Model-driven Specification of Web Services Composition and Integration with Data-intensive Web Applications*

Marco Brambilla, Stefano Ceri, Sara Comai, Piero Fraternali, Ioana Manolescu
Dip. Elettronica e Informazione - Politecnico di Milano - I 20133 Milano, Italy
{mbrambil, ceri, comai, fraterna}@elet.polimi.it , Ioana.Manolescu@inria.fr

Abstract

The integration of Web services and Web applications offers challenging opportunities to academic and industrial research; the relevant issues include new methods, paradigms, and standards for supporting Web services and their composition. This paper focuses on the specification of Web applications that compose Web services in order to support arbitrarily complex processes. The proposed approach extends a high-level modeling language, called WebML, used for the specification of the front-end of Web applications. We develop two orthogonal extensions for the inclusion of Web services inside Web applications, accompanied by suitable protocols for message exchange, and the empowerment of Web modeling primitives with workflow capabilities. The combined use of these two features gives to WebML enough expressive power for specifying complex Web service interactions.

1 Introduction

In modern Web applications, most pages are dynamically composed from content extracted from corporate databases, and the business logic is based upon several interacting components. The design of "data-intensive Web applications", i.e., of Web applications making extensive use of databases, takes advantage of declarative Web site specification and modeling languages, such as WebML (http://www.webml.org), capable of producing high-level, unified descriptions of the data and front-end requirements of the application. While such specifications adequately cover the needs of applications developed within a single organization, they do not cover well the integration of Web applications with externally provided business logic. Nowadays, the concept of Web service is gaining popularity as a uniform interface for distributed software components, and several related standards and tools are being proposed for wide-scale interoperability.

This paper addresses the design and specification of data- and service-intensive Web applications, leveraging an existing visual language for modeling data-intensive Web sites, and applies the benefits of high-level, declarative specification to the combined use of data and services. In order to represent the Web services behavior, we provide a set of graphical primitives that represent the services and allow one to compose services and hypertexts in a diagrammatic representation, amenable to automatic code generation.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This research is part of the WebSI (Web Service Integration) project, funded by the EC in the Fifth Framework.

One of the main contributions of our work is to recognize that primitives for invoking Web services and primitives for composing them into complex processes are orthogonal. In our proposal, the former primitives reflect the Web Service Description Language standard(http://www.w3.org/TR/wsdl), while the latter primitives correspond to the Workflow Management Coalition model(http://www.wfmc.org). These extensions give WebML the ability to describe arbitrarily complex processes, possibly interacting with remote services, granting a significant increase of expressive power; the incorporation of new primitives is seamless, thanks to the extensibility of WebML.

2 Modeling data-intensive Web applications using WebML

Traditional data-intensive Web applications can be modeled using high-level specification languages. In this paper we present Web Modeling Language (WebML) [1] [2] [3], but the examples and results are independent of its particular notations and could be applied to other specification languages. WebML allows specifying a Web site on top of a database. Such a conceptual Web specification consists of a data schema, describing application data, and of one or more hypertexts, expressing the Web interface used to publish these data.
The data model. The WebML data model is the standard Entity-Relationship (E-R) model, widely used in general-purpose design tools. As an example, Figure 1 shows the E-R schema describing the database of the online computer store. Every entity has a unique identifier attribute, named ID, which is not shown.

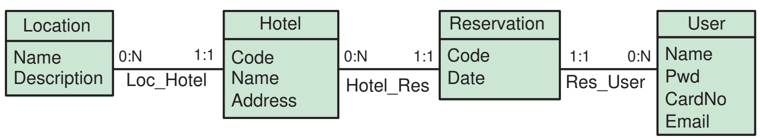

Figure 1: Sample data model for an online travel agent.

The hypertext model. Upon the same data model different hypertexts, called site views, can be defined, e.g., for different types of users or access devices. A site view is a graph of pages, which are presented on the Web. Pages enclose units, representing atomic pieces of information to be published; a unit selector specifies a predicate identifying the data to be extracted from the underlying database and displayed by the unit. Pages and units can be linked to form a hypertext.
Besides content publishing, WebML allows specifying operations, like the filling of a shopping cart or the update of the users' personal information. Basic data update operations are: the creation, modification and deletion of instances of an entity, or the creation and deletion of instances of a relationship. Operations do not display data and are placed outside of pages; special purpose operations can be specified, such as sending e-mail, login and logout, and e-payment.
Example. Figure 2 shows an instance of a simple online travel agent Web site, and the corresponding WebML graphical specification. The leftmost page shows a list of generic touristic locations, displayed as text anchors. The corresponding WebML model presents a Locations page enclosing an index unit named "Locations Index". When the user selects a location, he is taken to a page displaying data about the selected location and the list of related hotels. This target page is modeled in WebML enclosing a data unit ("Location Details"), displaying the attributes of the selected place, and an index unit displaying all the hotels. The Hotels page encloses an entry unit enabling the user to insert a new hotel; clicking on the "submit" button activates an operation chain composed by a create operation, generating a new hotel instance, and a connect operation, connecting the new hotel to the currently selected location.
The language includes several other units and operations, and is extensible with custom operations and units. WebML has a well-defined semantics that enables automatic generation of full-fledged, functional Web sites [2].

