

A Conceptual Modeling Approach to Business Service Mashup Development

Alessandro Bozzon, Marco Brambilla
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Milano - Italy
{bozzon, mbrambil}@elet.polimi.it

Federico Michele Facca
Semantic Technology Institute
University of Innsbruck, Innsbruck - Austria
federico.facca@sti2.at

Giovanni Toffetti Carughi
Faculty of Informatics, University of Lugano, Lugano - Switzerland
toffettg@lu.unisi.ch

Abstract

Professional mashups that include complex choreographies, data mediation, and result publishing within Web pages are still affected by implementation and design practices that rely either on very simple models or on low-level scripting and programming skills of developers, thus hampering the use of mashups in business context as rapid solution to immediate problems. Indeed, industrialization of their development is still a hard objective to achieve.

We propose a design methodology based on visual models to improve the quality and the productivity of service mashups and presentation of the results, thus increasing their acceptance as professional applications in the business scenario. Existing software engineering methods are combined together in an innovative mix, comprising standard business process modeling languages (namely, BPMN) to describe a high-level view of the mashup orchestration and on WebML (Web Modeling Language) to specify the detailed Web application model, including Web service interactions, hypertext navigation, event management, and rich user interfaces.

1 Introduction

Mashups are applications that leverage creative composition of existing functionalities, data, and interfaces to achieve a more complex goal. They integrate data from different sources, typically wrapped as Web services, and typically show the results of the integrated invocations within Web sites that feature good configurability, personalization, and usability. To achieve these objectives, applications integrate rich interface programming, based on AJAX or similar approaches. Mashups allow end users and practitioners to develop quickly customized solutions for their problems:

the implementation of simple cases or toy applications is quite straightforward thanks to easy coupling of RESTful services, provided by a plethora of online interfaces and tools (e.g., Yahoo Pipes).

Recently, mashup applications are gaining importance on the Web, thanks to their flexibility, composability, and ease of use. However, they are currently accepted and adopted only in the consumer scenario, where non-professionals implement simple combinations of services. On the other side, they are not yet exploited as professional applications in business scenarios, because of the lack of an industrialized development process.

Professional development of business mashup applications with currently available tools is often tedious due to the low level programming that is needed for producing professional service compositions (e.g., including both RESTful and SOAP services) and Web application interfaces. Model Driven Development (MDD) of service mashups can increase the productivity of developers and the maintainability of the application, thanks to good design documentation, semi-automatic code generation, and quick composition and update of models.

We propose an approach that exploits existing software engineering methods to specify the service composition and to integrate the resulting service invocations within user-oriented Web application interfaces. We rely on standard business process modeling languages (namely, BPMN) to describe a high-level view of the business mashup; then, through automatic model transformation we obtain a model in a Web-specific modeling language (WebML) that is subsequently refined. In WebML, a set of domain-specific primitives allows to visually model the Web service interactions, the events that trigger the different mashup parts and the (possibly partial) update of the Web interface showing the results. The specifications of the page navigation and of the user interface are described through Rich Internet Appli-

cation models, thus granting usability and configurability of user interfaces.

The main added value of the proposal consists in defining a sound mix of standard specification languages and model transformations for addressing all the aspects of Web business mashups design, together with the specification of a formalized development process. This position our research halfway between mashups and traditional conceptual modeling, thus enabling for the development of prosumer mashups.

The paper is organized as follows: Section 2 discusses some background; Section 3 presents a running case scenario; Section 4 presents our approach; Section 5 reports some implementation experiences and evaluations; Section 6 discusses the related work; and Section 7 concludes.

2 Background

The methodology presented builds on top of existing languages and methods to cover the different phases of mashup design. For the high level specification of the service composition, we adopt BPMN (Business Process Management Notation)¹, the reference model for business process, which allows to visually specify actors, tasks, and constraints involved in a business processes. 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. This language is powerful enough for describing orchestrations required by business mashup applications. Then, by applying MDD techniques, we transform BPMN models into more detailed WebML models.

