Patrick Freudenstein, Martin Nussbaumer, Frederic Majer, Martin Gaedke: A Workflow-Driven Approach for the Efficient Integration of Web Services in Portals. In Proceedings of the the IEEE International Conference on Services Computing 2007 (SCC 2007), July 2007, Salt Lake City, USA.

A Workflow-Driven Approach for the Efficient Integration of Web Services in Portals

Patrick Freudenstein[1], Martin Nussbaumer[1], Frederic Majer[1], Martin Gaedke[2]

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

As Service-oriented Architectures (SOA) mature, an efficient approach for the integration of Web services in portals is required. This holds true especially in medium and large-scale SOA-based systems with a multitude of Web services to be made accessible to the users. The integration scenarios, i.e. the Web service-based features within a portal, are usually composed of complex sequences of user interaction and service communication, aggravating the need for an efficient integration solution. We present an approach for the business process-driven modeling of these scenarios in form of 'user interaction (UI) workflows' as well as a technical framework enabling the model execution within existing portal systems. The integration of Web services in a portal can thus be realized very efficiently by modeling UI workflows and configuring highly generic activity building blocks for dialog construction, service communication and data presentation.

1. Introduction

Within the layer model of Service-oriented Architectures (SOA), the presentation layer plays an important role by providing interfaces for the users [1]. The majority of client applications found in this layer are portals acting as central access points to the services - usually Web Services - of the underlying layers [13]. Therefore, portal components realizing the service access as well as the rendering of appropriate interaction and presentation structures are needed.

Analyzing the required components for the service integration in a portal shows that their requirements can be quite complex and span across a variety of functional aspects: presentation and interaction aspects as well as aspects in the fields of data and service communication. While simple integration scenarios comprise only a parameterized service communication followed by the presentation of the received data, much more complex user interaction sequences of dialogs, service communication and data presentation are found in practice.

Given this complexity and the emerging variety of Web services in medium and large SOA-based systems which have to be made accessible to the users, an efficient approach for the integration of services in portals is required. Today's portal systems offer only very limited facilities and concepts for the integration of Web services which are not appropriate for the complex integration scenarios faced in practice. Developing dedicated portal components for each single integration scenario turns out to be too cost- and time-consuming, aggravates operations and maintenance and the enforcement of quality standards, e.g. regarding corporate design or accessibility guidelines [18].

Thus, a highly generic and reuse-oriented approach is not only from the efficiency perspective, but also regarding aspects like quality, flexibility and evolution, a desirable and inevitable solution. In this paper, we present an approach for modeling the user interaction with Web services and introduce a technological framework for its application within existing portal systems. We consider integration scenarios as 'user interaction (UI) workflows' composed of generic activity building blocks. Each of these building blocks is tailored to a distinct integration aspect like service communication, dialog and data presentation and is implemented as a configurable software component. The UI workflows can either be derived from business process models, e.g. in Petri net or the Business Process Modeling (BPMN) notation, or, due to our simple and intuitive modeling notation, designed from scratch with strong stakeholder collaboration. Finally, the resulting UI workflow model is being executed by a generic portal component. Thus, realizing complex Web service integration scenarios in portals is reduced to composing highly configurable building blocks along a UI workflow.

Section 2 gives an overview of the state of the art in today's portal systems and current research approaches, elaborating the need and the requirements for an adequate solution. Section 3 presents the cornerstones of our approach: the UI workflow modeling notation and how UI workflows can be derived from business process models as well as a detailed definition of our generic activity building blocks. In this context, we present a complete set of configuration aspects required for each of these in order to enable their direct physical execution. Section 4 describes the technological framework for executing UI workflows in portal systems. The practical application of our approach based on an example from a large-scale Enterprise Application Integration (EAI) project is demonstrated in section 5. Finally, section 6 draws the conclusions and presents future work.

2. State of the art

As to Web service integration, today's portal systems are still in their infancy. Although the great majority of vendors claim comprehensive Web service support for their products, reality looks different. The existing approaches can be divided into two groups: Template-based (e.g. RedDot Live Server [14]) and data-centric approaches (e.g. Bea WebLogic [3], Microsoft Office Sharepoint Server 2007 [12]).

Template-based approaches represent the simplest form of Web service integration. Developers can predefine Web service invocations in form of SOAP messages which can be filled with parameters and sent at runtime. The necessary instructions for sending such a message and processing the result message are integrated in form of script code within the pages.

