Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages

Patrick Freudenstein[1], Jan Buck[1], Martin Nussbaumer[1], and Martin Gaedke[2]

[1] University of Karlsruhe, Institute of Telematics,
IT Management and Web Engineering Research Group,
Engesserstr. 4, 76128 Karlsruhe, Germany
{freudenstein, buck, nussbaumer}@tm.uka.de

[2] Chemnitz University of Technology, Faculty of Computer Science,
Distributed and Self-organizing Computer Systems Group,
Straße der Nationen 62, 09107 Chemnitz, Germany
gaedke@informatik.tu-chemnitz.de

Abstract. The requirements for Web applications concerning workflow execution, interaction, aesthetics, federation and Web service integration are steadily increasing. Considering their complexity, the development of these "rich workflow-based Web applications" requires a systematic approach taking key factors like strong user involvement and clear business objectives into account. To this end, we present an approach for the model-driven construction and evolution of such Web applications on the basis of workflow models which is founded on Domain-specific Languages (DSLs) and a supporting technical framework. We describe our approach's core DSL for workflow modeling which supports various modeling notations like BPMN or Petri nets and outline a set of DSLs used for designing workflow activities like dialog construction, data presentation and Web service communication. In conclusion, rich workflow-based Web applications can be built by modeling workflows and activities and passing them to the associated technical framework. The resulting running prototype can then be configured in detail using the presented DSLs.

Keywords: Web Engineering, Workflow, Domain-specific Languages, Reuse, Evolution, Web Services, SOA, EAI

1 Introduction

The World Wide Web has evolved from a decentralized information medium to a platform for basic e-commerce applications. Currently, the next step in its evolution cycle towards a platform for sophisticated enterprise applications and portals with strong demands regarding workflow execution, rich user interaction, aesthetics, and strong Web service integration is taking place [16, 17]. Especially in the context of Enterprise Application Integration (EAI), Enterprise Information Integration (EII) or Business-to-Business (B2B) scenarios, these workflow-driven Web applications are gaining more and more importance. In order to cope with the immense increase in

these Web applications' complexity and their permanent evolution, a dedicated engineering methodology is required.

Besides specific requirements resulting from this new type of applications, a suitable engineering approach must also consider key factors like strong user involvement and clear business objectives arising from a project management's perspective. Their strong influence on a project's success was proved in comprehensive studies [20] and taken on in agile software development methods [2], being reason enough to soundly integrate them in today's Web engineering methods.

Facing these challenges, we present an evolutionary approach for the model-driven construction of rich, Web service-based Web applications on the basis of workflow models. The approach is based on our previous work, namely the WebComposition approach [8], the WebComposition Service Linking System (WSLS) [7] and our latest approaches towards DSL-based Web Engineering [14, 15]. By providing dedicated Domain-specific languages (DSLs) and an underlying technical framework, stakeholders and domain experts with diverse backgrounds and knowledge are enabled to directly contribute to the development effort. They can easily understand, validate and even develop parts of the solution which in turn leads to a much more intense collaboration and lowers the possibility of misunderstandings.

In section 2, we introduce a business process from a real-world scenario within a large-scale EAI-project to which we will refer to throughout the paper. We elaborate the particular requirements a systematic engineering approach for the described problem scope must fulfill. Section 3 gives a comprehensible overview of our evolutionary, DSL-based engineering method. In section 4, we describe the core DSLs for workflow modeling in detail and outline supporting DSLs for rich dialog construction, data presentation and Web service communication. Moreover, we present the supporting technical framework being able to interpret a workflow model and to assemble a corresponding Web application. Based on the presented scenario, we show exemplarily how a workflow modeled in e.g. BPMN notation and enriched with little annotations can directly be transferred into a running Web application. In section 5, we give an overview of related work. Finally, section 6 concludes the paper and outlines future work.

2 Challenges with Developing Workflow-based Web Applications

In the following, we first present a real-world scenario from a large-scale EAI project that serves as a running example throughout the paper. Subsequently, we introduce a general core set of requirements an engineering methodology for the systematic construction and evolution of modern workflow-based Web applications should meet.

2.1 The KIM Project - An Example EAI Scenario

We have been collaborating in the project "Karlsruhe's Integrated InformationManagement (KIM)" [13], a university-wide EAI-project, for several years now. One of the main challenges in this project is the extraordinary decentralized organizational structure of a university.

Technical challenges: On the one hand, there exists a huge diversity of heterogeneous IT systems which have to be integrated in order to enable a uniform access to information. In order to cope with these integration challenges, the KIM project is founded on a multi-layered Service-oriented Architecture (SOA). Therein, canonical Web service wrappers provide homogeneous access to existing heterogeneous legacy systems. These Web services are then orchestrated to realize value-added functionalities which are also exposed via Web service interfaces. Finally, the portal layer comprises mainly Web portals providing a centralized user interface for accessing the highly distributed Web services.

On the other hand, business processes are mostly spanning several organizational units and IT systems, thus suffering from media discontinuity issues. To improve the efficiency and quality of these processes, there is an urgent need for a support platform allowing for their integrated and uniform execution.