WebML (Web Modeling Language) [6] is a suite of orthogonal models that cover the specification of the contents, business logics, hypertext, presentation of a Web application. The content can be modeled using Entity-Relationship (E-R) diagrams, UML class diagrams, or ontology specifications. Upon the same content model, it is possible to define different hypertext models (called site views), targeted to different user roles. A site view is a graph of pages, possibly hierarchically organized into sub-pages. Pages comprise content units, i.e., components for publishing content coming from a class of the content model, possibly restricted by selectors (i.e., logical condition filtering the instances). Units are connected to each other through links, which carry parameters and allow the user to navigate the hypertext. WebML also allows specifying operations implementing arbitrary business logic (e.g., to manipulate content instances), and Web service invocation and publishing

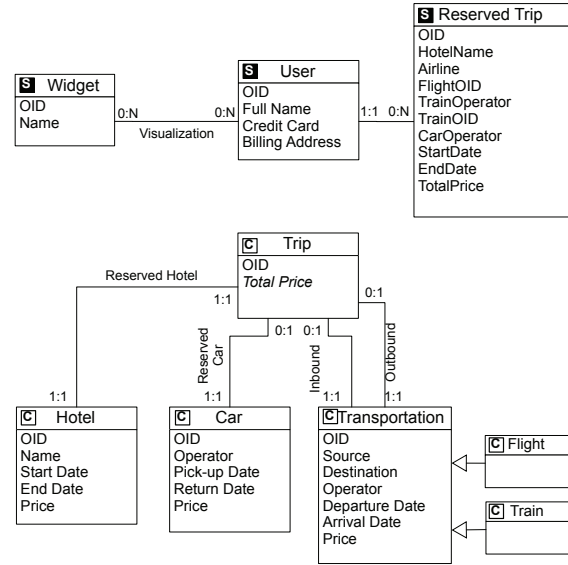


Figure 1. Travel Agency Content model.

[18]. Some existing works describe how to map business processes to Web application implementations [3]. Finally, WebML supports model driven design of Rich Internet Applications [5], featuring new interaction paradigms and application functionalities enabled by events, data, and distribution of computation between server and client.

A detailed description of the selected modeling languages is outside the scope of the paper. For further details the reader may refer to [21] and [6]. It's worth notice that the proposed approach is a general purpose method not bound only to business mashups. Furthermore adopted models can be easily replace by any alternative business model notation (e.g., BPEL, UML Activity diagrams, and so on) and to most Web engineering methodologies.

3 Case Study: yet another Travel Agency

To describe our methodology, throughout the paper we will use a case study: a company needs to set up a simple travel agency application to allow its employee for organizing travels to a specific event. The application provides a single-page Web interface that allows the user to specify his starting location, his destination, and his preferred transportation (flight or train), together with a range of dates available the trip. After submitting the data, the system retrieves the available flights (or train trips), as well as the hotels availability, and the car rental options. Then, the user can select one flight (or train trip): correspondingly the system automatically prunes the hotel and car rental options considering the chosen transportation. Different information (flight, train trip, etc.) are retrieved from different services, already available online.

¹<http://www.bpmn.org>

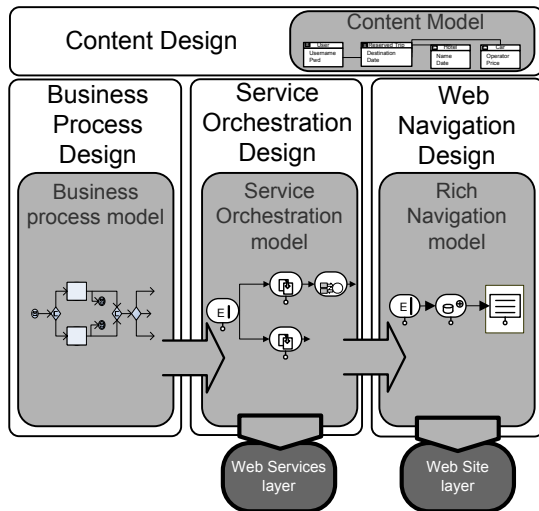


Figure 2. Overview of the development process, of the involved models, and of the resulting layers.