Data-centric approaches, which make up the second group, are slightly more advanced. Web services to be integrated have either to implement a certain interface or their interface has to follow a particular pattern. In the latter case, the concrete interface - both parameters and result schemas - of each Web service has to be described by means of a predefined XML format. In this group, Web services are considered more as data providers than as business capability providers. In the portal, the business objects encapsulated by the Web services can then be integrated via out-of-the-box portal components, usually in form of lists and simple detail views.

Beyond that, some systems like the IBM WebSphere Portal Server [17] ease only the development of dedicated portal components ("portlets") for each Web service to be integrated via generating appropriate proxy classes. As they do not provide any means for the efficient, (semi-) automatic integration of Web services and thus require the development of dedicated components for each single Web service, they are not considered any further.

In practice, integration scenarios, i.e. the interaction of users with Web services, are mostly complex sequences consisting of multiple steps. Considering, for example, a service which provides access to room information, an integration scenario realizing a room search feature within a portal could be as follows: Displaying a search form, running a search on the room data Web service, presenting a result list, requesting detail information for a selected search result from the room service and finally displaying the detail information. Such scenarios cannot be realized in an efficient way with today's portal system's concepts and approaches as described above. Both of the presented groups require a lot of manual work (predefining SOAP messages, processing response messages, scripting the user interface, describing parameters and response data schemas etc.). They do not consider reuse nor do they provide any means for enforcing quality guidelines regarding the user interface. Moreover, future adjustments are costly due to the strong interweavement of the user interaction workflow and the portal pages' source codes. Taking all this into account, the effort resulting from existing approaches is not reasonable, especially when considering medium- or large-scale SOA-based systems with a multitude of services and associated integration scenarios.

Beyond the examination of existing industry solutions, a further important group has to be considered when analyzing the state of the art: approaches from the Web Engineering research community (e.g. [4, 15]) dealing with the systematic construction of Web applications with particular consideration of Web-specific characteristics and requirements. These methodologies emphasize modeling and development aspects of modern Web applications, whereas they mostly act on the assumption that applications are built from scratch. Thus, existing Web applications and portal systems are usually not considered. Nonetheless, their ideas in the fields of process modeling and transformation in the Web Engineering context inspired our work. Regarding the consideration of existing Web applications and portal systems, the WebComposition approach [6] is an exception. It is based on the principles of evolution and Component-Based Web Engineering (CBWE) as well as the 'configuration instead of programming' paradigm [5]. Thus, it naturally considers compositional, integrative and federated aspects. With respect to the problem domain addressed in this paper - the efficient realization of complex Web service integration scenarios in existing portal systems - some of the WebComposition approach's core principles and concepts could be adopted for our solution.

3. Efficient integration with UI workflows and generic activity types

Analyzing a great variety of Web service integration scenarios in several large-scale EAI and SOA projects, we found so-called ‘user interaction (UI) workflows’ to be an ideal common denominator for modeling these scenarios. A workflow can be defined as “the computerized facilitation or automation of a business process, in whole or part” [7]. Our UI workflows comply with this definition as they facilitate the execution of a task within a business process by providing appropriate interaction structures to a user and managing the communication with underlying IT systems via Web services. However, in contrast to ‘normal’ workflows, UI workflows consider only a small part of a business process where *one* user interacts with the system to complete a particular task. Thus, they focus more on providing support for completing a task and less on controlling and running the whole business process from an overall perspective. When modeling business processes hierarchically with increasing degree of refinement, UI workflows make up the bottom layer. By describing the user interaction with IT systems through Web services, they represent - besides system-to-system interaction models - the highest degree of concretion within a business process model (cf. section 3.2).

Today, a great variety of business process modeling notations exists, e.g. BPMN, Petri nets, UML etc. In order to assure that our solution can be applied independently from the modeling language used as well as to provide a notation focusing only on the essential concepts of UI workflows, we chose Finite State Machines (FSM) as foundation. They provide a simple and intuitive notation, they can be mathematically defined and mappings from existing modeling languages to FSM models can easily be realized.

Modeling Web service integration scenarios in form of UI workflows with FSM, a user view (e.g. a search form) is represented by a state and the user navigation between views by triggering events (e.g. clicking on a button) corresponds to transitions. Moreover, our modeling concept comprises a set of generic activities for specifying the entry actions for the particular states. Therefore, we identified three elementary activity types found in the great majority of Web service integration scenarios: dialog construction, Web service communication, and data presentation. Having modeled a Web service integration scenario this way, it can be directly executed by a dedicated technical platform within a portal system (section 4).

