WERKSTATT UNTERNEHMENSSOFTWARE KARLSRUHE (WUSKAR)

GRUNDLAGEN ZUR 2. WUSKAR-FALLSTUDIE „UNIVERSITY SOA“

Jochen Weisser

Christian Emig
Sebastian Abeck

Cooperation & Management (C&M)
Institut für Telematik
Universität Karlsruhe (TH)

Vorwort

Die im Folgenden vorgestellten Arbeiten wurden im Rahmen eines WUSKAR-Projekts durchgeführt. Die Aufgabe bestand darin, einen Webservice zu entwickeln, der einen elektronischen Notenauszug (Transcript of Records) für einen beliebigen Studierenden erstellt. Außerdem sollte dieser Webservice, der in den Projekt-Dokumenten den Arbeitstitel ToRService trägt, alle Eigenschaften aufweisen, um eine Integration in eine beliebige serviceorientierte Architektur (SOA) zu ermöglichen. Zum Nachweis dieser Eigenschaften sollte der ToRService abschließend in die University SOA, die die IT-Unterstützung der Geschäftprozesse einer Hochschule als Gegenstand hat, integriert werden.

Diese Arbeiten sind aber auch als Vorarbeiten für die nächste WUSKAR-Fallstudie zu sehen, die das Ziel hat, den bestehenden ToRService an das System SAP Campus Management, welches in diesem Szenario als Beispiel für ein Lagacy-System zu sehen ist, anzudocken. Daraus leitete sich während der Analyse eine weitere Anforderung für den Entwurf ab: der ToRService sollte unabhängig von bestehenden Komponenten entworfen werden, die ihm die nötigen Informationen liefern. Dies war einer der Gründe, weshalb für die Realisierung des ToRServices eine neue, viel versprechende Sprache, die Business Process Execution Language (BPEL), eingesetzt wurde.

Die Generierung des Notenauszugs sollte als Transcript of Records (ToR) erfolgen. Dies bedeutet, dass er den ECTS-Normen, die im Rahmen des Bologna-Prozesses definiert wurden, entsprechen sollte. Dies hatte zur Folge, dass das Projektteam zuerst ein Entwurf einer Datenrepräsentation dieses Transcript of Records entwickeln musste. Zu diesem Zweck wurde ein XML-Schema entworfen, was unter anderem auch den Vorteil mit sich bringt, eine einfache, automatische Validierung der später erzeugten Notenauszüge zu ermöglichen.

Als der Output des ToRService (der ToR an sich) hinreichend spezifiziert war, konnte der Input näher betrachtet werden. Es kristallisierten sich vier Quellen heraus, aus denen ein Transcript of Records seine Informationen bezieht. In der Praxis entspricht jede dieser Quellen einer separaten Datenbank bzw. eines separaten Systems. Diese einzelnen Datenbanken werden üblicherweise nicht zentral gehalten, sondern sind über den gesamten Campus verteilt. Es werden Informationen zur Person des Studierenden, Informationen über dessen erbrachten Leistungen, Informationen über die sendende bzw. empfangende Hochschule sowie Informationen über die zu den Prüfungsergebnissen gehörenden Studienmodulen benötigt. Deshalb wurden im Rahmen des Projekts vier Datenbanken von unterschiedlichen Herstellern gewählt, um die mögliche Heterogenität der einzelnen Quellen hervorzuheben.

Zur Vereinheitlichung des Zugriffs auf die einzelnen Datentöpfe wurden elementare Webservices entwickelt. Hierbei wurde explizit darauf geachtet, dass die Funktionalität jedes einzelnen Webservices nicht nur auf die Bedürfnisse des ToRServices zugeschnitten ist, sondern dass es auch Sinn macht, deren Dienste in anderen höherwertigen Webservices zu nutzen. Aber auch Aspekte der Sicherheit und der Managementfähigkeit wurden im Projektteam diskutiert und sind in die Konzeption der aller Services eingeflossen. Aus den WSDL-Beschreibungen, die das Projektteam zu jedem dieser Services entwickelt hatte, wurden die Service-Klassen unter Zuhilfenahme neuster Entwicklungswerkzeuge (Apache Axis, Eclipse, und Apache Ant) semi-automatisch generiert.

Die so entstandenen elementaren Services konnten unter Verwendung der Business Process Execution Language (BPEL) orchestriert bzw. verschaltet werden, sodass der eigentliche ToRService die benötigten Komponenten lediglich an ihrer SOAP-Schnittstelle anspricht. Somit ist die darunter liegende Geschäftslogik beliebig austauschbar und der ToRService plattform- und technologieunabhängig. Für die Implementierung des ToRService als BPEL-Prozess wurden verschieden Entwicklungswerkzeuge getestet und eingesetzt. Als Beispiele seien an dieser Stelle Popkins System Architect und Oracles BPEL Designer genannt.

Da man zur Ausführung eines BPEL-Programms eine BPEL-Engine benötigt, waren an dieser Stelle ebenfalls einige Tests proprietärer Software erforderlich. Die Aufgabe einer BPEL-Engine ist derer eines Interpreten ähnlich, wie er beispielsweise bei Programmiersprache Java eingesetzt wird. In diesem Projekt wurde nach absolvierten Tests der Oracle Process Manager verwendet, da dies die ausgereifteste Applikation war und da die Synergien des Einsatzes zweier Produkte desselben Herstellers sich als schwer entbehrlich erwiesen.

Abschließende Tests sowohl unter normalen als auch unter Volllast-Bedingungen zeigten, dass der ToRService ohne weitere Bedenken im Wirkbetrieb eingesetzt werden könnte.

