Study Progress Visualized in a Web Portal

Philip Hoyer[1,2], Stefan Link[1,2], Michael Gebhart[1,2], Ingo Pansa[2], Sebastian Abeck[2]

[1]Karlsruher Integriertes InformationsManagement (KIM)
Steinbuch Centre for Computing (SCC)
[2]Research Group Cooperation & Management,
Karlsruhe Institute of Technology, Germany
{ hoyer | link | gebhart | pansa | abeck } @ kit.edu

Abstract. To improve the quality of research and teaching, the Karlsruhe Institute of Technology (KIT) set up a Web-based, service-oriented portal, supporting students and lecturers alike in their daily business. To narrow the gap between the user's requirements and the supporting IT systems, existing distributed legacy applications have to be leveraged to provide common interfaces that can be integrated as functional components to the portal. Since new requirements continue to emerge, a systematic development approach for deriving adequate interfaces and data schemas from requirements is taken. Therefore, we propose a model-driven approach for deriving Web service interfaces and data schemas based on requirements and exemplify our approach by extending the portal with a new feature called "study progress", which requires the integration of additional distributed legacy applications.

Keywords: Distributed Legacy Applications, Social Community, Web Services, Model-Driven Software Development

1 Introduction

Due to increasing competition between universities, at the Karlsruhe Institute of Technology (KIT) one of the key targets is the improvement of the quality of research and teaching [2]. Therefore, a Web-based, service-oriented portal, which is referred to in this paper as KIT-portal, has been established to support students and teachers in their daily business, ranging from registering for an exam to paying tuition fees (c.f. [17]). Hence, functionality provided by several existing applications has been integrated to the KIT-portal and can now be accessed in an efficient way.

As students represent a major part of the social community of a university, there is a constant demand of supporting additional and new requirements via the KIT-portal. For example, the students are very interested in being able to see their current study progress at a glance. Hence, they wish to keep track of passed, failed or outstanding examinations, they want to review their overall performance or they want to be informed about the next possible steps in their studies. To support these business requirements in the KIT-portal, the functionality of existing course and exam management applications has to be reused and improved to provide the required overview.

Besides the challenge of integrating existing applications, the space for visualizing study progress is limited due to the properties of the KIT-portal. Thus, a preliminary analysis of possible visualization techniques has been performed [2], suggesting the concept of tree maps [1] as an adequate visualization form meeting the given requirements.

In this scenario, we present a model-driven approach focusing on the requirements of students at our university, as, e.g., the visualization of the progress of a student. Therefore, these requirements are aligned with the provided functionality of existing distributed applications by using Web service interfaces. The required composition of different Web service interfaces is formalized using UML activity diagrams. They are transformed [9] into a service model [3] and in a last step transformed into Web service interfaces using the Web Services Description Language (WSDL) [5] and data schemas using the XML schema definition (XSD) [6].

The paper is structured as followed: Section 2 illustrates the model-driven approach for the study progress. Section 3 presents the most relevant related work in the context of modeling workflows with the UML and the transformations into interfaces for Web services. Section 4 concludes the paper and makes some suggestions for future research work.

2 Service-oriented and Model-driven Development Approach

In this chapter, our model-driven software development process is discussed, using the example of implementing a visualization of a student's progress in his/her studies for the KIT-portal. The development process starts with the definition of the requirements by means of a GUI sketch. The next step is to model the data types and the workflow according to the defined requirements and the available legacy applications. Afterwards, model-driven transformation techniques are applied, generating formal interface descriptions by transforming the workflow modeled, by means of an UML activity diagram into a service model. Finally, a second transformation step is used to generate Web service interfaces and corresponding XML-based data types.

2.1 Defining the Requirements

The KIT offers its students the KIT-portal [17], where each student can access his/her personal data and perform actions (e.g., to register for an examination) in a simple and intuitive way. The KIT-portal integrates several existing applications in a service-oriented manner using Web technologies and Web standards. At the KIT, several applications are available, each storing and providing individual data for students. However, none of the applications provides interoperable interfaces, hence preventing an easy and straightforward service-oriented integration. An important step towards service-orientation is the development of standardized and technology-neutral interfaces for accessing and manipulating the data provided by existing legacy applications [11]. These interfaces and the corresponding adapter logic have to be developed to allow the integration of existing applications.