In the following, we begin with clarifying the core ideas of our modeling approach based on an example (section 3.1). Subsequently, in section 3.2, we illustrate how our approach contributes to an efficient stakeholder communication and how existing business process models, e.g. in Petri net notation, can be mapped to our FSM-based notation. Finally, section 3.3 contains a detailed description of the three generic activity types which points out how the annotation of physical configuration aspects enables the transition from the model to its execution within a portal.

3.1. A modeling example

The following (slightly simplified) example is taken from a large-scale, university-wide EAI project called “Karlsruhe’s Integrated InformationManagement (KIM)” [10]. In this project, a multitude of Web services providing homogeneous access to heterogeneous legacy systems was developed. Amongst these, there is a Web service providing comprehensive course information based on a course management legacy system and another Web service providing access to course assignment data, i.e. which student has registered for which courses. Each of these services is to be integrated in a “Students Portal” in a variety of integration scenarios, e.g. university calendar, course search, room occupancy schedule, personal timetable, course registration etc. While our approach was applied for all Web service integration scenarios (cf. section 5), this subsection concentrates on the course registration feature which supports students in the process of searching and registering for courses at the beginning of a semester.

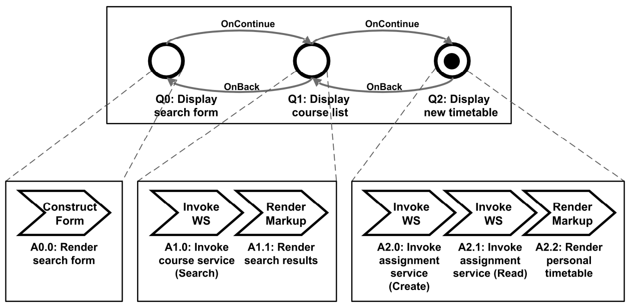

OnContinue
OnContinue
OnBack
OnBack
Q0: Display search form
Q1: Display course list
Q2: Display new timetable
Construct Form
A0.0: Render search form
Invoke WS
Render Markup
A1.0: Invoke course service (Search)
A1.1: Render search results
Invoke WS
Invoke WS
Render Markup
A2.0: Invoke assignment service (Create)
A2.1: Invoke assignment service (Read)
A2.2: Render personal timetable

Figure 1: FSM-based model of the “course registration” integration scenario

Figure 1 shows the two-layered model of the “course registration” UI workflow. It can be formally defined in terms of a FSM as $W = (Q, \Sigma, \delta, q_0, F, A)$ with

$Q = \{Q_0, Q_1, Q_2\}$: Set of user views

$\Sigma = \{\text{OnContinue}, \text{OnBack}\} = \Sigma_{\text{default}}$: Set of events which can be triggered by a user. Event sets that are likely to recur again in the future are defined as normalized Σ clusters, thus easing reuse in the implementation phase.

δ:	State transition function, i.e. possible navigation paths between the user views δ: $Q \times \Sigma \rightarrow Q$
q0 =	$\{Q_0\}$: Initial user view
F =	$\{Q_2\}$: Set of final user views
A =	$\{a_{q,i} \mid q \in Q, i \in N_0\} = \{a_{0,0}, a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, a_{2,2}\}$: Set of entry actions to be performed when entering state q

The set of user views consists of three states: In the first state, Q_0, a search form for specifying the parameters for the course search is displayed to the user. Therefore, a 'ConstructForm' activity ($a_{0,0}$) for generating the search form is executed when entering Q_0. Having filled out and submitted the form, whereby the event Σ_0 'OnContinue' is triggered, the user arrives in Q_1, the search results list. When entering Q_1, an 'InvokeWS' activity ($a_{1,0}$) is being executed which runs a search against the course information Web service based on the search parameters defined in Q_0. Afterwards, a 'RenderMarkup' activity ($a_{1,1}$) renders the Web service response in form of a search results list. When the user selects a course she wants to register for from the result list, the event Σ_0 'OnContinue' is triggered and the transition to Q_2 takes place. Alternatively, using a corresponding button for activating the event Σ_1 'OnBack', the user can navigate back to the search form (Q_0). In Q_2, the user has been registered for the selected course and her personal timetable including the new registration is being displayed. Therefore, three activities have to be executed when entering the state: First, an 'InvokeWS' activity ($a_{2,0}$) accomplishes the registration for the selected course by creating a new registration record for the given course and student via the assignment Web service. Subsequently, the current list of course registrations for the given student is retrieved from the assignment Web service, again using an 'InvokeWS' activity ($a_{2,1}$). Finally, a 'RenderPresentation' activity uses the received assignment data and renders the student's personal timetable.