Communication challenges: Besides these technical aspects, we found communication problems making up the second major problem area. Stakeholders belong to different faculties and departments with entirely different education and professional background. Hence, when specifying business processes with stakeholders from all over the university, each group uses its own "language". For example, some of them prefer Petri nets as a means of communication as they play a major role in their research context. Others favor the Business Process Modeling Notation (BPMN) [21] or UML Activity Diagrams for the same reason. And people with a background in humanities often like a notation in natural language better. However, assuring efficient, non-ambiguous and intense communication is especially in phases like requirements engineering and conceptual design a key factor [20].

2.2 The Master Thesis Business Process Example

One of our main goals within the KIM project was the development of a Web portal for all students of the university serving as a uniform access point to all study-relevant information and business processes. Within this paper, we will focus on a workflow-driven feature which supports the complete Master Thesis business process for all involved parties: The student, the advisors, the examination office and the library.

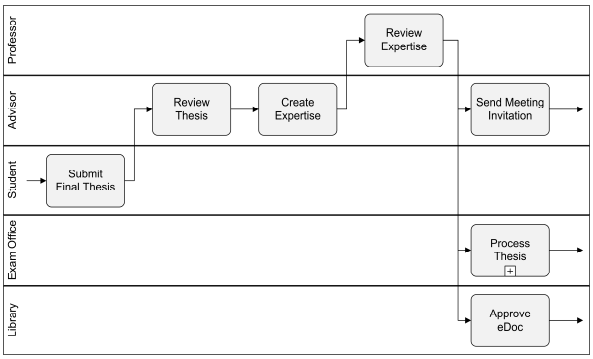

Fig. 1. Excerpt from the Master Thesis business process

Fig. 1 shows an excerpt from the Master Thesis business process modelled in BPMN. We chose BPMN as the involved stakeholders found this notation rather intuitive. However, in the following chapters, we will also outline how other notations, e.g. Petri nets, can be employed within our approach. Our example starts with the student submitting the final version of her thesis. Next, the advisor downloads and reviews it and submits her expertise. The associated professor reviews the expertise, which is then submitted to an existing exam management system via a Web service wrapper and further processed by the examination office. The university library approves the electronic thesis document, usually a PDF file, whereupon a dataset for the thesis is created in the library's central catalogue.

This process excerpt is a typical example found in advanced workflow-driven Web applications, especially in EAI, EII and B2B scenarios. It comprises different organizational units and roles, it contains typical building blocks like complex dialogs, data rendering and Web service communication, and various stakeholders are involved in the aspired Web application's specification. Moreover, due to the permanent restructurings taking place in the context of the Bologna Process [6] as well as the current merger of the University of Karlsruhe towards the "Karlsruhe Institute of Technology (KIT)", our business processes underlie frequent changes.

2.3 Requirements for a Workflow-Driven Web Engineering Methodology

From the general requirements for Web Engineering methods found in literature (e.g. [5, 9]) as well as based on our experiences in real-world projects, we identified the following requirements to be particularly important for a methodology targeting the construction and evolution of workflow-based Web applications. While the first three requirements concern the development process and the methodology itself, the last three aim at technical characteristics of modern workflow-based Web applications.

Agility & Evolution: Web applications in general and workflow-based Web applications in particular underlie a continuous evolution due to frequent changes, e.g. adjustments in the business process' structure, integration of new partners or presentational changes. Thus, agility in terms of supporting short revision lifecycles and the efficient adoption of such changes is essential. To this end, a model-driven approach seems to be a good solution as it allows for comparatively easy changes in the models which are then automatically propagated to the actual implementation. However, assuring consistency between models and implementation is crucial.

Reuse: With respect to requirements from the fields of evolution support, development efficiency and software quality, the systematic reuse of all kinds of artifacts throughout the development process plays an important role. Regarding workflow-based Web applications, especially the reuse of workflow models in whole or part as well as typical workflow building blocks, like e.g. dialogs, Web service communication and data rendering is of great interest. Thus, an engineering method should address reuse as a guiding principle throughout the development process.

Strong Stakeholder Involvement: In order to assure clear business objectives and to avoid misunderstandings between the developers and the business, stakeholders should be strongly incorporated in the development process. Especially regarding workflow-based Web applications, the future end-users know the underlying business

processes best. Thus, a dedicated engineering methodology must take into account the great diversity of stakeholders with different backgrounds and skills. Therefore, the methodology should allow for dedicated modeling languages and notations hiding unwanted complexity and being tailored to specific stakeholder groups [14]. Moreover, the ability to provide running prototypes from the very beginning of the development process further supports the communication between the end-users and the developers. Discrepancies between the requirements and the realization can be identified in the early stages of the development process and cost-efficiently resolved.

Rich User Interfaces: As dialogs play a dominant role in workflow-based Web applications, their usability has a great influence on how efficiently process participants can complete their tasks and thus contribute to the business process. Due to the increasing complexity of these tasks and the underlying data models, rich user interfaces supporting the users by reducing the cognitive overload are required. Therefore, these dialogues should be highly dynamic and offer guidance by e.g. showing only relevant options and providing immediate feedback and hints. Thus, their usability can be considerably improved [13]. Beyond that, aspects from the field of accessibility, i.e. providing accessible interfaces for people with disabilities have to be considered [22], especially in the public sector due to recent legal regulations.

