptual model of applications built on top of a data schema and composed of one or more hypertexts used to publish or manipulate data. 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

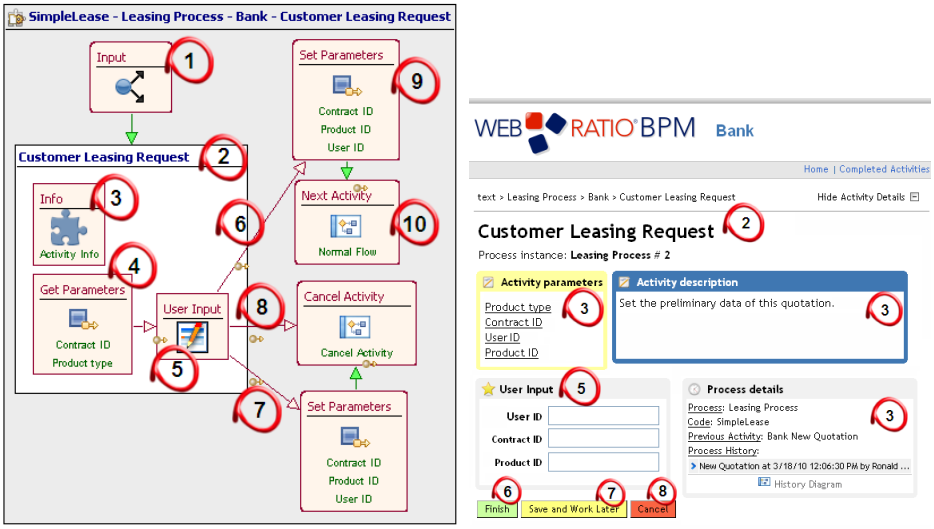
components. Units are related to each other through *links*, representing navigational paths and carrying parameters. Starting from a BPMN specification, an automatic transformation produces the logical representation of the process metadata and a WebML Application Model, comprising a Data Model dictating the application-specific concepts and a collection of Site Views and Service Views, including the primitives for the user interaction and Web service orchestrations.

**Process Data Model Generation.** The transformation from BPMN to the process data model consists of an encoding of the BPMN concepts in a relational database structure: the BPMN precedence constraints and gateways are transformed into instances of a relational representation. At runtime, the BPMN constraints, stored in the Process Metadata Schema, are exploited by the components of the Application Model for enacting the precedences among human-executed tasks and executing the service invocations.

**Application Model Generation.** The transformation from BPMN to WebML considers the type (User or Service) of the gateways and of the tasks, as well as the information on the control and data flows. The generated application models consist of a coarse set of user interfaces and prototype business operations. Process control is encapsulated thanks to the automatically generated process data model: the computation of the next enabled activities given the current state of the workflow is delegated to a specific WebML component, called *Next* unit, which factors out the process control logic. The *Next* unit exploits the information stored in the process data model to determine the current process status and the enabled state transitions. The tool also automatically generates the WebML model for user-driven and automatic tasks (e.g., performed by web services) and the hypertext for managing the tasklist and the process execution status.

Figure 3a shows an excerpt of the WebML model automatically generated from BPMN, describing a hypotethical activity that allows users to submit leasing requests online; the module contains an *Input* component (1) for initializing the possible data values in input to the activity (e.g., *ContractID*, *ProductType*); it includes a page (*Customer Leasing Request* (2) ) with a sub-module publishing information about the activity and the associated process execution (*Info* (3) ), a component fetching the current values of the parameters (*GetInput* (4) ) used to preload the fields of a form for submitting the user inputs (*UserInput*(5) ). From the input form, three links allow the user to close (6) , suspend (7) , and cancel (8) the activity. For example, when the user closes the activity, the parameter values are stored (by the *SetParameters* unit (9) ) and the next activities are calculated (by the *NextActivity* unit (10) ); the user interface automatically generated with a default graphical style is shown in Figure 3b.