Beyond advanced UI workflows like this, also simple scenarios consisting only of only one step, e.g. invoking a Web service and rendering the result, can be realized with our approach.

3.2. Improving stakeholder collaboration and correlation to business process models

Especially when collaborating with project participants with a non-technical background, our modeling approach has proved to be quite reasonable and efficient. In our experience, stakeholders could easily associate the modeling elements states and transitions with user views and the navigation between these. Also the modeling of entry actions turned out to be rather comprehensible due to the limitation on only three meaningful activity types as well as their associated descriptions in the models. Thus, with regard to avoiding misunderstandings and to assuring efficient communications between all project participants throughout the whole development cycle, our approach has proved its worth. Thereby, two of the most problem fields in software projects [16] – ambiguous requirements and lacking or inefficient communications between the developers and the business – could successfully be handled.

A further advantage of our two-layered, FSM-based modeling approach is that business process models can easily be transferred – manually or even automated – to the proposed notation. Thus, process models resulting from the requirements engineering or conceptual design phases can be directly incorporated in the implementation phase. Solely the second layer, i.e. the modeling of entry actions, has to be added manually, for example by a developer collaborating with appropriate stakeholders. In the KIM project, the majority of the business processes to be supported by adequate information systems was modeled with hierarchical Petri nets. Starting with a very abstract view of the business process on the top layer, the process model is being more and more refined in the subjacent layers. The penultimate layer contains the UI workflow models and the bottom layer comprises system-to-system interactions, e.g. used for designing Web service orchestrations. Figure 2 shows exemplarily how a model from the UI workflow layer in Petri Net notation can be transferred to our proposed FSM-based modeling notation. Likewise, transformations from other business process modeling languages like e.g. BPMN can be easily realized.

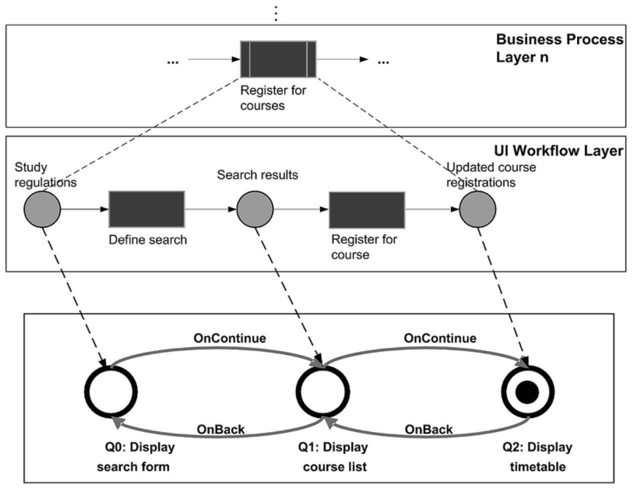


Figure 2: Deriving a FSM-based UI Workflow model from a hierarchical Petri net business process model

3.3. Generic activity types as fundamental integration building blocks

Based on the description of our approach's core ideas in the preceding sections, a detailed specification of the three generic activity types used for modeling a state's entry actions follows. As fundamental logical building blocks, they represent the functional units actually needed for integrating Web services in a portal: constructing forms, communicating with Web services and generating markup for displaying data in a portal. The activities were designed highly generic and thus configurable in order to assure their universal application for modeling all current and future integration scenarios. Each activity type provides specific configuration aspects for specifying parameters required to execute an UI workflow model by a dedicated portal component within a portal (physical design). Thus, these physical configuration aspects establish the transition from conceptual (business process models) and logical (FSM-based UI workflow model) to physical design. The configuration procedure should be supported by a dedicated editor, easing the configuration and assuring the preservation of configuration aspect interdependencies (cf. section 4).

3.3.1. Dialog construction – the 'Construct Form' Activity

Description: Dialogs are the central medium for user interaction in the World Wide Web. The logical building block 'ConstructForm' represents the generation of a Web form according to an XML-based specification. In contrast to manually developing forms, this approach leads to an improved development efficiency and form quality and enables the strict enforcement of quality guidelines. Especially in the field of Web accessibility, this gains more and more importance as many countries have passed ordinances requiring public institutions to assure their Web sites' accessibility, e.g. [8, 9]. Beyond that, the XML-based form specification is decoupled from the actual implementation and thus leads to a considerably decreased complexity. Our XML specification format is based on the W3C XForms standard. Hence, form specifications can be derived automatically from data schemas and subsequently be configured regarding layout and dynamic aspects.