Da nun die Vorarbeiten für die zweite WUSKAR-Fallstudie abgeschlossen sind, kann nun im Sommersemester 2005 die Portierung dieser ToRService in ein Unternehmenssoftware-Szenario mit SAP Campus Management erfolgen. Ziel der Fallstudie ist, wie oben bereits angedeutet, die Überführung einer Legacy-Unternehmenssoftware in eine serviceorientierte Architektur.

www.cm-tm.uka.de
C&M (Prof. Abeck)

USOA-ToR

University SOA – Transcript of Records Service

Jochen Weisser

Figure 1: SOA Part Documentation

Short Description

USOA-ToR is a part of the University SOA (USOA). It provides IT supported generation of a Transcript of Records. This document describes the development of the ToRService. Furthermore relevant ToR-related business processes are discussed as well as the involved core Web services are derived and designed.

Key Words

Transcript of Records (ToR), Service, Web service, University SOA (USAO) , ToRService, StudentDBService, ExaminationDBService, CourseUnitDBService, OrganizationDBService, XML, Transcript of Records (ToR), SAP Campus Management, UML, BPMN, BPEL, Java, XML, XSLT, SOAP, WSDL, MySQL, Apache Jakarta Tomcat, Apache Axis, Oracle BPEL Designer, Oracle Process Manager

Table of Contents

www.cm-tm.uka.de
C&M (Prof. Abeck)

(0) INTRODUCTION

Motivation; Embedding in a larger project e.g. as a milestone; Outlook; Tie in points / Entry points

(1) ANALYSIS

Embedding in the USOA; Transcript of Records Service (function and quality); roles and use cases, activity flow; GUI elements; Business objects; refinement of SOA Portal structure; BPMN diagrams for business processes

(2) DESIGN

ToR architecture as part of the USOA architecture; Mapping of the activity flow to the process layer (e.g. BPEL); Definition of web services; Mapping of web services to components; BPMN diagrams for the services

(3) IMPLEMENTATION

Portal implementation; process implementation (programming in the large); web service implementation (programming in the small); database implementation

(4) USE

Deployment and Installation; Configuration; Use; Embedding in the USOA Portal; GUIs supporting the use of the ToRServices

TABLES

Glossary, References

Figure 2: Overview

0 INTRODUCTION

0.1 Motivation

The Bologna Process introduced two major concepts: the European Credit Transfer System (ECTS) and the Transcript of Records (ToR). The superior goal was to establish the European Higher Education Area (EHEA). ECTS is a scheme that helps to evaluate the performed work of a student Europe-wide in a common way. It takes in account the workload of a course unit in general as well as the specific examination results of the student. A ToR is an extraction which includes all the performed work of a student in ECTS norm at a certain date.

These two results of the Bologna Process are basic concepts of this document. They provide several advantages for the students and the administration of the university:

- Better classification of the work that a student performed at a foreign university by another university or a third party
- Easier exchange of the performance data of a student during a university change
- A university can impute the work which a student performed at another university

The reasons why a university is interested in generating ToRs for their students are various but at the moment of writing this document IT support for the most parts is missing.

Another basic concept of this document is the Service-orientated Architecture (SOA). A Service-oriented Architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple message passing as well as it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. In practise Web services encapsulate these services. These Web services may be realized in any programming language. A key concept in this context is the data interchange and representation format XML and its derived standards like SOAP, UDDI and BPEL.

0.2 Approach

The ToRService that is described in this document generates an XML-based ToR that can then be converted into several other output formats. It consists basically of the following elements:

- An element which is integrated in the USOA Portal at GUI layer
- An element which route through the corresponding business process at choreography layer
- The central element which has the real functionality at composition
- Some elements at the layer of the core Web services, which encapsulate the access to the several databases
- And of course the databases, where the information is stored

The ToRService and the core components are realized as Web services. This scenario is the first milestone and it is displayed in Figure 3.