**Generated runtime architecture.** The run-time architecture of the applications generated by WebRatio starting from the application model exploits a set of off-the-shelf components for organizing the business tier: *Smart service creation* for components, created upon request, cached, and reused across multiple requesters; *XML parsing and access* granted by standard parsing tools; and *Con-*




(a) Generated WebML model (b) Generated user interface

Fig. 3: WebML model and user interface for Customer Leasing Request activity.

*nection pooling* for dynamically managing a pool of database connection allow to optimize performance and reliability. At runtime, only service component is deployed for each type of model concept and one runtime XML descriptor is deployed for each usage in the application model.

WebRatio supports also the new requirements imposed by Rich Internet Applications (RIAs), including complex user interactions such as drag and drop, partial page refresh, dynamic resizing of visual components, graphical editing of objects, in-page popup windows, splash screens, dynamic tooltips, and waiting animations, text autocompletion, on-event actions, and field dependencies.

## 3 Tool Implementation and Experience

**Tool implementation.** The proposed generative framework has been implemented as an extension of WebRatio, a Model-Driven Web application development tool allowing one to edit WebML models and automatically transform them into a running applications for JEE and Service Oriented Architectures. For supporting BPM design, the following extensions have been devised. The *model editing GUI* has been extended by with a new BPMN editor . The *code generator* has been extended with the transformation from BP model to application model; furthermore, the JEE code generation has been augmented to produce the relational instance of the Process Metadata and the Java code of novel WebML components (e.g., the Next unit). 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. This functionality invokes

in sequence the two transformations from BPMN to WebML and from WebML to JEE, yielding a dynamic, multi-actor application with a default look & feel, as shown in Figure 3b. The generator creates a few exemplary users for each BPMN actor, which allows the analyst to impersonate each role in the process.

**Industrial applications.** The proposed method and tool have been tested in several real industrial scenarios: as an example, we report the outcomes of a project conducted within the leasing division of a major European bank, which needed to reshape its entire software architecture according to a BPM-based paradigm. The resulting pilot application covers 52 business processes, comprising more than 1,100 activities spanning 30 user roles. The development team comprised 14 staff members from 3 organizations, with total effort amounting to 2551 man days, spent in 375 elapsed days. Rapid prototyping let the team deploy 4 major prototypes and 35 minor releases along one year.

We estimated the spared effort induced by automatic model transformation by measuring the percentage of automatically generated model elements, which consists of a percentage ranging between 17% and 20%.

## 4 Conclusion

We presented a tool suite for supporting the model-driven integrated design of business processes and enterprise Web applications. Visual model design and model transformations allow designers to produce both as early prototypes and final applications without coding.

## References


1. Roberto Acerbis, Aldo Bongio, Marco Brambilla, Stefano Butti, Stefano Ceri, and Piero Fraternali. Web applications design and development with webml and webratio 5.0. In *TOOLS (46)*, pages 392–411, 2008.
2. Marco Brambilla, Stefano Ceri, Piero Fraternali, and Ioana Manolescu. Process Modeling in Web Applications. *ACM TOSEM*, 15(4):360–409, 2006.
3. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, USA, 2002.
4. A.H.M. Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell (Eds.). *Modern Business Process Automation, YAWL and its Support Environment*. Springer, 2010.
5. John Noll and Walt Scacchi. Specifying process-oriented hypertext for organizational computing. *J. Netw. Comput. Appl.*, 24(1):39–61, 2001.
6. OMG, BPMI. BPMN 1.2. Technical report, http://www.bpmn.org/, 2009.
7. Hans Albrecht Schmid and Gustavo Rossi. Modeling and designing processes in e-commerce applications. *IEEE Internet Computing*, 8(1):19–27, 2004.
8. Olga De Troyer and Sven Casteleyn. Modeling complex processes for web applications using wsdm. In *Ws. on Web Oriented Software Technology (IWWOST)*, pages 1–12, Oviedo, 2003.