Figure 1 shows the content model of the application: classes can be stored at server-side (marked with ‘S’) or at client-side (marked with ‘C’). The *User* class, stored on the server, represents the user profile, also including the status of his interface, in terms of *Widgets* (i.e., atomic components of the interface) displayed in the page. A *Visualization* relationship instance indicates that the current page interface for a specific user is displaying the widget. The *User* entity is also associated with the entity *Reserved Trip*, representing a reservation performed by the user. The client-side *Transportation*, *Hotel*, and *Car* classes, instead, represent information temporarily stored on the user’s client as partial results of the mashup, gathered within the *Trip* class. Client-side data is obtained by the browser directly invoking composed services, thus lowering the burden on the application server. The retrieved information is then used in combination with client-side scripting to provide a richer interface for additional service invocations and manipulation of results (e.g., filtering based on user-specified conditions).

4 Model-driven Mashup Design

In order to address real-world business needs, the development of prosumer mashups should follow a formalized development process, possibly involving different development roles. In this sense, our design approach differentiates from early proposals of end user- oriented solutions (e.g., Yahoo Pipes and similar initiatives). To support the different professional roles involved in the implementation of a service mashup, a clear view of the development process must be shared among all the actors. We propose a design

process based on MDD based on the following steps (Figure 2 summarizes the approach):

- **Business process model:** as first step, the business manager specifies the high-level view of the mashup orchestration, by means of a business process model. The model is then refined by the IT expert: a task is devised for each service invocation, and then it is connected by control links to other invocations or, possibly, to primitives for throwing events to the hypertext model. Events are then caught in the hypertext navigation models and define the page refresh or the content calculation time.
- **Web service orchestration model:** starting from high level specifications, a set of automatic MDD transformations generates a more refined view of the orchestration in a Domain Specific Language (DSL) for Web applications and Web services. We choose WebML because of its good abstraction level and implementation support. The automatic transformation to WebML models consists on a pattern-based translation of BPMN diagrams in hypertext and business logic models. Each service invocation task is translated in a fixed pattern of business actions, that include service invocation (both SOAP and RESTful), local storage of the results (to support stateless interaction with services and to enable client-side logic), and data marshalling between services, thus granting proper support to service mashup.
- **Rich navigation model:** finally, the interaction designer specifies the logics and the navigation paths of the Web page that shows the results of the service mashup to the user. This step includes the design of the page interface behavior, the page refresh rules, and the allocation of content (client- or server-side). Different parts of the page can be computed and refreshed separately upon reception of events. Conversely, the user interactions with the page interface can throw events that trigger the execution of new elements of the mashup. Existing client-side components can be modularized in ad-hoc client side widgets, thus enhancing reusability. The ability to model the composition of “vertical” interface widgets (e.g., an interactive map) with content and events coming from services enables the specification of intuitive, usable yet powerful GUIs, adaptable to the specific business requirements.

The development process presented is envisioned to support automatic code generation. At this purpose, we extended a commercial CASE tool called WebRatio (www.webratio.com), that natively provides automatic code

generation of Web applications starting from WebML models. Thanks to our extensions, full-fledged service-mashups can be generated, with great advantages in terms of productivity and quality of the outcome.

4.1 Service Orchestration

The specification of the service interactions that constitute the mashup can be achieved by (1) describing the high level process model and (2) refining the resulting model, thus obtaining a detailed view of the orchestration.

4.1.1 Process Model

The business process design task focuses on the high-level schematization of the service mashup orchestration, and results in one or more business process diagrams, each one representing at abstract level the service composition chain to be executed. Each *service invocation task* can be annotated with an *implementation tag*, which specifies whether the service has to be modeled and implemented within the system (*internal*), shall be invoked as an existing third-party service (*external*), or needs to be searched and invoked by means of semantic discovery and invocation techniques (*semantic*).

The business process also includes the definition of *events*, specified as BPMN message events, that allow the bidirectional communication between the user interface and the background service mashup:

- The UI can issue events (e.g., upon user interaction) which are caught by the process model and trigger pieces of service mashup logic;
- The service mashup logic can issue events as well (e.g., upon completion of a service call), and correspondingly the UI reacts by updating the page rendering according to the rich navigation design.

Finally, control flows and BPMN gateways are used to define the logic of the service composition flow (e.g., whether two service have to be invoked in parallel or as an alternative).

The top part of Figure 3 shows the BPMN model of the case study: when the service composition receives a *TravelSearch* event, depending on the user's choice, trains or flights are retrieved and the corresponding event (*FlightOffer* or *TrainOffer*) is sent back to the user interface that generated the triggering event. Notice that the two activities *Flight* and *Train* are mutually exclusive as specified by the *XOR* gateway.