One feature of the KIT-portal to be developed is meant to facilitate a student's overview of his/her passed, failed or outstanding examinations in a graphical and easily understandable manner. Hence, several GUI sketches and prototypes were created prior to starting the development process, to get the look-and-feel for an adequate visualization form of the study progress. A modified version of a tree map provided the most promising results [1]. In the modified tree map, all the learning modules of a study course are visualized by rectangles using an equal width, but different height, depending on the amount of credit points (c.f. European Credit Transfer System, ECTS [18]) of the module. The same applies for the examinations allocated to a module. In addition, each examination is color-coded depending on the current state or result with regard to the student. For instance, a failed exam is colored red, whereas a passed exam is colored in different green tones, according to the examination result. Examinations that cannot be taken due to unfulfilled preconditions are colored grey, etc. This way, students can check their progress at a glance.

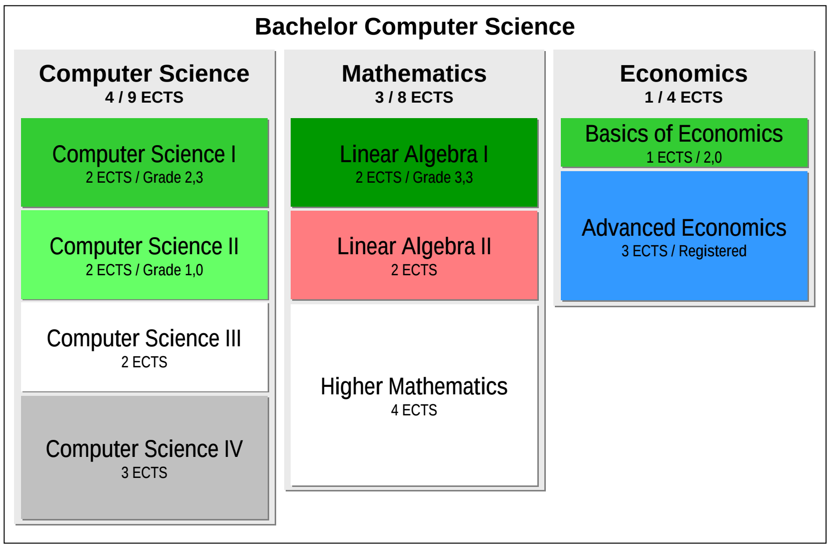


Fig. 1: An early GUI sketch for the study progress tree map

Figure 1 shows an early GUI sketch of the study progress tree map for the study course "Bachelor of Computer Science" consisting of three modules "Computer Science", "Mathematics" and "Economics". This sketch provides the starting point for realizing the technical solution that is aligned with the derivable requirements.

2.2 Analyzing and Designing the Workflow "Study Progress"

Based on the defined requirements we extracted from the GUI sketch, the needed data objects for the study progress tree map, such as examination results or personal information about the student, are specified. In almost all cases, the data objects can easily be derived from the native interface description of the legacy applications, or, if such an interface is not present, from the database schema used by the underlying application. Hence, the desired data objects are modeled as UML classes including properties and associations.

Having analyzed and modeled the required data objects for implementing the tree map, the next step is to design the workflow in a bottom-up way. During the execution of the workflow, several applications providing the required data are invoked and finally an XML representation of the tree map is available. The workflow for visualizing the study progress is represented by a UML activity “StudyProgress” (c.f. figure 2). The input to the workflow is the login name from the KIT-portal (student’s university e-mail address) only. The output is the tree map as a structured data type.

To specify the starting input and the final output of data, the activity has two ActivityParameterNodes (“loginEmail”, “treeMap”) attached to it. The activity contains several ActivityPartitions (“Study”, “Examination”), each representing a legacy application to be invoked during the execution of the workflow. To invoke an application, CallOperationActions are used and modeled (e.g., “GetStudentBaseData”, “GetExaminationResults”, “GetCourseCatalog”). As a minor restriction, it is not possible to invoke more than one application within one invocation. Therefore, each CallOperationAction must be contained in exactly one ActivityPartition. However, since one application can be invoked to retrieve different data sets, a UML ActivityPartition can contain several different CallOperationActions.

The activity diagram is refined by specifying the type of data sent to or retrieved from the invoked applications. The type of data sent to an application is modeled by adding InputPins and/or ValuePins to the CallOperationAction. In contrast, OutputPins represent the data returned from an application. According to the UML Superstructure [4], the UML Pin meta class is derived from the TypedElement and the MultiplicityElement meta class by Generalization. The former enables the user to type a Pin with a PrimitiveType (such as String, Integer, etc.) or one of the data objects modeled earlier as a UML class. The latter allows the collection of complex data structures in one invocation. The same applies for the two ActivityParameterNodes.