Federative Workflows: Workflows based on business processes are - in contrast to simple page flows - long-running and affect different people or roles respectively. Advanced workflow scenarios (e.g. B2B) even span multiple companies. This means, they involve people from different organizations and rely on multiple, distributed information systems. The integration of these systems is usually realized via Web service interfaces. Thus, supporting long-running and federative workflows as well as comprehensive Web service support are key requirements.

Multimodal Participation: In advanced workflow scenarios, e.g. in supply chain management, parts of a workflow take place outdoors or away from computers. Then, process participants must be able to collaborate with other devices, e.g. PDAs or smart phones. Beyond that, some tasks are better conducted in dedicated, task-specific applications, e.g. a spreadsheet application. Thus, even though trying to offer one integrated, browser-based user interface is a desirable objective, workflow-based Web applications must also allow for completing tasks off the browser.

3 An Evolutionary DSL-based Engineering Approach

In this section, we give an overview of our evolutionary engineering methodology for the model-driven construction of workflow-based Web applications starting from business process models. The details of the methodology as well as the associated technical framework will be explained in section 4. Both were designed and implemented with strong adherence to the requirements identified in section 2.

3.1 The Workflow DSL

The model-driven construction is based on previous work in the fields of DSL-based Web Engineering [15] and thus is founded on a dedicated Workflow DSL as a core

element. A DSL can be seen as programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. By providing various graphical notations and accompanying editors, each of them being as intuitive as possible for a particular stakeholder group, the usability of a DSL can be further improved. According to this definition, the Workflow DSL is an executable specification language for workflow-based Web applications which allows the use of various graphical notations known from the Business Process Modeling field, e.g. BPMN, Petri Nets, UML activity diagrams etc. as well as custom notations. By providing stakeholder-specific notations according to their individual skills and preferences, stakeholders can easily understand, validate and even specify parts of the solution being constructed. Following our DSL-based Web Engineering concept, the Workflow DSL consists of three core elements:

Domain-specific Model (DSM): The DSM represents the formal schema for all "DSL programs" that can be described with the DSL. With respect to our requirement to support various process modeling notations, the DSM can also be seen as a "Process Intermediate Schema", representing an (as far as possible) common denominator of multiple existing process modeling languages. Beyond business process information, the DSM comprises dedicated modeling constructs necessary for the transition from a pure business process model to a running workflow-based Web application. We chose the XML Process Definition Language (XPDL) [19] as a basis for the DSM. Serving both as an interchange format for process definitions and as a definition language for executable workflow specifications (including human interaction aspects) belonged to the major design goals of XPDL, making it an ideal foundation for our DSL. The extensibility mechanisms provided by XPDL were used to shape our DSM.

Domain Interaction Model(s) (DIM): Based on the DSM, a DIM comprises a dedicated (graphical) notation being as intuitive as possible for a particular stakeholder group. By using a DIM, stakeholders can understand, validate and even create DSL programs without being confronted with complicated source code.

Within the Workflow DSL, multiple DIMs for various stakeholder groups could be defined. Thereby, a DIM could either be derived from a well-known business process modeling notation like e.g. BPMN or Petri nets, or defined from scratch based on a custom notation. According to different incremental stages of the Web application construction process, DIMs can also cover only parts of the DSM. Accompanying editors support stakeholders in creating DSL programs based on a DIM notation.

Solution Building Block (SBB): A SBB is a software component being capable of executing programs developed with the DSL. Therefore, the SBB does not generate code but rather adapts its behavior according to a given DSL program. Thus, the SBB of the Workflow DSL can be configured with an XML-based specification of a workflow-based Web application, i.e. a Workflow DSL program. Thereupon, it constructs at runtime an associated workflow-driven Web application prototype. This immediately executable prototype employs SBBs from other DSLs to realize workflow activities like e.g. dialog construction, data presentation or Web service communication. These SBBs were initialized with a minimum configuration set derived from the workflow model and can then be configured in detail using the associated DSLs or their DIMs respectively.

3.2 The DSL-based Process Model

Fig. 2 gives an overview of our methodology's underlying evolution- and reuse-oriented process model as well as the involved roles. It is based on the WebComposition approach [8] and consists of three phases in a continuous evolution.