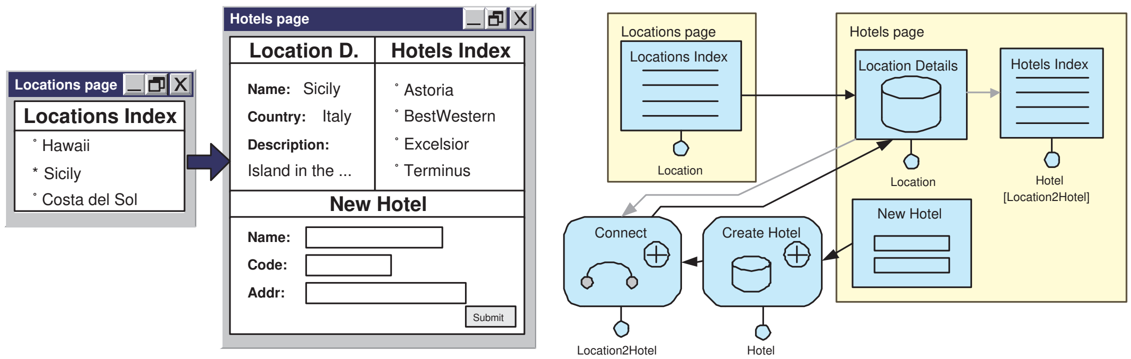

Figure 2: Hypertext fragment defined on the computer store data model.

3 Extensions for modeling Web services

According to WSDL , a Web service consists of operations that can be invoked following simple patterns; each operation may receive an input message from the Web application and may send an output message to it. Thus, the extension of WebML with support for web services requires:

- New operations for interacting with the Web service.
- Data extraction capabilities for transferring information from messages to the database hosted at the Web site and vice-versa.
- Support for synchronous and asynchronous bi-directional messaging.

The three requirements above are essential for the design and specification of a data- and service- intensive Web applications; fulfilling them is the novelty and the specific contribution of this paper. The main source of complexity arises from the fact that Web services are designed as building blocks for complex interactions among peers, which can be started by anyone of the peers. Web applications use instead the HTTP protocol, which is always initiated by user's requests. This mismatch has some consequences:

- By the nature of Web applications, we expect to see only a few cases of interactions started by the service (i.e., notification or solicit-response in WSDL).
- Most interactions with services included in Web applications are supposed to be synchronous, as imposed by the HTTP protocol, in which, after issuing a request, the client has to wait for the response. The solution adopted for asynchronous messages is to store them in a persistent manner and notify them to the user whenever he/she returns to the application.
- In a Web application, it is not possible to force the user to respond synchronously when solicited.

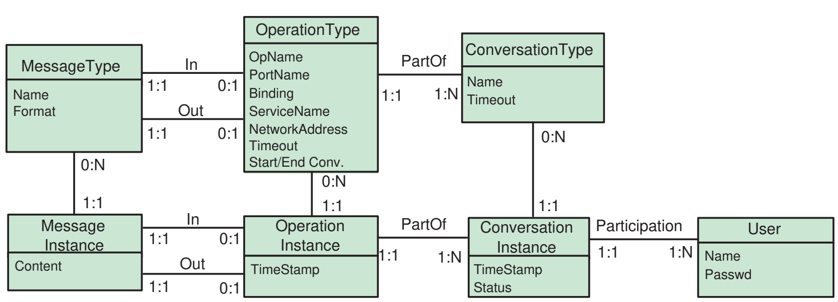

Figure 3: E-R diagram for user interactions with Web services.

Our solution provides (i) the data model used for recording the actual occurrences of conversations, operations, and exchanged messages, (ii) the WebML primitives describing WSDL operations, and (iii) the mapping languages for assembling data of the WebML application into XML messages and for disassembling the XML format of an output message into (E-R) data of the WebML application (not included here for space reasons).