Then, both the hotel and the car rental services are performed (*AND* gateway), and their corresponding events (*RentalOffer* and *HotelOffer*) are sent to the UI. The hotel reservation may also be triggered by the reception of an

HotelSearch event, sent by the UI when the user decides to refine the booking parameters (e.g., to manually chose an alternative location).

4.1.2 Mapping the Process Model to WebML

The abstract high-level service mashup model is used to generate a WebML skeleton of the corresponding orchestration specification. This is achieved by an automatic translation of the BPMN model [2]. In particular, BPMN elements are transformed to WebML primitives as follows:

- BPMN start events are translated to WebML *ReceiveEvent* primitives, while intermediate events and end message events are mapped to *SendEvent* primitives. The *SendEvent* primitive allows to notify an event to a (set of) recipient(s); the *ReceiveEvent* primitive triggers the associated action(s), upon reception of an event.
- Workflow activities become service invocation chains, including: a data marshalling primitive that performs the lowering of the application content model to the web service SOAP request; a Request-Response primitive that performs the actual service invocation; and finally a data marshalling primitive that performs the lifting of the web service SOAP response to the application data model. Data marshalling is concretely implemented as XSLT transformations.
- Workflow constraints are turned into WebML hyper-text constraints defined on the content model. Links to the invocation chains may include condition checking primitives. For instance, exclusive branches are generated as condition evaluation primitives that define which service invocation chain should be executed. A detailed description of the BPMN to WebML transformation rules is described in [3].

The lower part of Figure 3 shows a fragment of the WebML model corresponding to the BPMN diagram. The service composition is triggered by the *ReceiveSearch* primitive, which passes the attributes of the received event to the WebML control flow. The *TrainOrFlight?* switch unit tests whether the event contains a "Flight" or "Train" transportation type and activates the corresponding operation chain. In case of a "Flight" connection type, event parameters are passed to lowering XSLT script that prepares the message for the service invocation. After the service invocation, a set of flight offers is processed by the lifting XSLT script that bring them back to the inner data model formalism. The same applies in case of a "Train" connection type. Notice that in both cases several concrete services can be invoked at the by the for obtaining a good result. Moreover, the model may include interaction with other kinds of remote services, such as RSS feeds or REST web services.