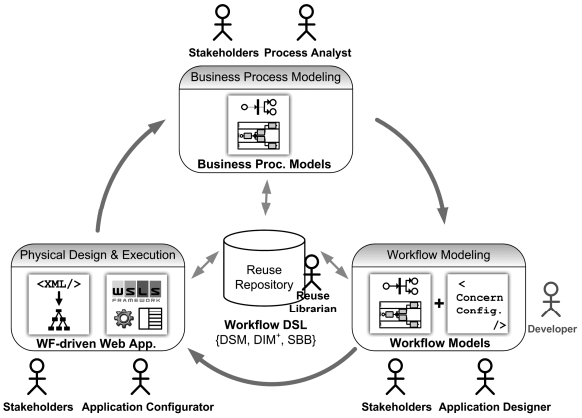

Fig. 2. Overview of the evolutionary, DSL-based engineering approach

Business Process Modeling: In this first phase, the business process to be realized by the workflow-based Web application is modeled using pure business process modeling constructs. Thereby, stakeholders representing the involved process participants and knowing the business process best as well as a process analyst supporting the modeling itself are involved. Moreover, the 'Reuse Librarian' role advises the modeling team regarding possibilities for reusing existing process models in whole or part. The resulting business process model is created by employing adequate DIMs and associated editors from the Workflow DSL.

Workflow Modeling: In this phase, the business process model from the previous phase is supplemented with workflow execution-relevant information, the 'Concern Configuration'. Thereby, to each activity in the process model a corresponding activity building block is assigned, whereby each activity building block has an associated DSL for its configuration. In our experience, a small core set of building blocks for dialog construction and processing, data rendering and Web service communication was sufficient for the most cases. If no suitable building block exists, the 'Developer' role designs and implements a new one. In addition to selecting the activity building block type, a minimum set of configuration information has to be provided. This set contains properties from the different concerns of a Web application, i.e. data, navigation, presentation, dialog, process, and communication. This minimum configuration set assures the automatic setup of a running prototype of the workflow-based Web application in the next phase. The detailed configuration of the activity building blocks is usually performed in the next phase by means of the associated DSLs. Like the previous phase, the Workflow Modeling phase is again conducted by stakeholders who are supported by an application designer and the reuse librarian. The application designer is experienced in workflow modeling and knows the activity building blocks, the associated DSLs and the required minimum

configuration sets. She assists the modeling team in related issues. The reuse librarian advises the team concerning the reuse of existing Concern Configurations and activity building blocks from the reuse repository. The Workflow modeling phase is also supported by adequate Workflow DSL DIMs and accompanying editors, usually the same as used in the previous phase but extended with Concern Configuration facilities. The result of this phase is a valid Workflow DSL program in form of an XML document, whereby process structure information and Concern Configuration are loosely coupled, thus easing reuse and evolution.

Physical Design & Execution: This phase deals with the transformation of the DSL program into a running prototype of the aspired workflow-based Web application. Therefore, the DSL program is passed to the Workflow DSL's SBB, which configures an associated workflow-driven Web application prototype. This prototype can then either be configured in detail using the activity building blocks' associated DSLs or directly be used for creating and processing workflow instances. The WebComposition Service Linking System (WSLS) [7] serves as our approach's technical platform and facilitates the assembly and configuration of SBBs. As in the previous phases, stakeholders can strongly participate in this phase, assisted by an 'Application Configurator' role who is experienced in WSLS and its configuration capabilities as well as the activity building blocks' associated DSLs.

Evolution: In case of changing or new requirements, our method provides strong support for adopting changes, either in the business process model or the Concern Configuration or both. Changes in the business process can easily be performed in the Business Process Modeling Phase while keeping the Concern Configuration in the Workflow Modeling phase unchanged. Changes in the Concern Configuration can either be performed on model-level in the Workflow Modeling phase or directly in the Physical Design & Execution phase by using appropriate DSLs. Our approach and the technical platform preserve model consistency throughout all phases.

4 The Workflow DSL Approach Applied - Realization Details

This section describes our approach in detail based on the example excerpt from the 'Master Thesis' business process presented in section 2 (Fig. 1), which could be the output of the Business Process Modeling phase and serves as a starting point. To ease the understanding of the following subsections, section 4.1 outlines a selection of activity building blocks for the realization of workflow activities. Section 4.2 focuses the Workflow Modeling phase and shows how an appropriate Concern Configuration could look like. We present our XPDL-based DSM, an adequate DIM editor and excerpts from the resulting DSL program. Finally, section 4.3 covers the Physical Design & Execution phase, and presents the approach's underlying technical platform for constructing, configuring and executing workflow-based Web applications.

4.1 Activity Building Block DSL Catalogue

A major design goal of our activity building blocks was that one activity from a business perspective can be realized by one activity building block, and must not be

split up into several activities from a system perspective. Thereby, the business process model's structure can be kept throughout the construction process, easing the collaboration with stakeholders. Thus, as we especially aim at service-based Web applications, the Web Service Communication building block can be integrated with other building blocks. Therefore, the DSLs allow for the loosely coupled integration of external code in their programs which is forwarded at runtime to the appropriate SBB. For example, a Dialog Construction DSL program can thus submit a filled form to a Web service.

Web Service Communication: This DSL allows the specification and execution of Web service calls. The DSM is represented by an XML Schema which defines, amongst others, elements for specifying the Web Service endpoint, WSDL URL, operation name, input parameters and security policy information based on the WS-Security Policy standard (to be submitted to OASIS). The DIM editor is realized in form of a property editor. The SBB generates a SOAP message according to the DSL program, sends it to the Web service and returns the received response.