Web Services Data Model. To record the interactions between the Web application and the Web services, we use a data schema, depicted in Figure 3. This schema is centered on the OperationInstance entity, which is described by the related OperationType; an operation may have one outgoing message and/or one incoming message, whose content is described as a text. Operations are part of conversations. Each conversation is a sequence of operations fulfilling a common goal, e.g., browsing a product catalog to buy some product. A conversation refers to a given user, and a user can participate to multiple conversations.

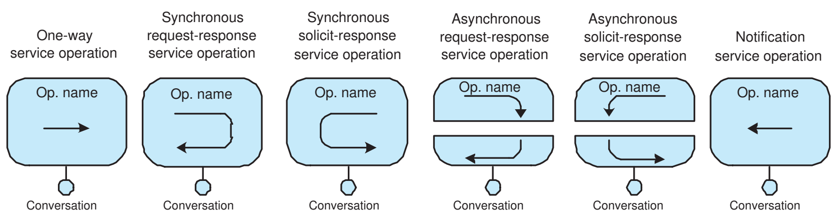

Figure 4: WebML primitives for communicating with Web services.

Web Services Hypertext Primitives. We propose six new graphical WebML operations reflecting the WSDL specification, shown in Figure 4. Operations belong to four categories: *one-way* (initiated by the user, by issuing an outbound message), *notification* (initiated by the service, by sending an inbound message), *request-response* (initiated by the user, with one outbound message followed by one inbound message), and *solicit-response* (initiated by the service, with one inbound message followed by one outbound message). Two-message operations are represented as round-trip arrows; arrows from left to right correspond to outbound messages (i.e., messages sent out by the Web application); asynchronous operations are represented by two superposed halves, one for each constituent message. The two parts of an asynchronous operation have independent lives, since they are triggered by different events and lead to different possible actions; therefore, each part can be considered as an independent WebML operation.

As any other operation, these new ones have input and output links, which are used to transfer context and to navigate within the hypertext. Four operations are triggered by the navigation of their input links: synchronous request-response, one-way operations, the upper half of an asynchronous response-request, and the lower half of asynchronous solicit-response. In all cases, input parameters and state information are assembled into messages, sent to the remote Web service; encoding of WebML parameters and state information into XML is specified by means of a simple mapping language. In the case of a synchronous request-response, the user waits until the response message is received, then continues navigation as indicated by the operation's output link. In the other three cases, the user resumes navigation immediately after the message is sent. Four operations are instead triggered by the reception of an external message: notification, synchronous solicit-response, the upper half of asynchronous solicit-response, and the lower half of asynchronous request-response. None of these operations can have output links leading to pages, since we cannot force the user to browse to a given page following an external event; instead, these operations can be linked to other WebML operations, carried out by the Web application without requiring the user's participation. The encoding of XML messages into state information is specified by means of a simple mapping language.

To express the boundaries of a conversation, the first operation of a conversation is tagged with a StartConversation property and the last operation is tagged with an EndConversation property. Eventually, a disposal mechanism can be provided, to terminate conversations when a timeout occurs. The execution of service-related

operations causes the implicit update of the data: a new OperationInstance is created with the proper parameters; it is connected to the OperationType and to the current Conversation by a PartOf relationship; for each message sent or received, Message instances are created and connected to the relative Operation instance; all operations sending a message include rules to construct the XML message from relational data; operations receiving a message include the symmetric rules for extracting data from the XML content of the message, to create new instances of the data model.
We now illustrate a Web application that includes interactions with Web services; Figure 5 describes some pages of the Web site of a travel agent offering to its costumers the possibility of getting weather forecasts and making hotel reservations. Both functionalities are provided by remote services, which the site integrates.

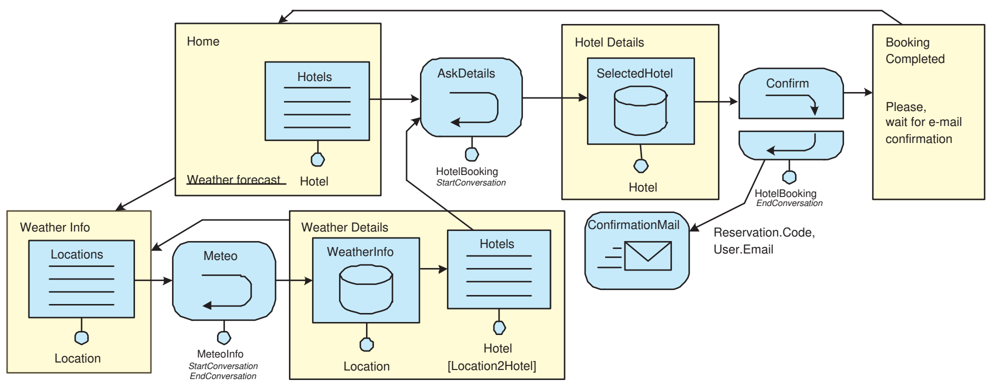

Figure 5: Hypertext model for the purchase application.