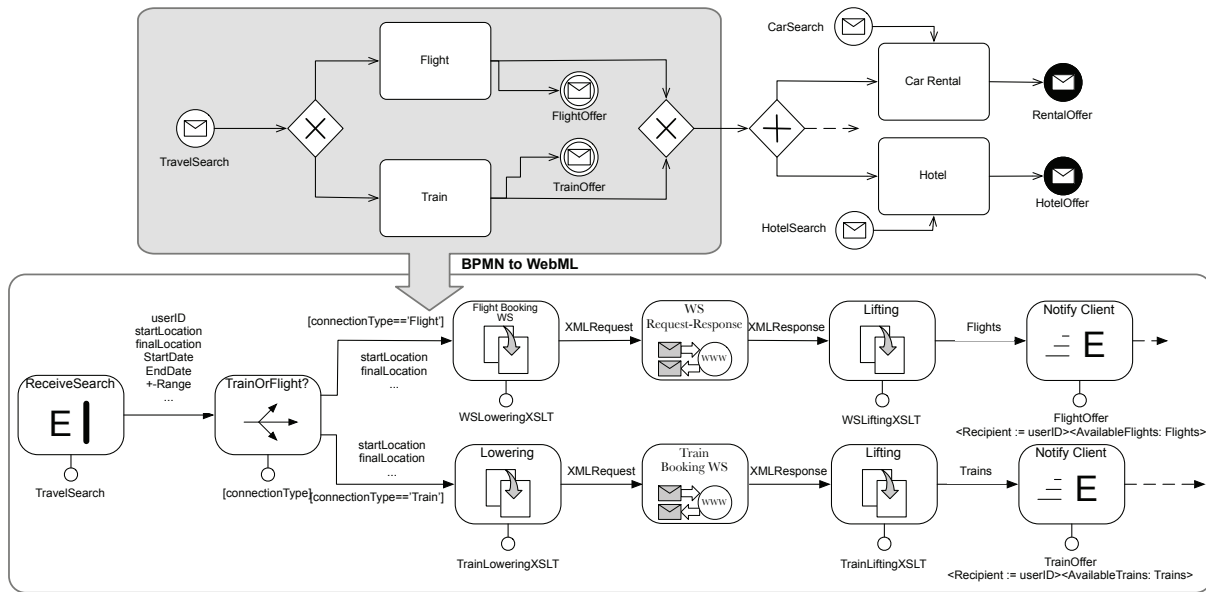


Figure 3. Business Process Model and Service Orchestration Model.

4.2 Rich Navigation model specification

The rich navigation model defines the UI composition and behaviour. This can be achieved through WebML RIA page structure and navigation models, that include primitives for content publishing (indexes, data details, and so on) and rich user interaction (links, forms, menus, widgets). Such approach allows fine-grained specification of the mashup GUI in terms of all its components: a computational model lets the application designer tie user- and service-generated events with the interface providing a complete specification of the application behaviour.

Figure 4 depicts a fragment of the hypertext model for the case study application. It represents a RIA page (*My Trip*) with computation occurring on the client side (the browser), therefore it is marked as ‘C’. The page includes four sub-pages: *MyFlights*, *MyTrainRides*, *MyAccommodations*, and *MyCars*, each one representing a widget in the user interface. Page composition is performed at run-time, based on user preferences: the visualization of the widgets is automatically bound to the presence of a *Visualization* relationship instance.

The reservation process is performed by means of a search form, represented by the *Trip Details* entry unit. The link exiting the entry unit represents the submission button of the form, which, by triggering the *RequestTransportation* operation, throws a *TravelSearch* event. The execution of the workflow represented in Figure 3 triggers the *ReceiveFlight* (or the *ReceiveTrainRide*) and the *ReceiveHotel* receive-event units. As effect of their activation, the *Flight* (*Train*) and *Hotel* client entities are asyn-

chronously and independently populated, and the units contained in the *MyFlights*, *MyTrainRides*, and *MyAccommodations* widgets display the retrieved values. The selection of a flight (or a train) as inbound or outbound transportation provide the *AvailableHotels* unit with the parameters requested for the computation of its client selectors (*[availableSince]* and *[availableUntil]*), which locally filter the available hotels against the selected dates while maintaining the current state UI and, thus, the state of the other widgets. Finally, the selection of a hotel from the *AvailableHotels* index unit activates the notification of the *BookEverything* send-event unit, which triggers the back-end booking confirmation process.

5 Implementation and evaluation

Our methodology has been implemented within *WebRatio*, a CASE tool that supports WebML-based application design and provides automatic code generation. To cover the specific needs of service mashups, several extensions have been implemented both in the design tool and the runtime software libraries that support the automatically generated applications. The business process design has been supported by an Eclipse plugin [2], that comprises fully configurable MDD transformations of workflow models to the needed target models. A specific transformation has been designed and implemented to generate the WebML models according to the method presented in this paper. A fully online version is under development too [4], thus moving towards on-the-fly mashup design. In both cases, the resulting WebML model can be refined by the designer in We-