Physical configuration aspects:

- *XML specification:* The specification of the form to be rendered based on the XForms standard. By means of a special tag, values from the result document of a previous 'InvokeWS' activities can be referenced.
- *URL to Web service interface description:* In Web service-based integration scenarios, forms are usually submitted to Web services. In these cases, the form is based on the data schema of a Web service operation. Using this and the 'data type selector' configuration aspects, a data schema from a Web service interface description (usually a WSDL document) can be referenced. Based on this data schema, the 'ConstructForm' activity generates a first XML specification of an appropriate form and stores it in the 'XML specification' configuration aspect. There, it can be further refined.
- *Data type selector:* An XPath expression for selecting a data schema from the Web service interface description referenced in the previous configuration aspect.
- *Data schema:* Instead of referencing a data schema from a Web service interface description using the two preceding configuration aspects, a data schema can be directly stored in this configuration aspect.

3.3.2. Web service communication – the 'InvokeWS' activity

Description: This logical building block represents invoking a Web service, receiving the return value, and storing it in an XML representation. If necessary, the communication can be realized securely based on WS-Security using encryption and / or digital signatures. The configuration of these security parameters can be achieved via the WS-Policy or WS-Security Policy standards [2] respectively.

Physical configuration aspects:

- *Web service URL:* URL of the Web service endpoint to be called.
- *Interface description URL*: URL where the Web service's interface description, usually a WSDL document, can be found.
- *Operation*: Name of the Web service's operation to be invoked.
- *Input parameters*: Values or references to values from previous activities to be passed as the operation's input parameters. For example, the user's input in a form field could be referenced as an input parameter. The input parameters are specified in form of an XML document using a special tag for referencing values from former activities.
- *Security policy source:* Used for specifying whether the WS-Security Policy based information about how a secured Web service call shall be realized, should be extracted from the WSDL document or from the subsequent configuration aspect.
- *Security policy*: Used for specifying a security policy for the Web service communication based on the WS-Security Policy standard.

3.3.3. Data presentation – the 'RenderMarkup' Activity

Description: The presentation of data, e.g. returned from a Web service invocation, is being represented by this logical building block. Similar to the 'ConstructForm' building block, the inherent preservation of quality guidelines plays an important role here, too. With regard to Web pages composed of autonomous components, enforcing and verifying quality guidelines at development time is rather sophisticated. Regarding for example accessibility aspects, this is due to the fact that the composition of components which are themselves accessible is not necessarily accessible [11].

Physical configuration aspects:

- *Data reference*: Reference to the 'InvokeWS' activity whose result data shall be displayed.
- *XSL template:* Reference to an eXtensible Stylesheet Language (XSL) document specifying the data transformation to the desired output format (e.g. XHTML, PDF etc.). If this configuration aspect remains empty, the 'RenderMarkup' activity shall generate an automatic presentation of the XML data. This can be achieved by presenting name-value pairs whose layout is derived from the XML document's structure.

4. Technical framework for executing UI workflows within existing portal systems

Figure 3 gives an overview of our technical framework's architecture consisting of four layers: The bottom layer contains the Web services to be integrated in the portal. Above, the 'UI Workflow' layer comprises FSM-based workflow instances as described in the previous section. The 'Data Exchange Service (DES)' layer holds mediating components decoupling workflows from the clients executing them. Therefore, a DES component offers a well defined interface to both parties based on the set of possible user events Σ. For the different Σ clusters, e.g. $\Sigma_{default}$ (cf. section 3.1), appropriate DES components already exist. In case a different Σ set is required, a custom DES component can easily be developed. Finally, the top layer contains instances of a generic portal component which is able to instantiate all kinds of workflows and to send and receive events to or from them via the DES layer.

Our current implementation is based on the Microsoft Windows Workflow Foundation (WF) as workflow engine. The FSM-based workflows as well as the entry action sequences can be modeled very comfortably using a graphical editor within Visual Studio 2005 (cf. section 5). The activity types described in section 3.3 were implemented as highly configurable software components, so-called 'Custom Activities'. When modeling an UI workflow, they can be easily integrated and configured via drag & drop and a dedicated property editor.

Regarding the portal component layer, we developed a component for the Microsoft Office SharePoint Server 2007. The so-called 'Web Part' is configurable in terms of the workflow library to be executed and the DES component to be used for communicating with the workflow.