To represent the data flow between the invocations, we add ObjectFlows between InputPins and OutputPins. The ObjectFlows also specify in which order the invocations must be executed. Additionally, if a typed InputPin does not have a matching incoming ObjectFlow, the required data has to be collected in some other way. In such a case, we need to model additional Actions, which return the required data and provide an OutputPin for that. An example is the data of the current term that is needed as an input parameter to get the student’s current study program and examination list. Thus, we add the Action “GetCurrentTerm”, which has no InputPins but one OutputPin containing the current term as a String. The same applies for the student’s matriculation number (“GetMatricNumber”), since the portal system only knows the student’s university e-mail address, which has to be entered during the KIT-portal login. Of course, the appropriate application which holds the data (in our case, the accounting application) must be known in advance. Thus the application has to be added as an ActivityPartition, if not present yet (e.g., “Account”).

With the use of an activity diagram, we have formalized how the data is processed, which applications are invoked and in which order the invocations occur. Figure 2 shows a part of the final activity diagram labeled as “Analysis & Design Model”. Due to space restrictions, Figure 2 only shows the upper part of the activity diagram.

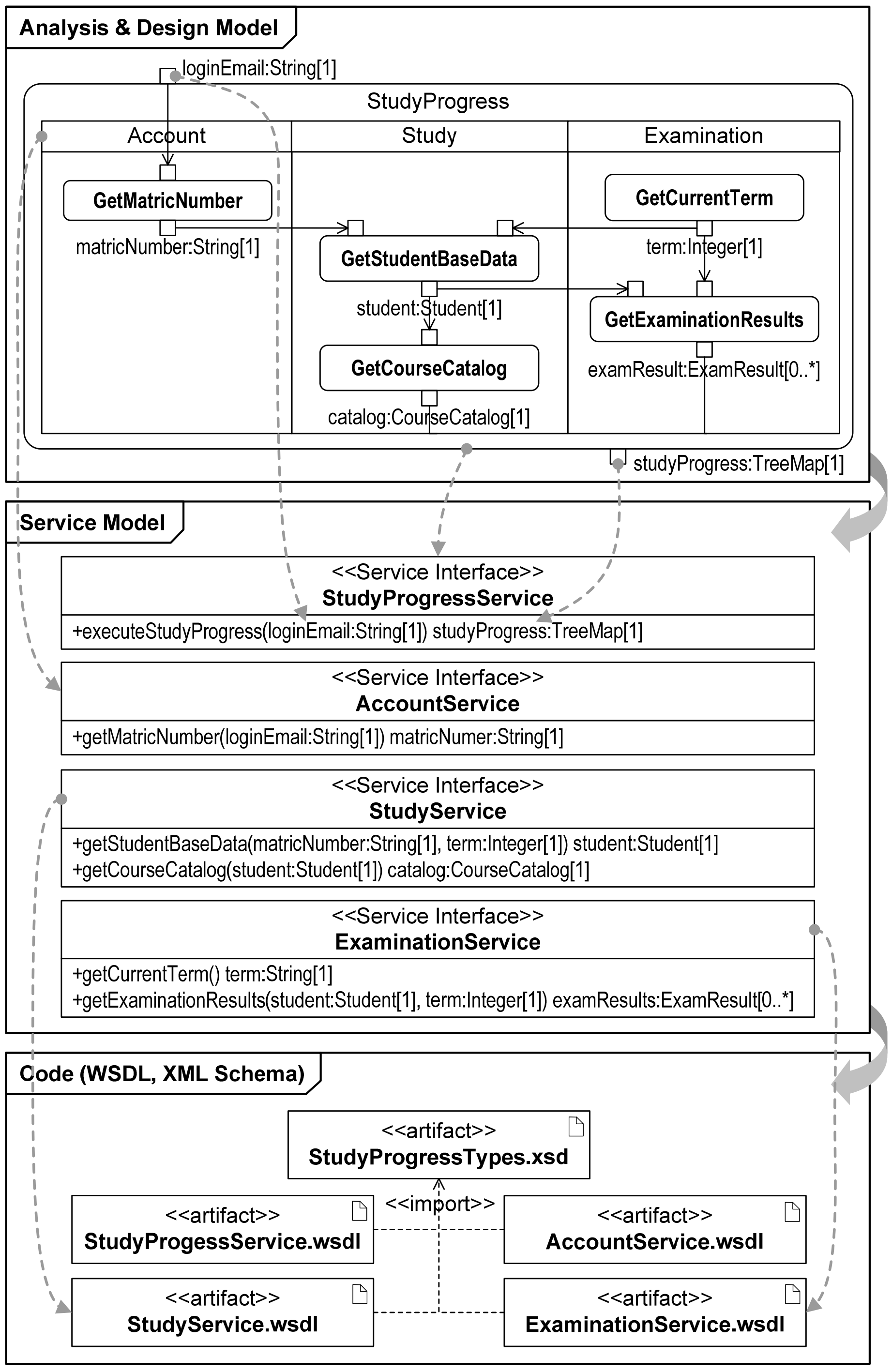


Fig. 2: Overview of our approach

2.3 Transformation to a Service Model

To generate standardized Web-based interface descriptions and data types, the next step is to transform the model described in the previous chapter to a service model [3], which, among other details, specifies the interfaces for each legacy application and the study progress workflow itself.

The transformation rules are formalized in the transformation language “Queries, Views, Transformation” (QVT) [7]. Since the transformation rules are described by meta model elements, the transformation itself is independent from the concrete model, and thus can be reused for other development projects.

The transformation uses the created model as the source and generates a target model according to a set of transformation rules. Since each ActivityPartition represents an application, we transform the ActivityPartitions into a UML Interface. Each CallOperationAction contained in the ActivityPartition results in an Operation of the associated Interface. Finally, InputPins and OutputPins