Dialog Construction: This DSL is used for the specification of highly interactive and dynamic dialogs. The DSM is based on the W3C XForms standard. We defined a DIM notation on the basis of the XForms user controls as well as Petri net-based structures for modeling the dialog's dynamic behavior. The SBB is capable of automatically creating an XForms-based form prototype from an XML Schema specification or a WSDL document. Moreover, it renders and processes the XForms document by means of a JavaScript library. A browser-based DIM editor allows the detailed design of the form, including the verification of accessibility guidelines [12] .

Data Presentation: This DSL addresses the presentation of data. The DSM provides elements for referencing the data to be displayed as well as an XSL document specifying the data transformation to the desired output format. If a DSL program contains no reference to an XSL document, the SBB automatically generates a prototypical XSL transformation into XHTML. A browser-based DIM editor allows the detailed design of XSL stylesheets on the basis of Cascading Style Sheets (CSS), again with integrated verification facilities for accessibility guidelines.

4.2 Workflow Modeling

The Workflow DSL's DSM and the associated DIMs constitute the basis for the Workflow Modeling phase. The DSM is based on XPDL which we extended by dedicated modeling elements as depicted in Fig. 3.

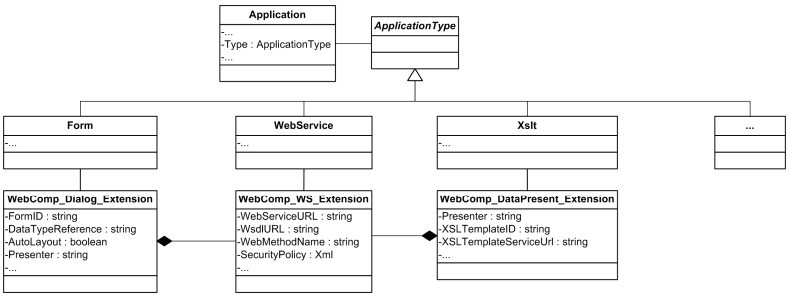

Fig. 3. WebComposition XPDL extensions appended to existing ApplicationTypes

The XPDL schema contains so-called “ApplicationTypes” for modeling applications used for performing activities. We extended the XPDL ApplicationTypes “Form”, “WebService” and “Xslt” in order to integrate properties for the configuration sets required by the activity building blocks presented above. Thereby, the “WebComp_Dialog_Extension” serves for the Dialog Construction building block, the “WebComp_WS_Extension” for the Web Service Communication building block and the “WebComp_DataPresent_Extension” for the Data Presentation building block. Beyond the minimum configuration set, we are currently working on a full coverage of the DSLs’ modeling elements in form of attributes. Thus, fully configured building blocks resulted from prior Physical Design & Execution phases and stored in the Reuse Repository could be already reused in the Workflow Modeling phase.

To provide support during the Workflow Modeling phase, we adapted Microsoft Visio as a visual editor for a BPMN-based DIM notation (Fig. 4). The editor provides shapes according to the BPMN notation (left pane) as well as a property editor for annotating the Concern Configuration, i.e. the selection of an activity building block and its corresponding minimum configuration, to process activities (bottom pane). In the picture, the ‘Create Expertise’ activity is currently modeled as a Dialog Construction building block and a data schema for the dialog as well as attributes regarding the Web service the form shall be submitted to are provided.

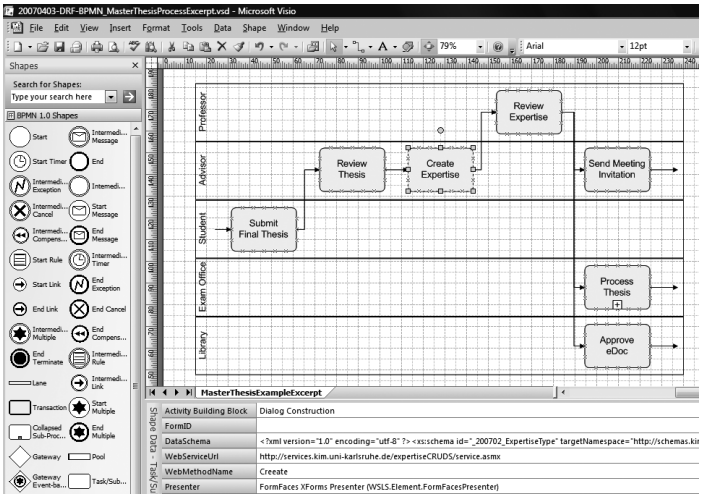

Fig. 4. A BPMN-based DIM Editor in the Workflow Modeling Phase

Having completed the Workflow Modeling, the workflow model can be exported as a valid Workflow DSL program. Thereby, the mapping of the BPMN notation and the annotated Concern Configuration to a DSM-conform DSL program is performed. The mapping of BPMN symbols to XPDL elements is described in the XPDL specification [19]. Regarding other graphical notations, additional mappings have to be defined. We are currently working on a DIM based on the Petri Net notation and to be supported by the process modeling tool INCOME [11]. As our support for various notations aims primarily at improving stakeholder collaboration by providing intuitive notations, their usability and simplicity is more important than covering even the most

complex semantic aspect of a DIM notation's underlying modeling language. Having analyzed a great variety of process models from the KIM project, we found that XPDL provides a sufficient (and extensible) set of generic business process modeling elements, being suitable to accomplish a variety of DIM notations derived from popular modeling languages like e.g. Petri nets and UML Activity Diagrams.