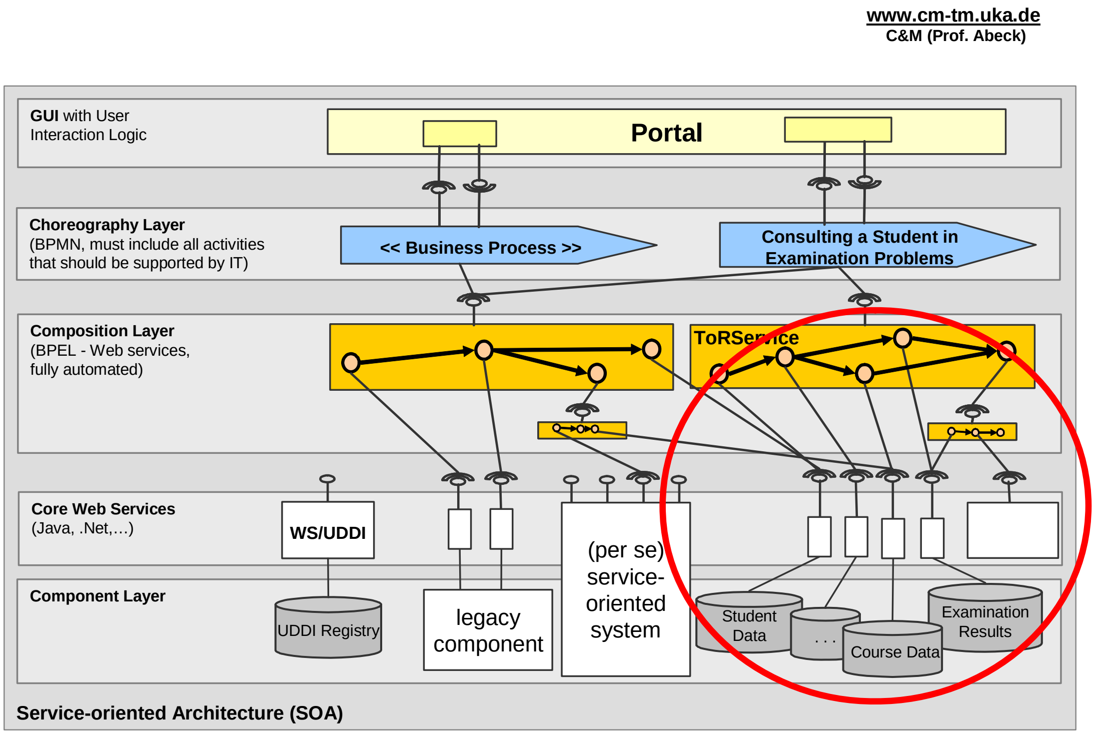

www.cm-tm.uka.de
C&M (Prof. Abeck)
GUI with User Interaction Logic
Portal
Choreography Layer (BPMN, must include all activities that should be supported by IT)
<< Business Process >>
Consulting a Student in Examination Problems
Composition Layer (BPEL - Web services, fully automated)
ToRService
Core Web Services (Java, .Net,…)
WS/UDDI
(per se) service-oriented system
Component Layer
UDDI Registry
legacy component
Student Data
. . .
Course Data
Examination Results
Service-oriented Architecture (SOA)

Figure 3: INTRODUCTION - First Cornerstone of Realization

The next step is to transport the developed concepts to the scenario of Figure 4. In this second milestone a connection to a legacy system, which in this context will be SAP Campus Management, should be developed. In ideal case the unchanged ToRService can be transferred to the new context by only changing the core Web services, so that they can connect to the SAP CM.

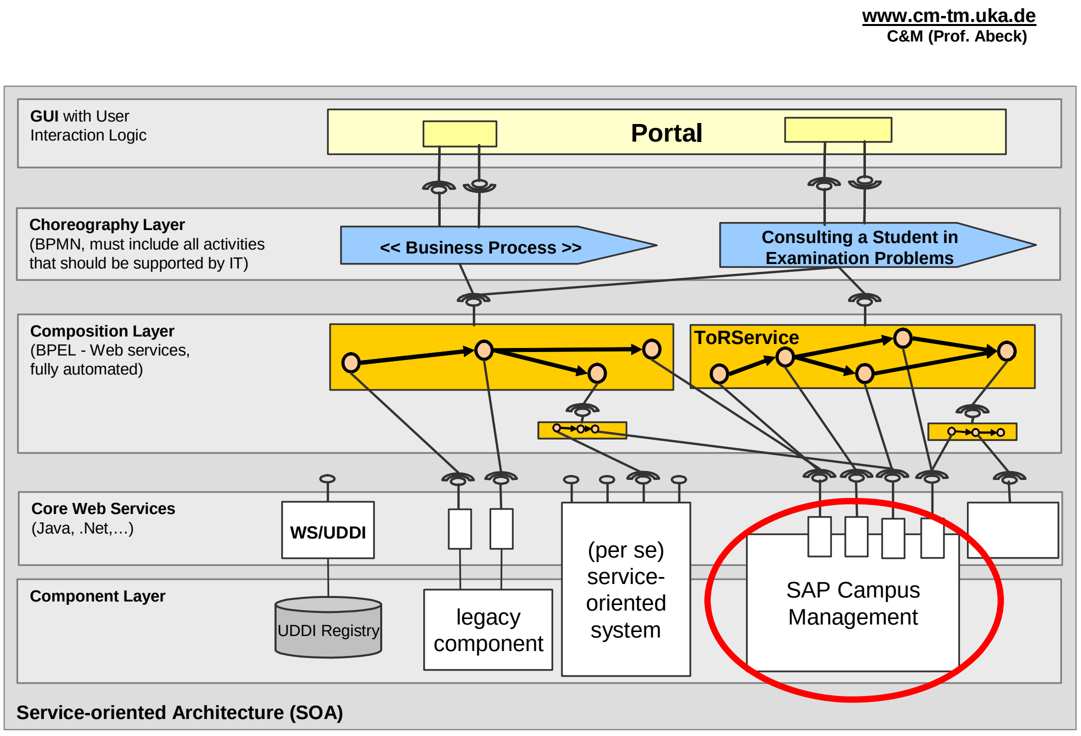

www.cm-tm.uka.de
C&M (Prof. Abeck)
GUI with User Interaction Logic
Portal
Choreography Layer (BPMN, must include all activities that should be supported by IT)
<< Business Process >>
Consulting a Student in Examination Problems
Composition Layer (BPEL - Web services, fully automated)
ToRService
Core Web Services (Java, .Net,…)
WS/UDDI
(per se) service-oriented system
Component Layer
UDDI Registry
legacy component
SAP Campus Management
Service-oriented Architecture (SOA)

Figure 4: Second Milestone of Realization

This document solely deals with the first milestone of realization.

0.3 About this Document

This document embodies a blueprint of a SOA Part Documentation (SOA Part Doc). It contains information that would not be included in a normal SOA Part Doc because the author wanted to point out all problems that aroused by designing and modeling the SOA Part Transcript of Records. It provides the basis for further developments in this SOA Part (quod vide Figure 4) as well as for other research projects in the SOA context. Furthermore the development of all in this SOA Part relevant core Web services is also documented here. To provide a better overview about the whole SOA Part everything is aggregated in only one document. So it is easier to get familiar with the realization process of a SOA Part by using the Transcript of Records Service as an example.

The following facts have to be considered when reading this document:

- The surrounding USOA which the ToRService will be embedded into did not exist when writing this document. This included also the SOA Portal which the ToRService has to interact with.
- Integral aspects of a SOA like authentication, UDDI registry and manageability of services are not discussed.
- As mentioned this is a first approach of a SOA Part Doc.