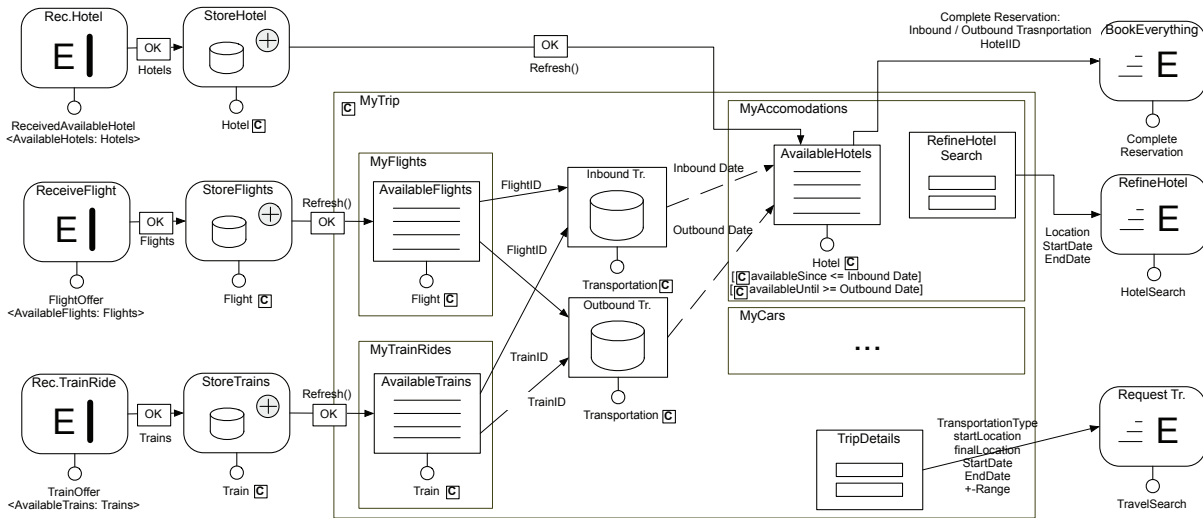


Figure 4. Rich Hypertext Model of the mashup results rendering page.

bRatio, which has also been extended to support client-side and server-side modeling of data, navigation, and events. The new components have been included in the runtime libraries, that are now able to manage events and client-side data management. Finally, the generated code for the hypertext pages now comprises the behavior of rich Web interfaces, including runtime page personalization (to move and/or hide UI components). This is achieved by generating the appropriate client-side code to allow users to edit the page interface, together with the logic to reload interface preferences according to each user profile.

The implementation of the illustrated primitives granted us the opportunity to adopt business mashup conceptual modeling for the development of real-world applications. A representative example is the work performed within the PHAROS project², where the described modeling primitives and code generation facilities have been exploited to build a working prototypes of an audiovisual search engine. PHAROS combines complex workflows for the Content Processing (CP) as well as for the Query and Result Presentation (QRP) processes. Our approach have been successfully applied in order to design both CP and QRP, where the workflows included the interaction with several enterprise-class services, like search engines (e.g. textual, content-based, etc.), social interaction modules (e.g. tagging, commenting, etc.) and third party geographic services (e.g., Microsoft live maps).

The evaluation of the approach concentrated on two essential issues for a MDD, that are ease of use and effectiveness: the devised solution should result effective in terms of learning curve, intuitiveness and communicability of the specifications, and ease of maintenance. Ease of use and

effectiveness can be qualitatively evaluated by comparing model-driven development of mashups with their manual coding: this is a long debated issue [20], which is outside the scope of this paper. Nonetheless, the use conceptual models and code generation enables the generalization of the software design and implementation into an abstract modeling and development frameworks. Such an approach results beneficial especially when technology is evolving and common standards are lacking: this is the current situation with mashup development, where a variety of different tools and architectures are available for much the same objectives. An alternative approach may consider a comparison of the model-driven development using a traditional Web Engineering approach with an extended methodology designed to address specific mashups requirements: the experience of adopting the proposed approach to developers trained in Web application developers revealed both positive and negative aspects. On the positive side, the high-level, conceptual modeling of business processes, service orchestration and rich interfaces was considered an extension both intuitive and with substantial added value: more specifically, the automatic generation of working prototypes directly from the business model specification, as well as the capability of creating libraries of reusable business logic component and client-side graphical widget has been perceived as a significant advantage. On the negative side, the less satisfactory aspect concerns the effort required to adapt existing components to work within the modeling and code generation framework: if new RIA interactions are needed, new components must be developed. Analogously, if new orchestration patterns are envisioned, the BPMN-WebML transformation must be adapted accordingly.

²www.pharos-audiovisual.eu

6 Related Work

The composition of RESTful services to provide new ones is widely referred to as *mashup* [24]. The work in [17] advocates simple lightweight service composition for non technical-savvy users: different Web-based tools such as Yahoo Pipes (*pipes.yahoo.com*), Microsoft Popfly (*www.popfly.ms*), Google MashupEditor (*editor.googlemashups.com*), and IBMs QEDWiki (*services.alphaworks.ibm.com/qedwiki*) have been developed to hide the technical details of such a chore. The common approach is to offer a set of available services to be visually chained together as well as some wrappers for simple XML operations. While these approaches provide a good starting point, their main shortcoming resides in the number of combinable services which are usually company-dependent, or simply RSS and ATOM feeds. To obviate to this drawback [14] proposes semantically annotating HTML pages for REST services as well as specifying lifting and lowering schemas to ease their integration and data mediation.