When a customer enters the Web site, he can choose a hotel from an index. More details about the chosen hotel are supplied by the AskDetail service, a synchronous request-response operation whose input message contains the hotel identity and whose output message contains details about the hotel. If the customer decides to make a reservation, he invokes the asynchronous request-response Confirm service; after the request, the buyer is immediately taken to the BookingCompleted page. When the response message is received, the reservation code is extracted from the output message of the Confirm operation, and an e-mail notification is automatically sent to the costumer by using the SendMail WebML operation. The user can also browse an index of locations and then request weather information for that location; such information is displayed in the WeatherDetails page, together with an index of hotels in that location. Users can next ask details about those hotels, and confirm reservations.

4 Extensions for modeling workflows

Pure workflow managements systems aim at imposing strict behavior constraints to their users, guiding them in the accomplishment of a particular objective; instead, conventional Web applications leave users completely free of of choosing their next activities in a navigation. We combine hypertext and workflow into what we call *workflow-driven hypertexts*: these hypertexts represent well-defined situations between the two ends of the spectrum, e.g., hypertexts representing well-defined processes, possibly with complex activity structure and execution constraints, but supported by the HTTP protocol, where each user can freely detach from the workflow and start independent browsing sessions. The main ingredients for the high-level specification of workflow-driven hypertexts are:

- A workflow data model for representing hypertext and workflow concepts, for managing access control, tracking the workflow activities and supporting logging and auditing.
- Constraint modeling primitives, for representing the execution sequence of activities. In particular, WfMC proposes: Sequences of activities, AND-split and join (a thread of control splits into two or more threads,

and joins when all the activities have been performed), OR-split and join (a thread of control makes a decision upon which branch to take, and joins when the activities of one of the branches terminate), Iteration (repetitive execution of one or more workflow activities), Pre-conditions and Post-conditions. The WebML primitives that we add enable the high-level specification of processes but are then mapped into lower-level hypertext specifications; conditional execution is modeled by *switch* operations in the hypertext.

Workflow Data Model. The workflow data model (Figure 6) includes the entities and relationships needed for managing one or more workflows. At the type level, it comprises Groups, ActivityTypes, and Processes; at the instance level, it contains Users, ActivityInstances, and Cases. ActivityInstances and Cases describe the actually occurred activities and include information about their start and end time. Application data can be added and connected at will to the Workflow data model entities by means of semantic relationships. A default one-to-many relationship may connect each ActivityInstance to application entities, with the meaning that entity instances are "assigned-to" a given activity. The default assignment of a given object (e.g., a Document) to a given activity instance indicates that the object is used (or will be used) by the activity instance (e.g., it will be created, read or modified by that activity). The default relationship linking an applicative entity to the ActivityInstance entity is denoted by adding a "W" symbol to the applicative entity in the E-R diagram (see Figure 6). If the default relationship is used, unit selectors over entities may be simplified with the shorthand shown in Figure 7, where the "W" symbol denotes a default predicate linking the entity to the relevant activity instance.

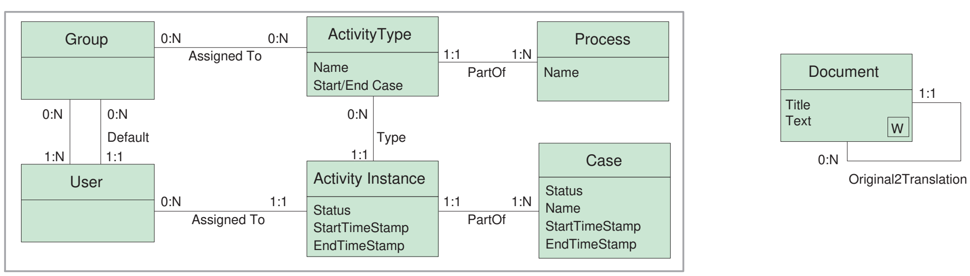

Figure 6: Workflow Data Model

Workflow Management Hypertext Primitives. The minimal extension required for supporting workflows includes operations for starting and ending activities. Operations *StartActivity* and *EndActivity* indicate the events at which operations start and terminate; their execution causes several updates to the workflow data.