There exists an archive file that contains all code and files that were addressed in this document. It has the same name as this document but with the file extension zip. The term $ARCHIV embodies the absolute path to the folder where this archive file was unzipped.
The archive is structured as followed:

- The folder docs contains this document, the corresponding slides, the document 'Einführung in Java-Webservices' and other continuative material.
- The directories java respective bpel contain the core Web services and the ToRClient respective the ToRService.
- In the dbs folder the files to create the four databases are included.
- The config directory includes several configuration files e.g. for Tomcat that had to be edited to run the ToRService.
- And of course the two files RunRemoteClient.bat and RunToRClient.bat helps the user to easily run the corresponding client.

0.4 Outlook

To tie in with this work the complex business processes in which the ToRService maybe useful should be further explored. Because starting from an empty set of core Web services it makes no sense to talk about more complex business processes where more services like the ToRService are involved. Due to the fact that in the research activities an ideal technology for managing the upcoming needs of the choreography layer was not found, more time should be invested in this section. The only business process that is realized is the elementary process 'Generating a ToR', which for example a student may execute.

1 ANALYSIS PHASE

The analysis phase provides a user-driven specification of the service that should be developed. It helps the customer to understand importance of this new service within his existing SOA. The user can influence the look and feel of the new functionalities but he is not interested in how they were realized. To demonstrate the appearance of the ToRService to the customer in the following subchapter mainly UML is used. For modeling the business processes BPMN was taken. The business processes are only identified in this chapter. Of course business processes have to be modeled in the analysis phase because the created models have to be coordinated with the customer. But in this document it was useful to shift it to the following chapter. The reason why the business processes are not modeled in this chapter is explained at its end.

1.1 Embedding in the University SOA

Before exploring the SOA Part 'Transcript of Records' in detail, it is obvious that the surrounding University SOA has to be well-known. For this reason the first subchapters deal with USOA-related aspects. Figure 5 lists some typical business processes of a university.
Starting from a multiplicity of heterogeneous business processes at a university the first task when designing a SOA Part is to identify in which processes the considered services are important. But therefore the complete business area of the university has to be modeled. Because this would go beyond the scope of this document the amount of business processes is only implied in this figure.

www.cm-tm.uka.de
C&M (Prof. Abeck)

- Abstract business process with central goal: Educate students
- Ad-hoc listing of included BPs, not necessarily sequentially!
- **Goal:** Support these BPs using a university ressource planning system

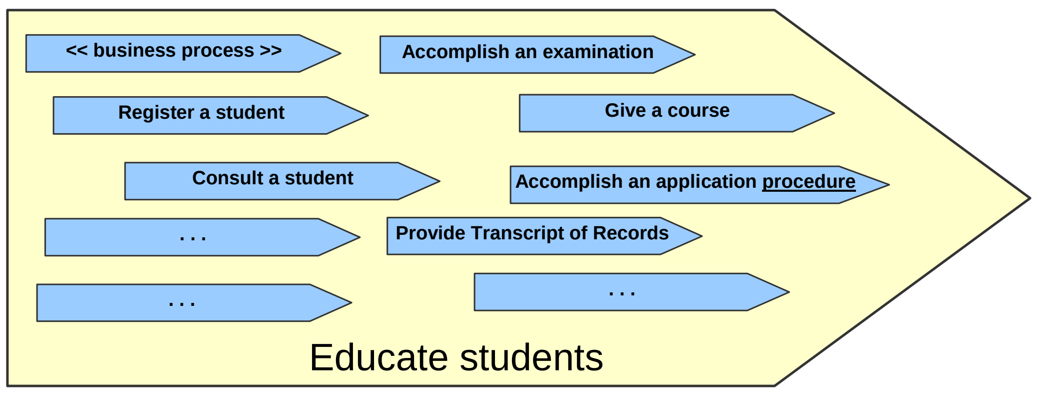


Figure 5: ANALYSIS PHASE - Business Processes at a University

Each of these business processes contains several tasks. Some tasks can be executed automatically for example as BPEL processes. Other tasks need at even one point information from the user and therefore a Portal GUI is needed. Figure 6 displays this issue for an arbitrary business process. The ToRService would be one of the automated services in some maybe completely different business processes. The way of executing the ToRService does not depend on the context out of which it is called.

- A business process can consist of a interconnection of tasks
- Some tasks need user interaction
- Some tasks can be fully automated

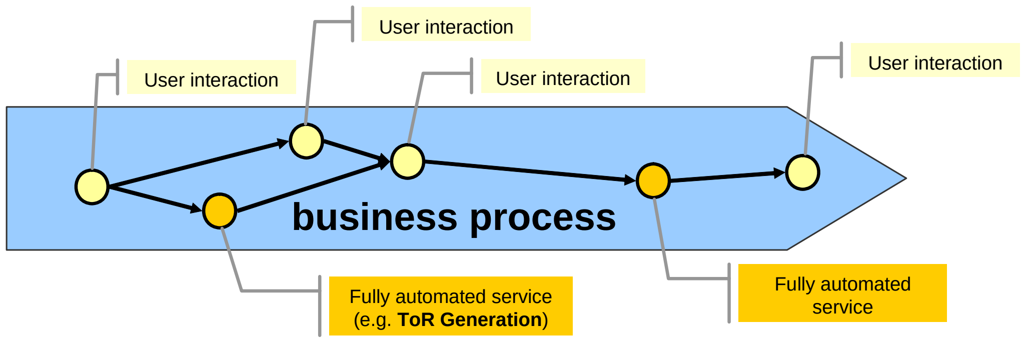


Figure 6: Consult a Student as a concrete Business Process

It can be easily seen that tasks with user interaction need to be in touch with the SOA Portal and the fully automated services can be realized in BPEL. But as mentioned above a suitable technology for realizing a business process with user interaction at some locations during its execution is missing when writing this document.