The previous approaches aim at quickly combining existing services (bottom-up) to produce a simple (build once - use once) mashup. As such, they offer small, flexible, simple components to produce a limited set of trivial compositions. While ontologies can indeed simplify the task by reducing part of the data mediation tasks, such a low-level approach suffers from the lack of an adequate perspective w.r.t. the composition process as well as the mashup interface. Within our modelling approach we address both process-related as well as service and interface composition issues. By lifting the abstraction level, we bring in the benefits of conceptual modelling enabling the creation of solid and complex mashups as the one requested by business context, at the same time, we provide a lightweight, on demand model-driven development as advocated in [10].

Even if several methodologies and tools exist for the conceptual modelling of both general-purpose and vertical applications, to our knowledge very few works in literature explored how to address the requirements of mashup applications using conceptual modeling, automatic code generation, and model property checking.

In [8] mashups are observed from the point of view of user interface integration: w.r.t. this aspect our approach allows for both integration at the application level (which is inherently more powerful as the authors confirm) as well as integration at the UI level. Even at this level, our solution offers the additional benefit of abstraction, thus being able to conceptually model the integration logic needed to map the output of a component to one's input, especially if it involves complex operations (e.g., converting an address into geospatial coordinates by using a sequence of external service calls).

Since a service mashup can be defined as an orchestrated set of services, facilitating and improving the quality

of service orchestration is of primary interest for developers. Several approaches have addressed in the past the composition of services by means of orchestration [1]. Even some specific languages for describing service choreography have been specified, such as WS-CDL³. Distributed and decentralized orchestration of services has been widely addressed as well (e.g., [11]). However, mashups need a more lightweight approach to service composition. Some tools exist that effectively perform service composition (*www.iks.ethz.ch/jopera*), but they typically lack the appropriate management of the results: indeed, mashup design tools must also integrate visual rendering of the results and user interaction design.

Mashup application development could be considered as a special case of 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) described by Noll and Scacchi [19], for instance, is an early proposal for the automatic generation of simple Web-based applications that allows users to enact their participation to the process. Koch et al. [13] 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, where standard UML constructs are used. The design of the application model, instead, is separated. In our work, both approaches 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.

An alternative approach is proposed by Torres and Pelechano [23], where BPM and OOWS [12] 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 et al. [15] presents a set of transformations for automatically generating a set of UML artifacts from BPM. With respect to the literature on business process integration within general-purpose Web Engineering methods, our work differs in its specific focus on mashups, which highlights the core processes behind these solutions and demonstrates how data-centric and process-centric MDD methods can be brought to bear on this emerging class of applications.

From the navigation and presentation standpoint, we leverage Rich Internet Applications (RIAs) experience, that combines the benefits of the Web with the interactivity and

³<http://www.w3.org/TR/2004/WDws-cdl-10-20041217>

power of desktop applications to enable the implementation of Web 2.0 applications. Although tool support is fairly good for coding the interface and defining the client-side behavior, none of the existing approaches has yet addressed the problem as a whole, including the server side development of the functional viewpoint of RIAs. Several Web Engineering methodologies have been proposed in literature (e.g., OOHDM [22], WAE [7]), but few of them fully address the novel architecture, functionalities, and behavior introduced by Rich Internet applications. An exception is represented by a recent work [16], where authors provide a method for adapting legacy model-based applications to the requirements of RIAs.

7 Conclusion

In this work we have presented a methodology, a set of models, and a toolsuite implementation for supporting top-down, model-driven design of professional service mashups like the ones needed in business domain. We have stressed that also integration within user interfaces of data resulting from mashups is an important success factor. At this purpose, we have included in our methodology a set of facilities for designing rich interfaces that allow good fruition of the contents by the users.

Mashup development with our methods and tools has been tested on several case study applications and the achieved results are promising, both in terms of productivity of the developers and of quality of the implemented mashup applications. The next step in our research work is towards the integration of Semantic Web capabilities in the Web service orchestration, with the aim of providing runtime service discovery and better content matching between the services.