Our implementation can be adapted very easily to be used in other portal systems. If the portal system's underlying platform is able to run the .NET Framework, only a portal system-specific portal component being able to communicate with the workflows via the DES layer has to be developed. All other layers of our implementation can remain untouched. For portal systems running on platforms which are not compatible to the .NET Framework, the Windows Workflow Foundation supports encapsulating workflows as Web services which can then be used from any platform. Thus, our Web service integration framework can easily be incorporated in all kinds of portal systems as the portal system-specific development effort is restricted to one specific component located in the top layer of the presented architecture. This portal component is rather simple as its only functionality lies in receiving markup from the workflow and sending back events triggered by a user – both via the DES layer.

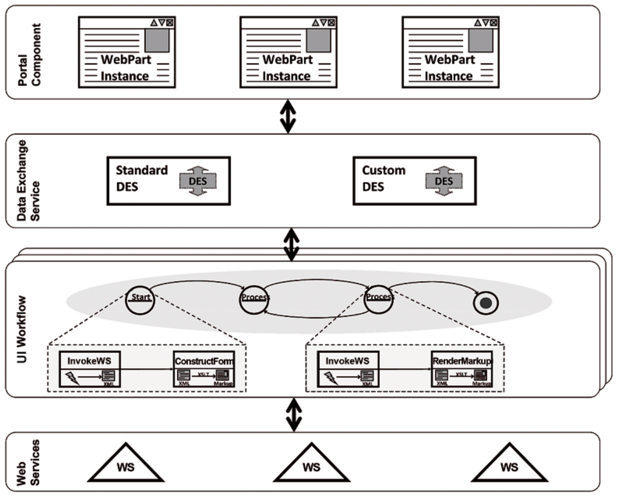


Figure 3: Overview of the architecture of our technical integration framework

5. UI workflows applied – developing a Web service-based portal

In the KIM project, we developed a completely service-based portal for the students of the University of Karlsruhe. Starting from October 2007, the portal shall serve as a uniform access point to all study-relevant information and business processes and provide novel features realized by integrating diverse legacy systems and processes.

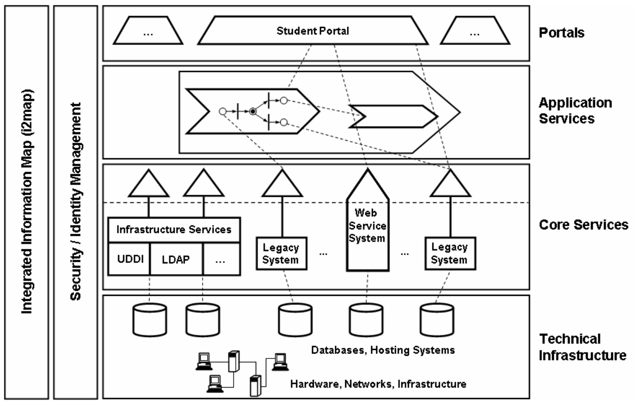


Figure 4: Overview of the Service-oriented Architecture from the KIM project

The KIM project is founded on a Service-oriented Architecture (SOA) as depicted in Figure 4. The Core Services layer mainly comprises highly reusable Web service wrappers where each service provides access to a semantically cohesive set of business objects stored in a legacy system or a database. For example, there are Core Services for courses, persons, rooms etc. An Application Service is also a Web service which composes Core Services to drive business processes or realize value-added functions. The Portal layer comprises mainly Web portals, e.g. the students portal, providing a centralized user interface for accessing the heavily distributed Application and Core Services. Security aspects and a support system for maintenance and evolution of the system landscape are comprised by two orthogonal layers.

The students portal integrates a multitude of Core and Application Services in various integration scenarios. Figure 5 shows an extract from the portal sitemap whereas the Web services used for a feature's realization are annotated in brackets.