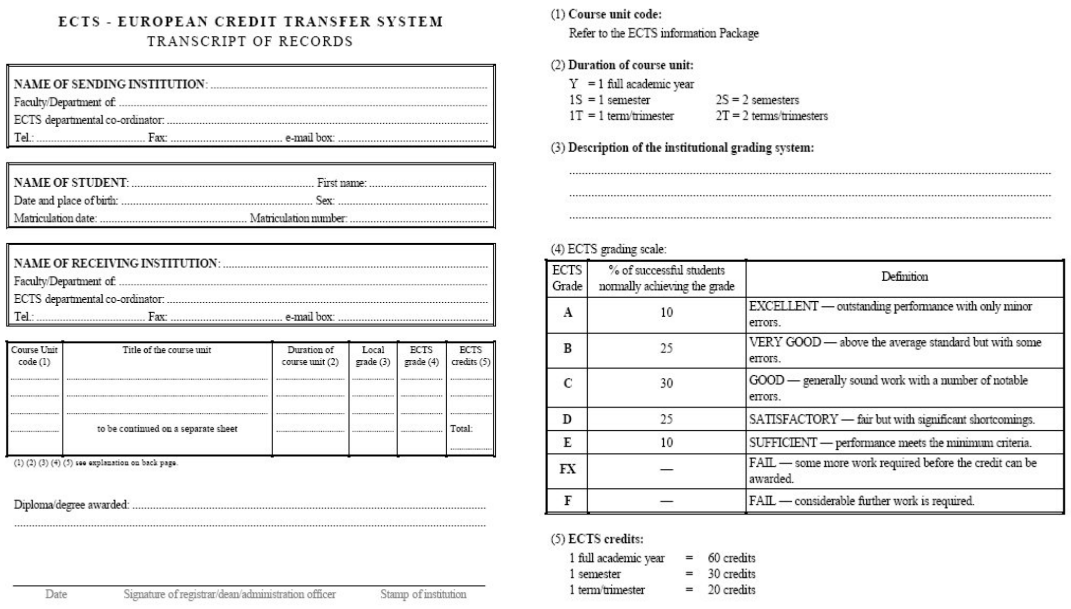
1.2 Specification of the Objectives

For the further work it is necessary to explore the objectives, use and potential user groups of the ToRService as well as in which manner the ToRService is reused in the specific business processes. Of course this also includes the GUIs that are motivated by the ToRService and that are an integral part of it.

1.2.1 Objectives and use

The scenario in which the ToRService should be integrated can be described as follows:
There exists a University SOA with many heterogeneous Web services (see Figure 3). There are core Web services and other Web services which came into existence by orchestrating some other Web services. The ToRService orchestrates four some services to provide a XML-based ToR. The core services provide access to the ToR-relevant databases. The necessary information can also be derived from legacy systems, but in this context an assumption was made that the data originates from some example databases.
Students or other users may deliver requests for ToR via the SOA-Portal to the ToRService directly or according to one activity of a large business process. To enable the user to interact with this service at least one GUI at the SOA Portal is needed. Because in this special case the SOA Portal does not exist at present, a special application called ToRClient is used to manage user interaction with the service.

The blank ToR in Figure 7 serves as a sample for the XML-based ToR and its XML schema.



ECTS - EUROPEAN CREDIT TRANSFER SYSTEM
TRANSCRIPT OF RECORDS

NAME OF SENDING INSTITUTION: ……………………………………
Faculty/Department of: ……………………………………
ECTS departmental co-ordinator: ……………………………………
Tel.: ……………… Fax: ……………… e-mail box: ………………

NAME OF STUDENT: ……………………… First name: ………………
Date and place of birth: ……………………… Sex: ………………
Matriculation date: ……………… Matriculation number: ………………

NAME OF RECEIVING INSTITUTION: ……………………………………
Faculty/Department of: ……………………………………
ECTS departmental co-ordinator: ……………………………………
Tel.: ……………… Fax: ……………… e-mail box: ………………

Course Unit code (1)	Title of the course unit	Duration of course unit (2)	Local grade (3)	ECTS grade (4)	ECTS credits (5)
…………	…………………………	…………	…………	…………	…………
…………	…………………………	…………	…………	…………	…………
…………	…………………………	…………	…………	…………	…………
…………	to be continued on a separate sheet	…………	…………	…………	Total: …………

(1) (2) (3) (4) (5) see explanation on back page.

Diploma/degree awarded: ……………………………………
……………………………………

Date Signature of registrar/dean/administration officer Stamp of institution

(1) **Course unit code:**
Refer to the ECTS information Package

(2) **Duration of course unit:**

Y = 1 full academic year	
1S = 1 semester	2S = 2 semesters
1T = 1 term/trimester	2T = 2 terms/trimesters

(3) **Description of the institutional grading system:**
……………………………………
……………………………………
……………………………………

(4) ECTS grading scale:

ECTS Grade	% of successful students normally achieving the grade	Definition
A	10	EXCELLENT — outstanding performance with only minor errors.
B	25	VERY GOOD — above the average standard but with some errors.
C	30	GOOD — generally sound work with a number of notable errors.
D	25	SATISFACTORY — fair but with significant shortcomings.
E	10	SUFFICIENT — performance meets the minimum criteria.
FX	—	FAIL — some more work required before the credit can be awarded.
F	—	FAIL — considerable further work is required.

(5) **ECTS credits:**

1 full academic year	=	60 credits
1 semester	=	30 credits
1 term/trimester	=	20 credits

Figure 7: A blank Transcript of Records

It should be clear that the second page looks always the same for all ToR of a university. That is the reason why it is left out for all further considerations.

1.2.2 Potential user groups and necessary roles

The ToRService is mainly used by employees of the university administration, student consultants and students.
The employees working at the student office have to provide information about the performed work of a student who asks for it. The same student should also be able to get this information via internet to unload the employees of the student office. And of course a student consultant needs the up to date information about a student’s performed work to give this student an accurate advice.