The first activity of a process performs the creation of a new case; termination of either one of the "last" activities of a process may cause the termination of a case. Workflow specification also requires generic switch units for expressing conditional navigation. A switch unit evaluates, for each output link, a condition based on the values received by its input links, and enables the navigation of the first output link whose condition is true. This permits the modeling of workflow constraints, like AND and OR splits, the corresponding joins, iterations, pre- and post-conditions; we omit a further presentation of switch units for brevity. Additional units and default notations simplify the specification of workflows. Assign units connect application objects to the ActivityInstance they refer to using the default relationship, as discussed in the previous paragraph. A special "W" tag on a unit extracting application data expresses a conjunctive clause in the unit's selector, for selecting the data connected to the current ActivityInstance. Information about current activities and cases is available as WebML global parameters.

Figure 7 shows a simple application of the workflow concepts, to model a process for the creation and translation of documents. At high-level, the process is composed by two activities, creation and translation of documents, to be executed one after the other. The two activities are implemented as two hypertexts for the two different users involved in the process. The hypertext exploits the StartActivity, EndActivity and Assign operations. In

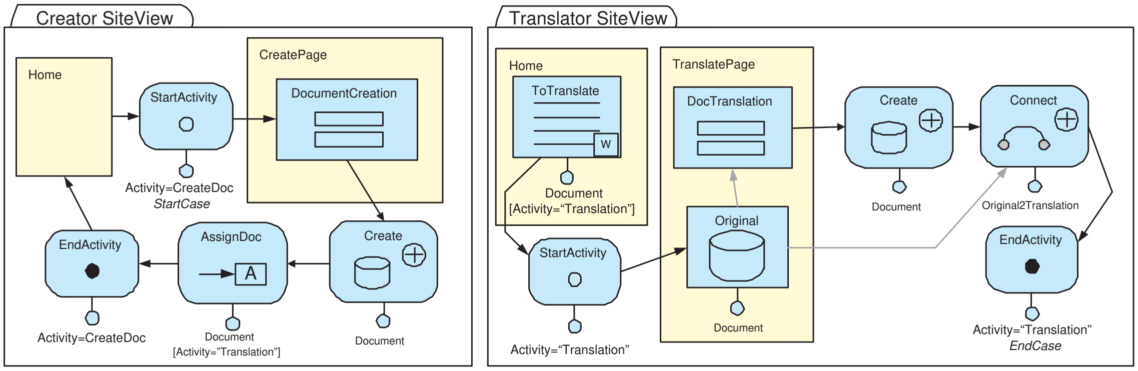

Figure 7: Hypertext implementing a simple workflow example.

particular, when a Creator enters the Home page of his/her siteview, he can start creating a new document by clicking on the proper link. The Create Document operation is the first of the process, thus the StartActivity is marked also as StartCase. The activity consists of the submission of a new document through a form in the CreatePage. The submission triggers the creation of the new document and the assignment to the Translation operation.
In his/her Home page, the translator can choose a document to translate, among the documents assigned to the Translation activity. Then, he can submit the new translation, which will be created as a new document, and connected to the original version. Once done, the activity and the current case are ended.

5 Workflows and Web services at work

The extensions proposed for Web services and workflows increase the expressive power of WebML and support the modeling of enhanced Web applications, incorporating Web services conversations and orchestrations. In fact, simple Web services conversations can be achieved through the use of the proposed Web services data model and hypertext primitives (as shown by the example in Figure 5). More complex conversations, involving branching on condition testing, parallel executions of Web services calls, and so on, can be obtained by integrating the two proposed extensions. In particular, it is always possible to define activities whose implementation consists of Web services calls; such activities, as parts of a complex process, can then be structured according to an arbitrary workflow, and can be executed depending on pre- or post-conditions.
Conclusions. This paper shows how visual specifications of Web applications can be integrated with Web services and workflows. These two extensions are orthogonal and require the introduction of new operations, which can be accomplished in a very natural way due to the extensibility of WebML. Experimentation is ongoing in the context of the WebRatio Web modeling and code generation tool (http://www.webratio.com).

References

[1] Stefano Ceri, Piero Fraternali, Aldo Bongio: Web Modeling Language (WebML): a modeling language for designing Web sites. *WWW9/Computer Networks*, 33(1-6), 2000: pp. 137-157.

[2] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera: Designing Data-Intensive Web Applications. Morgan-Kaufmann, December 2002, to appear.

[3] Stefano Ceri, Piero Fraternali, Maristella Matera: Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing*, 6(4), July-August 2002, pp. 20-30.