of the Action are converted into Parameters of the Operation with the appropriate direction flag (in or out). It is not required to transform the data types modeled as Classes. Still, the data types are needed in the target model. The Classes representing the data types either can be imported from the source model in the target model or copied one-to-one to the target model. The same applies for the UML Activity and the containing Actions. The operation Property of the CallOperationActions can now be associated with the generated Operations of the Interfaces. Finally, another interface “StudyProgressService” and a containing Operation “executeStudyProgess” are generated, representing the Activity “StudyProgress”. The Parameters for this Operation are generated according to the ActivityParameterPins. In total, $n + 1$ interfaces are generated, whereby n correlates to the number of invoked applications.

The middle part of Figure 2 shows the resulting target model, containing Interfaces for each ActivityPartition and the Activity itself. Due to space restrictions, we omitted most stereotypes in Figure 2 as specified in [3]. The grey dashed lines show some exemplary transformations from the activity diagram model elements to model elements of the Service Model.

2.4 Transformation into Web Service Interface Descriptions

As the final modeling step, we transform the technology-neutral UML Interfaces from the service model into WSDL [5] and the corresponding Classes to XML Schema [6]. The transformation rules are mainly straightforward. Each Service Interface is transformed into an abstract part of a WSDL file with exactly one port type. The port type contains the same number of operations as the UML Interface specified. The generation of the messages for the input and output of the Web service depends on the WSDL style. Since it is most common and recommended by WS-I [16], we use the style “document/literal-wrapped” [13]. For this style, each message acting as input or output for a Web Service contains exactly one part, even if multiple UML Parameters are specified as input or output. To distinguish between the Parameters, XML Schema is used to build an RPC-like XML structure, using the operation name as the top XML element and the names and types of the parameter as XML child elements.

The data types specified as UML Classes are transformed to one XML Schema file [6], containing all needed data types as complex types. The schema file is imported by every WSDL file generated to have a common set of XML data types for different Web services.