1.3 Overview about the SOA Part

Various users may use the ToRService. Therefore Figure 8 shows some potential business processes in which the ToR Service is needed. The ToRService plays a significant role in sub business areas like ‘Student self-administration’, ‘Student administration’ and ‘Student consultation’. It is of course possible to identify more relevant business processes when the whole business area of a university is explored but the discussed aspects these four are sufficient.

- Identification of business processes where the **ToRService** is needed
- **Derived goal:** to design the ToR Service that it fulfills the heterogenous requirements of all relevant business processes

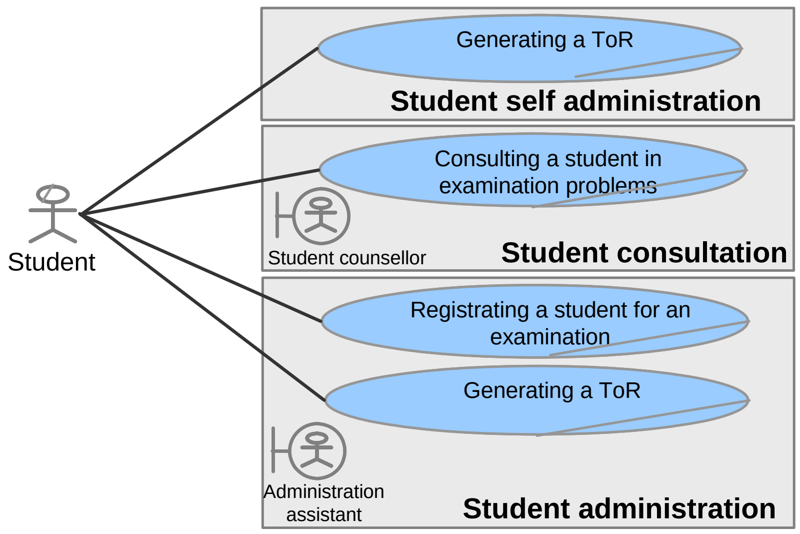


Figure 8: ToR-relevant Business Processes

Of course the student triggers all the business processes. The next figure embodies a different perspective of the ToR-relevant business processes. It shows the ToRService as a connector between the business processes of two different business partners. Furthermore Figure 9 displays a new business area is introduced. The business partner 'IT provider' is responsible for the technical realization of the IT supported business processes.

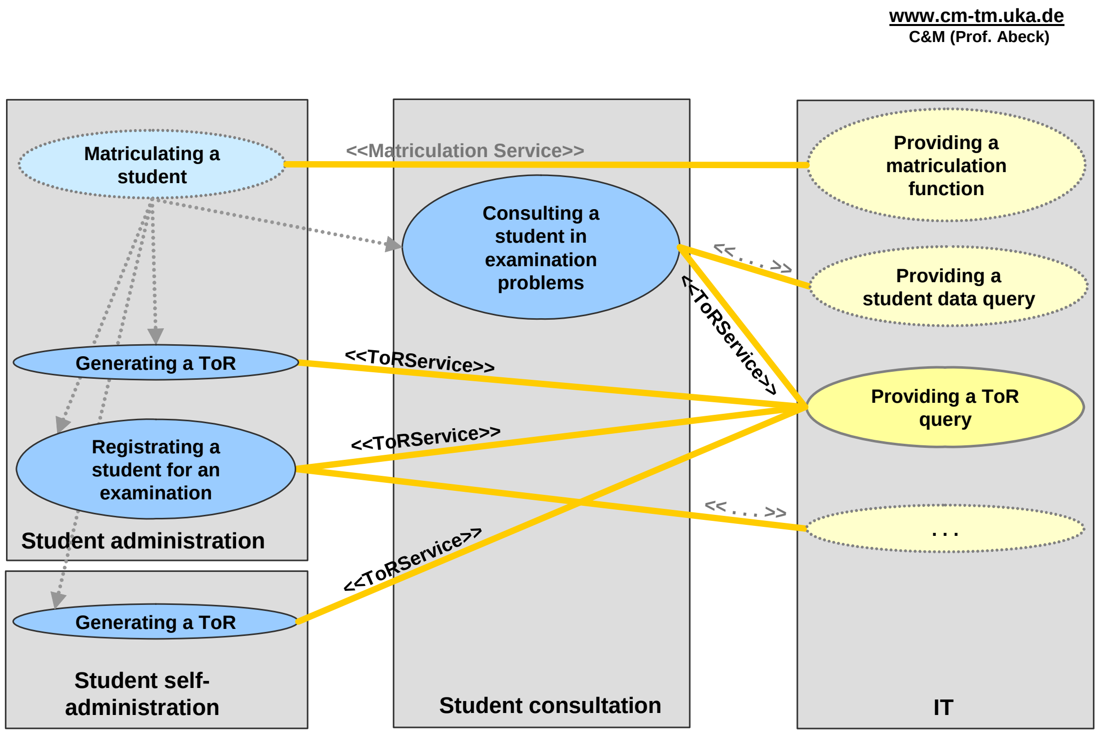


Figure 9: Interaction of Processes and Services in the ToR Context

Because the interaction with the ToRService is always very similar no matter where it is used, it is sufficient to observe here the generic use case 'Generating a ToR of a student' in which the main focus lays on the activities that take place on the user perspective. The technical aspects are modelled in chapter 2 using BPMN.

1.3.1 Use Case - Generating a ToR of a Student

In this section the appearance of the ToRService towards an arbitrary user is discussed. This contains also the way the ToRService is used. For this reason the description of the use case in Figure 10 provides a good entrance.

www.cm-tm.uka.de
C&M (Prof. Abeck)

Short description:	Automatic collection of ToR-relevant information and provision of this ToR in XML format
Actors:	StudentDBService, ExaminationDBService, OrganisationDBService, CourseUnitDBService
Triggers:	Student, student counsellor, administration assistant
Precondition:	User is logged in (authentified user)
Input information:	Student information, maybe information about receiving institution
Results:	ToR in XML format
Postcondition:	-
Procedure:	Enter student information and optional information about receiving institution Request ToR Receive ToR

Figure 10: Description of a Use Case – Generating a ToR of a Student

The use case describing activity diagram helps to understand which GUIs are needed and how the user can interact with the ToRService. It should be clarified that all the business objects have its seeds in the specific sections of the ToR in Figure 7.