References

- [1] B. Benatallah and Q. Z. Sheng. Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. *Distrib. Parallel Databases*, 17(1):5–37, 2005.
- [2] M. Brambilla. Generation of WebML Web Application Models from Business Process Specifications. In *ICWE 2006*, pages 85–86, 2006.
- [3] M. Brambilla, S. Ceri, P. Fraternali, and I. Manolescu. Process Modeling in Web Applications. *ACM Trans. Softw. Eng. Methodol.*, 15(4):360–409, 2006.
- [4] M. Brambilla and A. Origgi. MVC-Webflow: An AJAX Tool for Online Modeling of MVC-2 Web Applications. In *ICWE 2008*, pages 344–349, 2008.
- [5] G. T. Carughi, A. Bozzon, S. Comai, and P. Fraternali. Modeling Distributed Events in Data-Intensive Rich Internet Applications. In *WISE 2007*, pages 593–602, 2007.
- [6] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., 2002.
- [7] J. Conallen. *Building Web applications with UML, 2nd ed.* Addison Wesley, 2002.
- [8] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*, 11(3):59–66, 2007.
- [9] S. Debal, W. Nejdl, F. S. Nucci, R. Paiu, and M. Plu. PHAROS - Platform For Search of Audiovisual Resources Across Online Spaces. In *SAMT (Posters and Demos)*, volume 233 of *CEUR Workshop Proceedings*, RWTH Aachen, 2006. CEUR-WS.org.
- [10] S. Dustdar. Towards Autonomic Processes and Services. In *BPSC 2007*, 2007.
- [11] G. B. C. et al. Decentralized orchestration of composite web services. In *WWW Alt. 2004*, pages 134–143, 2004.
- [12] J. Fons, V. Pelechano, M. Albert, and O. Pastor. Development of web applications from web enhanced conceptual schemas. In *ER Workshop on Conceptual Modeling and the Web*, volume 2813 of *LNCS*, USA, 2003. Springer.
- [13] N. Koch, A. Kraus, C. Cachero, and S. Meliá. Integration of business processes in web application models. *J. Web Eng.*, 3(1):22–49, 2004.
- [14] J. Lathem, K. Gomadam, and A. Sheth. SA-REST and (S)mashups: Adding Semantics to RESTful Services. *ICSC 2007*, pages 469–476, 2007.
- [15] P. Liew, K. Kontogiannis, and T. Tong. A framework for business model driven development. In *STEP '04: Proceedings of the 12 International Workshop on Software Technology and Engineering Practice*, pages 47–56, Washington, DC, USA, 2004. IEEE.
- [16] M. Linaje, J. C. Preciado, and F. Sanchez-Figueroa. Engineering Rich Internet Application User Interfaces over Legacy Web Models. *IEEE Internet Computing*, 11(6):53–59, 2007.
- [17] X. Liu, Y. Hui, W. Sun, and H. Liang. Towards service composition based on mashup. *Services, 2007 IEEE Congress on*, pages 332–339, July 2007.
- [18] I. Manolescu, M. Brambilla, S. Ceri, S. Comai, and P. Fraternali. Model-Driven Design and Deployment of Service-Enabled Web Applications. *ACM Trans. Inter. Tech.*, 5(3):439–479, 2005.
- [19] J. Noll and W. Scacchi. Specifying process-oriented hypertext for organizational computing. *J. Netw. Comput. Appl.*, 24(1):39–61, 2001.
- [20] M. Nussbaumer, P. Freudenstein, and M. Gaedke. Towards dsl-based web engineering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 893–894, New York, NY, USA, 2006. ACM.
- [21] M. Owen and L. Raj. Bpmn 1.0: Omg final adopted specification. Technical report, 2006.
- [22] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Hypertext*, pages 116–128, 1996.
- [23] V. Torres and V. Pelechano. Building business process driven web applications. In *Business Process Management*, volume 4102 of *LNCS*, pages 322–337. Springer-Verlag, 2006.
- [24] B. Worthen. Mashups Sew Data Together: Software Tools Can Cut Costs, Time for Linking Information Sources. *The Wall Street J.*, v250 n25 pB4(1), 31 July 2007.