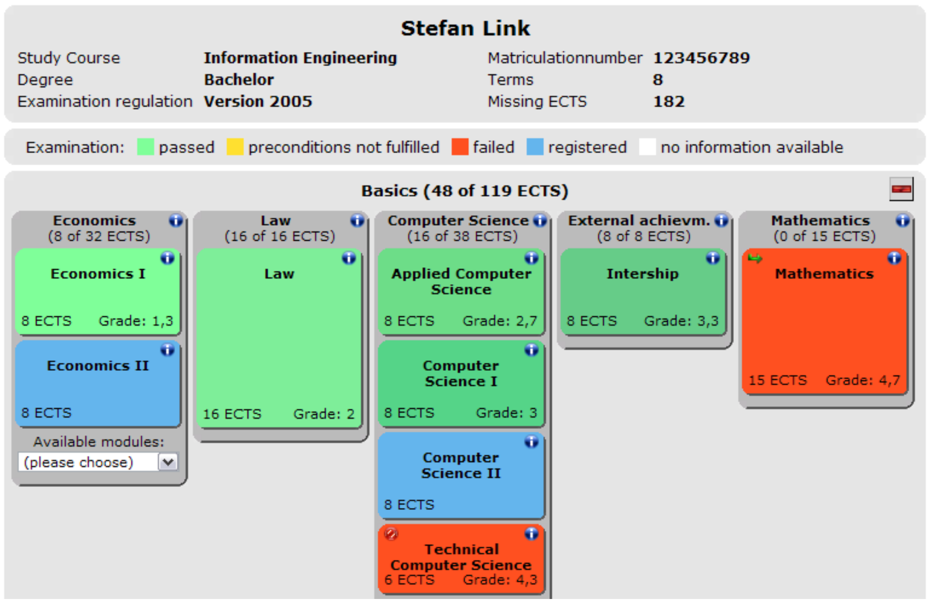
To also generate the concrete part of the WSDL file, the proposed service model can be extended by using UML Components and attached Ports, as in [3, 10]. A Port acts as WSDL bindings and refers to the generated Service Interfaces as provided interfaces or if needed by composite components (like the StudyProgress) as required interfaces.

2.5 Final Steps

To finalize the integration, the required Web services have to be implemented. The generated WSDL files can be used to create skeletons for the implementation. For this purpose, existing approaches are applied [9] that are already part of several development tools. For the implementation of the Web services to integrate existing applications, Java or .NET is used.

The entire study progress process is also provided as a Web service. It is implemented using the Business Process Execution Language (BPEL) [8], which can also be generated from the UML activity diagram. For the sake of simplicity, we omitted this part in the paper but will present it in a future work.

With the implementation of the Web services and the workflow, the entire Study Progress workflow can be executed. Figure 3 gives the result of the engineered solution, showing a late prototype of the study process running in the KIT-portal.



Stefan Link

Study Course	**Information Engineering**	Matriculationnumber	**123456789**
Degree	**Bachelor**	Terms	**8**
Examination regulation	**Version 2005**	Missing ECTS	**182**

Examination: 🟩 passed 🟨 preconditions not fulfilled 🟥 failed 🟦 registered ⬜ no information available

Basics (48 of 119 ECTS)

Economics (8 of 32 ECTS)
- **Economics I** — 8 ECTS Grade: 1,3
- **Economics II** — 8 ECTS
- Available modules: (please choose)

Law (16 of 16 ECTS)
- **Law** — 16 ECTS Grade: 2

Computer Science (16 of 38 ECTS)
- **Applied Computer Science** — 8 ECTS Grade: 2,7
- **Computer Science I** — 8 ECTS Grade: 3
- **Computer Science II** — 8 ECTS
- **Technical Computer Science** — 6 ECTS Grade: 4,3

External achievm. (8 of 8 ECTS)
- **Intership** — 8 ECTS Grade: 3,3

Mathematics (0 of 15 ECTS)
- **Mathematics** — 15 ECTS Grade: 4,7

Fig. 3: Visualization of the advanced prototype

3 Related Work

As our approach targets a wide area of different artifacts supporting a model-driven development approach (GUI sketches, service model, WSDL and Web services), there are several related studies.

Considering the overall development approach, starting with formal requirements and leading to a set of executable code, Meijler, Kruithof et al. illuminate the advantages of model-driven integration aligned with service-oriented principles [12]. An integrated approach combining both top-down (requirements to software components) and bottom-up (existing tool assets) approaches is proposed. Therefore, we decided not to follow strictly a top-down development approach that would hamper the integration of existing applications, but to follow a combined middle-out approach enabling the description of existing applications early in the transformation process.

The idea of visualizing hierarchically structured information in terms of tree maps initially was published by Johnson and Shneiderman [1]. Based on their concepts, Allerding, Buck et al. present an approach using tree map concepts and focusing the requirements of students managing their studies [2]. Adapting their idea of visualizing the study progress of a student, we used an early sketch of a tree map as input for a model-driven development approach. The execution of integration projects following a model-driven development approach based on Web service technologies is discussed in scientific and commercial communities alike.

Model-driven development of Web services has already been discussed in several previous works, for instance in [3, 15, 16]. Based on these approaches, we focused on capturing business requirements with models and mapping these models to existing distributed legacy applications. Considering the integration of legacy applications using Web services, a generic model for application integration is presented [11]. Since different legacy applications often use different formats and standards for describing their data schemas, a mapping of these different data schemas has to be realized additionally. The proposed approach in [11] focuses the integration of several different data schemas by implementing adapter components realized with Web services. Within the special requirements of our scenario, not only the integration of existing data schemas but also the integration of existing business logic is needed; thus our approach considers the aspect of integration from a system-oriented direction.