The following code snippets are part of the resulting Workflow DSL program. Extract (1) shows the representation of the 'CreateExpertise' activity. According to the business process model shown in Fig. 1, it is assigned to the 'Advisor' role. The activity is linked to the 'CreateExpertiseDialog' application definition which defines a dialog according to the Concern Configuration shown in Fig. 4.

```
<xpdl:Activity Id="CreateExpertise">                                    (1)
  <xpdl:Implementation><xpdl:Task>
      <xpdl:TaskApplication Id="CreateExpertiseDialog">
        <xpdl:ActualParameters><xpdl:ActualParameter>
           Expertise</xpdl:ActualParameter>
        </xpdl:ActualParameters>
  </xpdl:TaskApplication></xpdl:Task></xpdl:Implementation>
  <xpdl:Performer>Advisor</xpdl:Performer>
</xpdl:Activity>
                                  ...
<xpdl:Application Id="CreateExpertiseDialog">                           (2)
  <xpdl:FormalParameters>
    <xpdl:FormalParameter Id="ExpertiseParams" Mode="OUT">
      <xpdl:DataType><xpdl:DeclaredType Id="ExpertiseType" />
  </xpdl:DataType></xpdl:FormalParameter></xpdl:FormalParameters>
  <xpdl:Type><xpdl:Form>
      <webComposition:DialogExtension>
        <DataTypeRef>ExpertiseType</DataTypeRef>
        <Presenter>WSLS.Element.FormFacesPresenter</Presenter>
        ...
         <webComposition:WSExtension>
            <WebServiceUrl>http://services.kim.uni-karlsruhe.
            de/expertiseCRUDS/service.asmx</WebServiceUrl>
            <WebMethodName>Create</WebMethodName>
            ... </webComposition:WSExtension>
      </webComposition:DialogExtension>
</xpdl:Form></xpdl:Type></xpdl:Application>
```

4.3 Physical Design & Execution – The Technical Platform

Fig. 5 depicts the approach's underlying technical platform, mainly consisting of the Workflow Web Service and the Workflow SBB running on the WSLS framework.

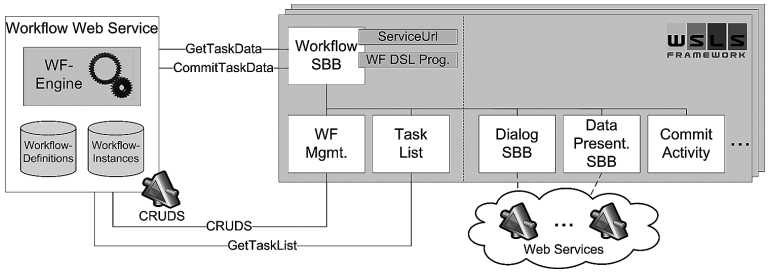

Fig. 5. Overview of the Technical Platform: Workflow Web Service and Workflow SBB

The Workflow Web Service has two core functions: First, it is used to manage workflow definitions, i.e. Workflow DSL programs, and workflow instances, via a *CRUDS* (Create, Read, Update, Delete, Search) interface. Second, it can be used to participate in a workflow by retrieving the current tasks for a particular role (*GetTaskList*) or the actual input parameters for a given task of a workflow instance (*GetTaskData*) as well as sending the results of a completed task back to the workflow instance (*CommitTaskData*). For the realization of these functions, the Microsoft Windows Workflow Foundation (WF) is used as workflow engine. When creating a new workflow definition, the Web service converts the process structure from the DSL program into an executable WF library which serves as input for the engine. By encapsulating these functionalities in a Web service, all kinds of clients from any platform can participate in a workflow across organizational borders.