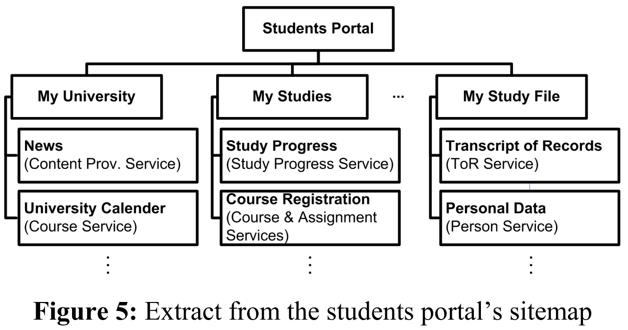
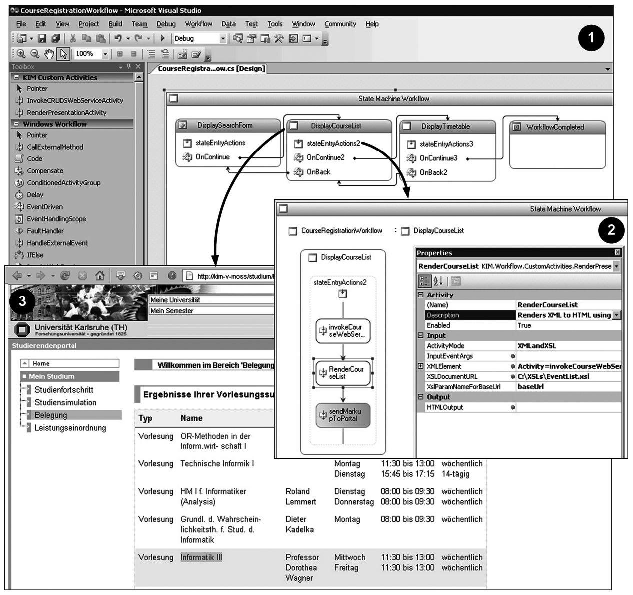


Figure 5: Extract from the students portal's sitemap

In the following, we outline the practical application of our modeling approach and technical framework for the realization of the 'Course Registration' feature as introduced in section 3.1. Figure 6-1 shows the graphical model of the FSM-based UI workflow within Visual Studio 2005 which corresponds to the UI workflow model in Figure 1. The modeling of entry actions as well as their physical configuration according to section 3.3 using a dedicated property editor is shown exemplarily for the state 'DisplayCourseList' (Q_1) in Figure 6-2. Corresponding to the model shown in Figure 1, an ‚InvokeWS' activity named ‚InvokeCourseWebService' and a 'RenderMarkup' activity named 'RenderCourse List' were placed consecutively and appropriately configured in the state's initialization phase. Having completed the workflow modeling, an instance of our generic Web Part (cf. section 4) is inserted on a portal page and configured with a reference to the compiled workflow library. After that, the realization of the integration scenario or the UI workflow respectively is completed.

When a user navigates to the portal page, the Web Part instantiates the workflow which in turn enters its first state 'DisplaySearchForm' (Q_0) and executes the entry activity modeled therein. The resulting output – in the course registration scenario a search form – is sent to the Web Part which displays it on the portal page. After the user has submitted the form which triggers the 'OnContinue' event, the workflow makes the transition to the second state 'DisplayCourseList' (Q_1), cf. Figure 6-3. By clicking on a course title, the event ‚OnContinue' is triggered and the workflow proceeds to the third state 'DisplayTimetable' (Q_2). Alternatively, a link at the bottom of the course list triggers the 'OnBack' event whereby the user returns to the 'DisplaySearchForm' state (Q_0).



Properties
RenderCourseList KIM.Workflow.CustomActivities.RenderPrese

Activity	
(Name)	RenderCourseList
Description	Renders XML to HTML using
Enabled	True
Input	
ActivityMode	XMLandXSL
InputEventArgs	
XMLElement	Activity=invokeCourseWebSer
XSLDocumentURL	C:\XSLs\EventList.xsl
XslParamNameForBaseUrl	baseUrl
Output	
HTMLOutput	

Typ	Name				
Vorlesung	OR-Methoden in der Inform.wirt- schaft I				
Vorlesung	Technische Informik I		Montag Dienstag	11:30 bis 13:00 15:45 bis 17:15	wöchentlich 14-tägig
Vorlesung	HM I f. Informatiker (Analysis)	Roland Lemmert	Dienstag Donnerstag	08:00 bis 09:30 08:00 bis 09:30	wöchentlich wöchentlich
Vorlesung	Grundl. d. Wahrschein-lichkeitsth. f. Stud. d. Informatik	Dieter Kadelka	Montag	08:00 bis 09:30	wöchentlich
Vorlesung	Informatik III	Professor Dorothea Wagner	Mittwoch Freitag	11:30 bis 13:00 11:30 bis 13:00	wöchentlich wöchentlich

Figure 6: UI workflow modeling in Visual Studio 2005 (1+2) and its execution within the students portal (3)