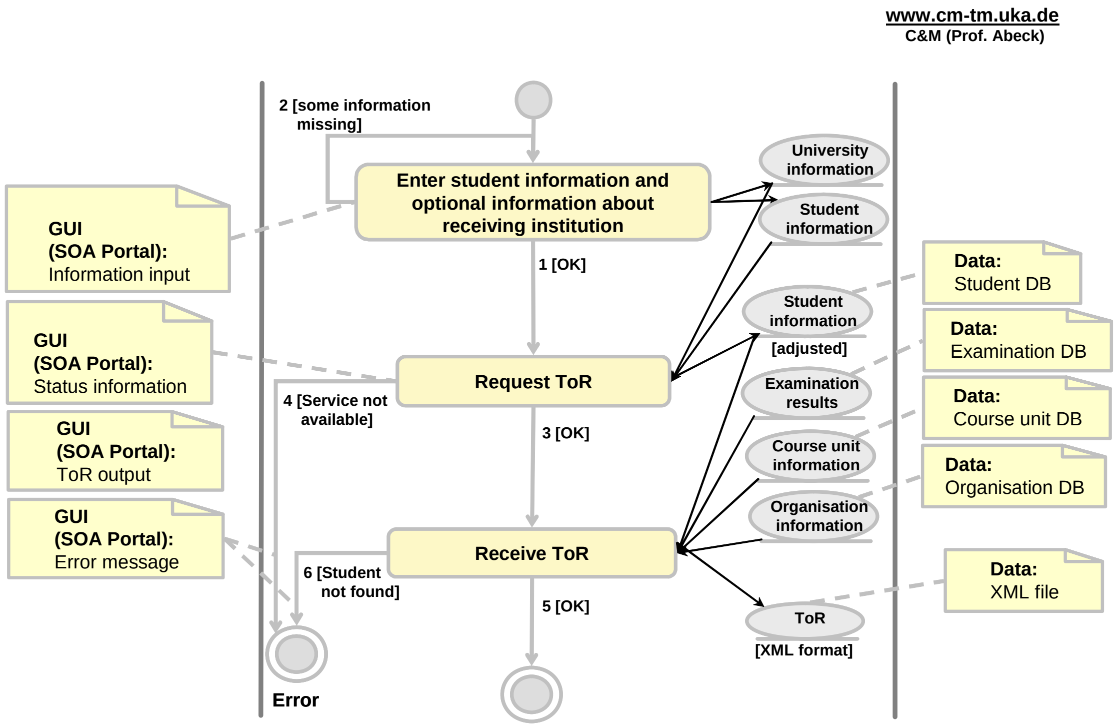


Figure 11: Generating a ToR for a Student

Enter student information and optional information about receiving institution

First of all the ToRService needs some information about the student. Optionally information about the receiving university can be entered. The necessary information about the sending institution is automatically added to the generated ToR.

www.cm-tm.uka.de
C&M (Prof. Abeck)

- Invitation for entering information about the student and maybe about the receiving institution
- Memorizing this informationen and latching it for further usage

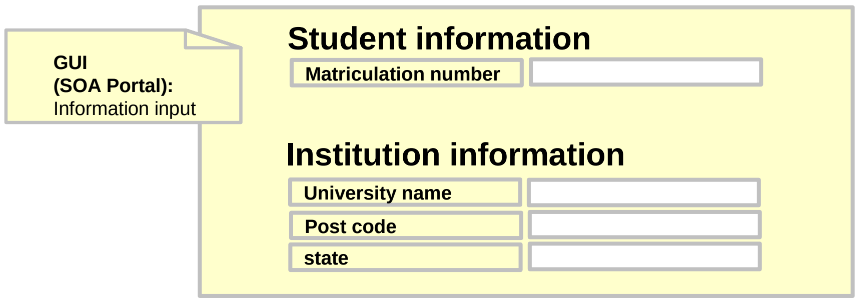


Figure 12: Activity – Enter Student Information and optional Information about receiving Institution

- **GUI (SOA Portal):** Information input
- **1[OK]:** All necessary information concerning the student was entered. In addition information about the receiving institution can be entered.
- **2[some information missing]:** Some information about the student is missing.
- **University information:** University name, post code and state
- **Student information:** Given name and surname of the student, matriculation number

Request ToR

After sending the pre-assign information the ToRService informs the user about the progress of the ToR generation.

www.cm-tm.uka.de
C&M (Prof. Abeck)

- ToR Service collects all necessary information
- The information is assembled into a XML ToR

GUI
(SOA Portal):
Status information
Collecting ToR informations…

Figure 13: Activity – Request ToR

- **GUI (SOA Portal):** Status information
- **3[OK]:** All services are available and all necessary information can be collected.
- **4[service not available]:** One of the involved services is not available, e.g. the ToRService is not reachable or the ToRService can not connect to one of the four core Web services.
- **University information:** University name, post code and state

- **Student information:** Given name and surname of the student, matriculation number

Receive ToR

After all information is collected and the requested ToR is generated, the ToRService returns it and the ToR is shown to the user.

www.cm-tm.uka.de
C&M (Prof. Abeck)

- Displaying the received ToR
- Persistent storage of the ToR in a XML file for further processing