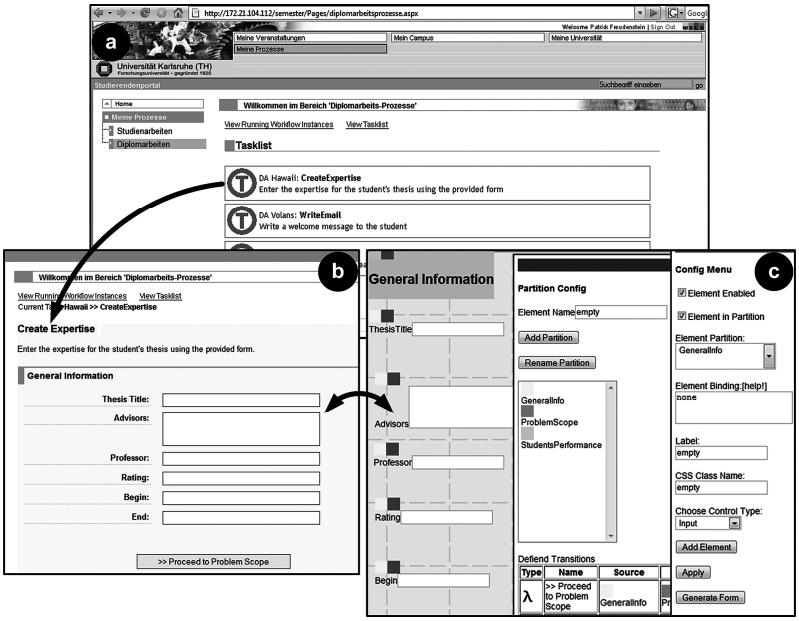

Fig. 6. (a): The Current User's Global Task List **(b):** The 'Create Expertise' Activity Dialog **(c)** Detailed Design of the Dialog using the Dialog Construction DSL's DIM editor

The WSLS framework supports the systematic development and evolution of Web applications by assembling and configuring components with respect to the 'Separation of Concerns' principle at runtime. It aims at realizing the 'Configuration instead of Programming' paradigm and thus at making the process of evolution faster and more flexible. The Workflow SBB is a WSLS component that can be configured with a Workflow DSL program which is then sent to the Workflow Web Service in order to create a new workflow definition. As all DSL programs are accessible via the Web service, other WSLS installations can easily retrieve them, thus enabling federation scenarios. The Workflow SBB uses the Concern Configuration information contained in the DSL program to instantiate and configure a child component for each process activity. Therefore, it employs the SBBs of the presented activity building blocks which possess automation features enabling fast prototypes with minimal

configuration as well as a 'Commit Activity' component. The latter is used for confirming the completion of tasks performed offside a PC like shipping a package. Furthermore, the Workflow SBB instantiates two default child components: 'Workflow Management' for starting and managing workflow instances, and 'Task List' (Fig. 6a) for displaying a cross-workflow task list for the currently logged in user. From this moment on, all child components can be configured in detail (at runtime) using the associated DSLs and the comprehensive WSLS configuration facilities. The changed configurations are propagated back to the DSL program to preserve consistency between physical design and the workflow model. Fig. 6c shows the detailed design of the 'Create Expertise' dialog using the Dialog Construction DSL's DIM editor. If the user selects a task, the Workflow SBB retrieves the task input parameters from the Web service and displays the child component associated with the task (Fig. 6b). After the completion of a task, the Workflow SBB sends the results back to the Web service which passes it to the workflow engine. Afterwards, the Workflow SBB retrieves the new task list for the current user and the current workflow instance and displays it or - in case of only one outstanding task - directly switches to the page containing the associated child component.

5 Related Work

Having recognized the increasing importance of workflow-based Web applications, established Web Engineering approaches like e.g. OOHDM [18], UWE [10], and WebML [3] were extended towards modeling workflow-based Web applications. In the following, we will point out the differences compared with our approach based on the requirements presented in section 2.

All of them are model-driven design approaches, i.e. they incorporate business processes in their various Web application models and thereof generate the Web application's code. Thus, they support agility and evolution in terms of adopting changes on a model-level. However, changes in the generated code are not propagated to the model-level and thus get lost when regenerating an application.

Model reuse is considered in the above mentioned approaches. However, reuse in terms of component reuse, i.e. reusing common activity building blocks, is not covered. Thus, the design models derived from business process models still require intensive design effort regarding the concrete realization of process activities. In [1], an advanced generic workflow modeling approach considering also application logic contained in activities is presented. Though, they focus primarily on data processing and navigation and leave out other concerns like presentation or interaction.

The strong involvement of stakeholders throughout the development process by providing dedicated notations for individual stakeholder groups has not been in the focus of current Web Engineering approaches yet. However, by means of model transformations, they could also support multiple process modeling notations. The integration of stakeholders in later stages beyond business process modeling has also to be investigated further. Audience-driven approaches, e.g. from DeTroyer [4], rather consider the future end-users, their characteristics and information requirements in their design process in order to develop user-centered Web applications.

Presentation modeling naturally plays a major role in most of today's Web Engineering approaches. However, only few of them, e.g. WebML, have also presented concepts concerning the modeling of dynamic and adaptive user interfaces required for reducing the cognitive overload caused by complex dialogs. The integrated consideration of accessibility guidelines in the development process is also an open issue in most approaches.

Regarding technical requirements like federative workflows and multimodal participation, most approaches provide only limited support. OOHDM, for example, mentioned the need for supporting federative workflows in their above cited paper's outlook. WebML and UWE already support long-running workflows with different roles, but so far their technical platform do not allow for federated participation scenarios, e.g. from other Web portals or multimodal access from diverse clients.

Beyond the only limited fulfillment of the above requirements, each of the existing Web Engineering approaches has other unique characteristics and ideas worth considering and learning from. WebML, for example, presented inspiring work in the field of exception handling and constraint modeling. Likewise, OOHDM's concepts for adapting business processes to given contexts and users is also an interesting topic not yet covered by our approach. The UWE approach, for example, is very strong in the field of meta-modeling and model-transformations. Their ideas and concepts serve as a valuable source for the transformations required in our approach.