6. Summary & future work

The user interaction with the emerging multitude of Web services in medium to large-scale SOA-based systems is usually realized via portals. These portals provide appropriate interaction and presentation structures and realize the service communication. The integration scenarios, i.e. the Web service-based features within a portal, mostly consist of complex sequences of user interaction and service communication. With today's portal systems, such scenarios cannot be realized in an efficient and reuse-oriented way; however, with respect to the large number of services to be integrated, an efficient methodology is essential. We present an approach for modeling these scenarios in form of 'UI workflows' as well as a technical framework for their execution which can easily be integrated in existing portal systems. Our model for describing the user views, the possible navigation trails between them and the actions to be performed when entering a state is based on Finite State Machines (FSM) and a core set of generic activity types. We propose three activity types targeting the areas of dialog, service communication and data presentation. The models can either be derived from existing business process models or designed from scratch with strong stakeholder involvement.

Our approach was successfully applied for the development of a Web service-based portal in a large-scale EAI project. We identified the high efficiency and flexibility when realizing new integration scenarios or adapting existing ones to be the approach's main advantages. Beyond that, due to the building block-based modeling approach as well as the inherent 'configuration instead of programming' paradigm, quality guidelines, e.g. concerning Web accessibility, could be effectively preserved.

At the moment, we are working on the enhancement of the existing activity building blocks. As for the service communication activity, emphasis lies on incorporating federated security concepts while the dialog construction activity will be enhanced by further dynamic interaction aspects. Furthermore, we are intensively examining how the core ideas of our approach can be transferred to the realization of long-running, distributed workflows.

7. References

[1] Arsanjani, A., Service-oriented Modeling and Architecture - 2004), IBM: http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/ (05.07.2006).

[2] Bajaj, S., et al., Web Services Policy 1.2 - Framework - W3C Member Submission - 2006: http://www.w3.org/Submission/WS-Policy/ (4.12.2006).

[3] Bea Systems, BEA WebLogic Homepage - 2006: http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/.

[4] Ceri, S., Fraternali, P., and Bongio, A. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. in 9th International World Wide Web Conference (WWW). 2000. Amsterdam, Nethderlands. p. 137-157.

[5] 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.

[6] Gaedke, M. and Turowski, K., Specification of Components Based on the WebComposition Component Model, in Data Warehousing and Web Engineering, S. Becker, Editor. 2002, IRM Press: Hershey, PA, USA. p. 275-284.

[7] Hollingsworth, D., The Workflow Reference Model (1.1) - 1995, The Workflow Management Coalition: http://www.wfmc.org/standards/docs/tc003v11.pdf (22.01.2007).

[8] IT Accessibility & Workforce Division (ITAW) - Office of Governmentwide Policy - U.S. General Services Administration, U.S. Section 508 Guidelines - 1998), The Rehabilitation Act: http://www.section508.gov

[9] Japanese Standards Association, JIS X8341-3:2004 Guidelines. 2004, Japan Industrial Standards Committee

[10] Juling, W., KIM Project Homepage - 2005, University of Karlsruhe: http://www.kim.uni-karlsruhe.de/ (29.01.2007).

[11] Luque Centeno, V., et al.: Web Composition with WCAG in Mind. in Fourteenth International World Wide Web Conference (WWW), International Cross-Disciplinary Workshop on Web Accessibility (W4A). 2005. Chiba, Japan: ACM. p. 38 - 45.

[12] Microsoft, Microsoft Office SharePoint Server 2007 Homepage - 2007: http://office.microsoft.com/en-us/sharepointserver/default.aspx (01.02.2007).

[13] Phifer, G., A Portal May Be Your First Step to Leverage SOA, Gartner ID G00130149. 2005: Stamford, CT.

[14] Reddot Solutions, RedDot LiveServer Homepage - 2006: http://www.reddot.com/products_personalization_and_integration.htm

[15] Schwabe, D., Rossi, G., and Barbosa, S. Systematic Hypermedia Design with OOHDM. in ACM International Conference on Hypertext' 96. 1996. Washington, USA.

[16] The Standish Group International, CHAOS Research - Research Reports (1994-2006): http://www.standishgroup.com.

[17] Thomas Schaeck - IBM Software Group, WebSphere Portal Server and Web Services Whitepaper - 2006: www.ibm.com/software/solutions/webservices/pdf/WPS.pdf.

[18] World Wide Web Consortium, Web Accessibility Initiative (WAI) Homepage - 2006: http://www.w3.org/WAI/.