Finally, the presented intermediate model for service descriptions (c.f. chapter 2.3) is based on the work of Emig, Krutz et al. [3]. While the approach presented in [3] targets towards a holistic and technology-independent possibility for describing service interfaces of service-oriented components, we improved the proposed development approach by the integration aspect of existing software assets. Similar to [3], Johnson demonstrates the use of a technology-independent approach for describing service-oriented software components [10]. An UML2.0 Profile as an extension to existing modeling tools is proposed, although specific modeling elements are introduced regarding the very special needs of the appointed vendor-specific tool chain.

4 Conclusion and Outlook

In this paper, we outlined how a study progress as integration scenario of existing distributed legacy applications can be realized using a model-driven development approach. The visualized study progress allows students to see their status quo at a glance. To provide the required functionality in a service-oriented manner, legacy applications have been integrated via standardized Web service interfaces. First, the necessary composition of functionality is modeled using UML Activity Diagrams. Their application enables the usage of several existing UML modeling tools and allows a formalized and visual description of the application logic and thus enables domain experts to describe the integration process. To derive the Web service interfaces, automatic transformations are applied that help to avoid transformation errors due to human interpretations. As the approach started by gathering user requirements by means of a GUI sketch, we consider our solution user-aligned and a promising enhancement of existing integration approaches. Due to the successful realization of the study progress at the KIT, we plan to establish this development approach for future works as an integrated course catalog and a library that require the integration of various distributed legacy applications. With these features, we offer our students as a social community further innovative functionality.

References

1. Johnson B., Shneiderman B: Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures, IEEE Computer Society Press, http://hcil.cs.umd.edu/trs/91-06/91-06.html, October 1991.
2. Allerding F., Buck J., Freudenstein P., Klosek B., Höllrigl T., Juling W., Keuter B., Link S., Majer F., Maurer A., Nussbaumer M., Ried D., Schell F.: Integriertes Service-Portal zur Studienassistenz, Proceedings of the 38th GI Conference - Lecture Notes in Informatics, München, Germany, Munich, 2008.
3. Emig C., Krutz K., Link S., Momm C, Abeck S..: Model-Driven Development of SOA Services, Cooperation & Management, Universität Karlsruhe (TH), Internal Research Report, 2008.
4. Object Management Group (OMG): Unified Modeling Language (OMG UML), Superstructure Version 2.2. http://www.omg.org/cgi-bin/doc?formal/09-02-02
5. World Wide Web Consortium (W3C): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/wsdl20/
6. World Wide Web Consortium (W3C): XML Schema Definition Language (XSD) 1.1 Part 1: Structures. http://www.w3.org/TR/xmlschema11-1/
7. Object Management Group (OMG): Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.0. http://www.omg.org/spec/QVT/1.0
8. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
9. The Apache Software Foundation: Code Generator Wizard - eclipse Plug-in, http://ws.apache.org/axis2/tools/1_0/eclipse/wsdl2java-plugin.html
10. Johnston S.: UML 2.0 Profile for Software Services, IBM developerWorks http://www.ibm.com/developerworks/rational/library/05/419_soa/, April 2005.

11. Harikumar A., Lee R, Yang H., Kim H., Kang B.: A Model for Application Integration using Web Services, Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, July 2005.
12. Meijler T.D., Kruithof G., Beest N.: Top Down Versus Bottom Up in Service-Oriented Integration: An MDA-Based Solution for Minimizing Technology Coupling, Lecture Notes in Computer Science Volume 4294/2006.
13. Butek R.: Which style of WSDL should I Use, IBM developerWorks, 2003. http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/
14. Gronmo R., Skogan D., Solheim I., Oldevik J.: Model-driven Web Service Development. International Journal of Web Services Research, Volume 1, Number 4.
15. Marcos E., Castro V., Vela B.: Representing Web Services with UML: A Case Study. 1st International Conference on Service-Oriented Computing (ICSOC), Trento, Italy, December 2003.
16. Web Services Interoperability Organization: Basic Profile Version 1.2. http://www.ws-i.org/Profiles/BasicProfile-1_2(WGAD).html
17. Karlsruhe Institute of Technology (KIT): The KIT study portal, http://studium.kit.edu