6 Conclusions & Future Work

Facing the challenges found in the development and evolution of advanced workflow-based Web applications, we presented a methodology for their model-driven construction employing Domain-specific Languages. DSLs can define various modeling notations, each of them being as intuitive as possible for a particular stakeholder group and thereby improving communication and collaboration throughout all stages of the construction process. Dedicated software components can be configured at runtime with DSL programs and adapt their behavior accordingly.

Our approach places emphasis on evolution and reuse and enables rapid prototypes automatically derived from slightly annotated business process models. This is supported by a set of activity building blocks for the realization of rich dialogs, data presentation and Web service communication, each of them being able to work with a minimum configuration set derived from the workflow model. Associated DSLs allow their detailed design at runtime, thereby assuring consistency between models and implementation. The approach's technical platform provides strong support for evolution and reconfiguration and enables the execution of federative workflows and multimodal participation.

As the presented approach is the result of more than two years research, not all aspects could be described in detail. Thus, we are working on further publications focusing e.g. the formal transformations from different process modeling notations to our XPDL-based Process Intermediate Language and from there to the specification formats of various workflow engines. This is achieved by employing model transformation techniques like XSLT, QVT and ATL. Moreover, we will present the

details of our activity building blocks, especially the Dialog Construction DSL, in a separate paper. Beyond that, interesting extensions were given in section 5.

References

1. Barna, P., Frasincar, F., and Houben, G.J.: A Workflow-driven Design of Web Information Systems. in International Conference on Web Engineering. 2006. Palo Alto, USA
2. Beck, K., et al.: Manifesto for Agile Software Development - (2001): http://agilemanifesto.org/ (10.11.2006)
3. Brambilla, M., Ceri, S., Fraternali, P., and Manolescu, I.: Process Modeling in Web Applications. ACM Transactions on Software Engineering and Methodology (TOSEM), 2006. 15(4): p. 360 - 409
4. De Troyer, O.M.F. and Leune, C.J.: WSDM: a user centered design method for Web sites. Computer Networks and ISDN Systems, 1998. 30(1998)
5. Deshpande, Y., et al.: Web Engineering. Journal of Web Engineering, 2002. 1(1): p. 3-17
6. European Union: The Bologna Process - Web Site (2005): http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html (23.02.2006)
7. Gaedke, M., Nussbaumer, M., and Meinecke, J.: WSLS: An Agile System Facilitating the Production of Service-Oriented Web Applications, in Engineering Advanced Web Applications, S.C. M. Matera, Editor. 2005, Rinton Press. p. 26-37
8. Gaedke, M. and Turowski, K.: Specification of Components Based on the WebComposition Component Model, in Data Warehousing and Web Engineering,2002,IRM Press, p. 275-284
9. Kappel, G., Pröll, B., Reich, S., and Retschitzegger, W.: Web Engineering: The Discipline of Systematic Development. 1 ed. 2006: Wiley
10. Koch, N., Kraus, A., Cachero, C., and Melia, S.: Modeling Web Business Processes with OO-H and UWE. in 3rd Int. Worskhop on Web-oriented Software Technology. Spain, 2003.
11. Lausen, G., et al.: The INCOME approach for conceptual modeling and prototyping of information systems in the 1st Nordic Conference on Advanced Systems Engineering. 1987.
12. Luque Centeno, V., Delegade Kloos, C., Gaedke, M., and Nussbaumer, M.: Web Composition with WCAG in Mind.In 14th International World Wide Web Conference, 2005
13. Nielsen, J.: Forms vs. Applications, in Jakob Nielsen's Alertbox, 19.11.2005: http://www.useit.com/alertbox/forms.html (02.04.2007)
14. Nussbaumer, M., Freudenstein, P., and Gaedke, M.: Stakeholder Collaboration - From Conversation To Contribution. in 6. International Conference on Web Engineering (ICWE). 2006. SLAC, Menlo Park, California: ACM
15. Nussbaumer, M., Freudenstein, P., and Gaedke, M.: Towards DSL-based Web Engineering. in 15. International World Wide Web Conference (WWW). 2006. Edinburgh, UK: ACM
16. O'reilly, T.: What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software - (2005): http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (18.10.2005)
17. Phifer, G.: The Fifth Generation of Portals Supports SOA and Process Integration, in Gartner Reports. 2006, Gartner: Stanford, CT, USA
18. Rossi, G.H., Schmid, H.A., and Lyardet, F.: Customizing Business Processes in Web Applications. in 4th International Conference on E-Commerce and Web Technologies, 2003
19. Shapiro, R., et al.: XML Process Definition Language (XPDL) 2.0 Specification - (2005), Workflow Management Coalition
20. The Standish Group International: CHAOS Research - Research Reports (1994-2005): http://www.standishgroup.com
21. White, S.A.: Business Process Modeling Notation (BPMN) Specification - (2006), OMG
22. World Wide Web Consortium: Web Accessibility Initiative Homepage: http://w3.org/WAI/