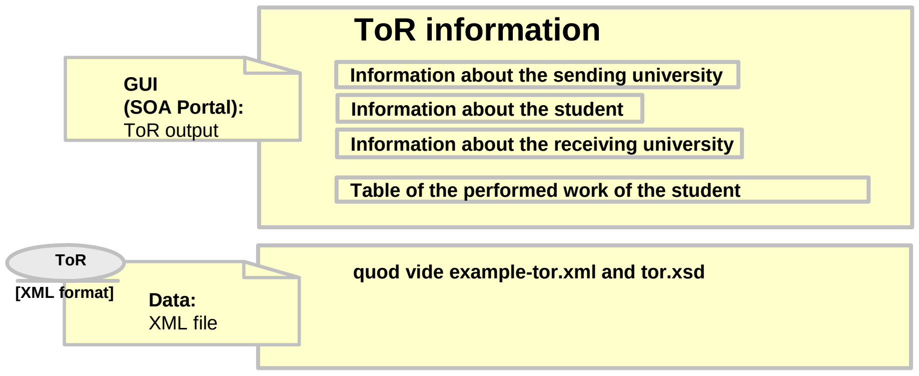

ToR information
GUI (SOA Portal): ToR output
Information about the sending university
Information about the student
Information about the receiving university
Table of the performed work of the student
ToR [XML format]
Data: XML file
quod vide example-tor.xml and tor.xsd

Figure 14: Activity – Receive ToR

- **GUI (SOA Portal):** ToR output
- **5[OK]:** ToR was successfully created. By the way a ToR can also be generated if the student has not yet performed an examination.
- **6[student not found]:** the student for which a ToR should be created does not exist in the database.
- **Student information:** Complete set of information about the student. Attributes in this business object are for example matriculation number, given name, surname, date of birth, place of birth and nationality.
- **Examination results:** Complete sets of information about all the examinations that the student has performed. Attributes in this business object are for example matriculation number, course unit code, local grade and ECTS grade.
- **Course unit information:** Complete set of information about the course units that correspond to the examinations. Attributes in this business object are for example course unit code, course unit title, course unit duration and ECTS credits.
- **Organisation information:** Complete set of information about the sending institution. Attributes in this business object are for example university name, post code, faculty name, ECTS coordinator's given name, ECTS coordinator's surname, ECTS coordinator's telephone number, ECTS coordinator's fax number and ECTS coordinator's e-mail address.
- **ToR in XML format:** The XML-representation of the student's ToR

1.4 Business Object Diagram

Concluding this chapter the business object diagram of Figure 15 provides an insight into the dependencies between the components of the ToR. Each component refers to a specific section of the ToR of Figure 7. Because the ToRService deals with the creation of ToR these components are also the business objects of this service.

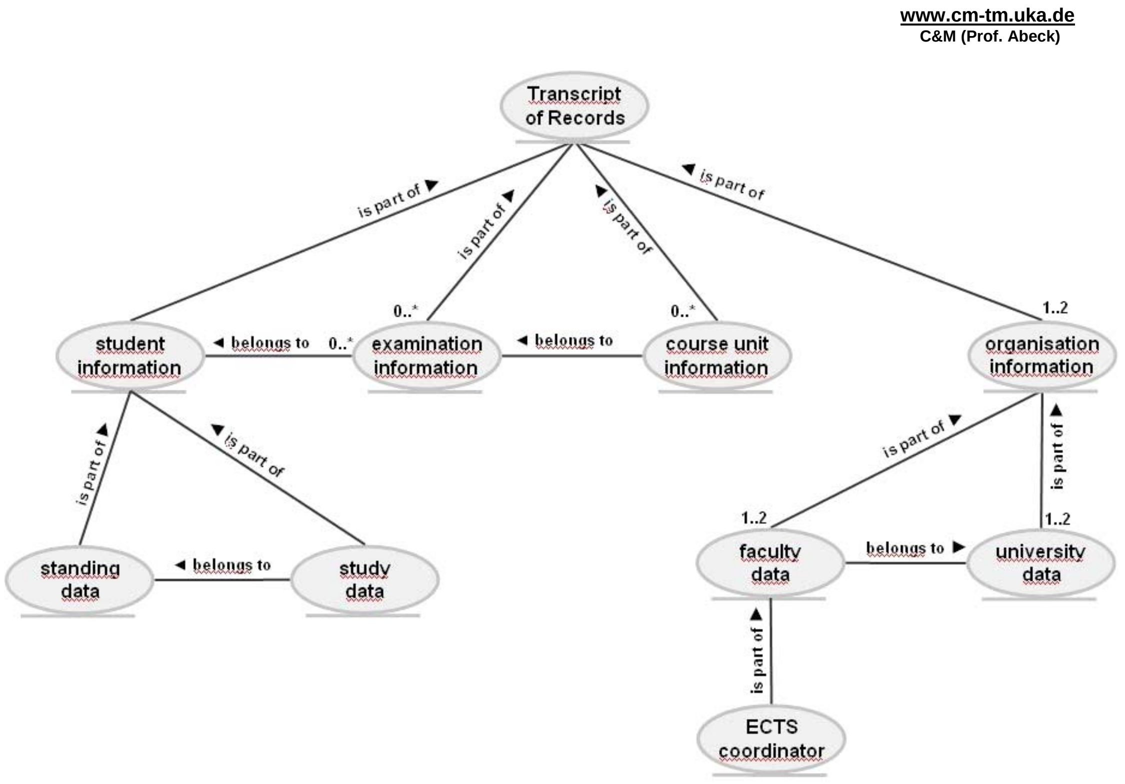


Figure 15: Business Object Diagram

Starting from this diagram, which contains all ToR-relevant information, all involved databases can be derived. In combination with the ToR of Figure 7 it is the basis for the XML ToR of the next chapter.

Concluding this chapter the reason why BPMN in this document is not used until design phase should be motivated in a few sentences. In the analysis phase of a SOA Part Doc the corresponding SOA Part with its services is explicitly described. The customer wants to know which services he gets and the developer has to have a specification of functionality and appearance that the customer expects from the services. Because this document concentrates on developing the ToRService not the maybe complex business processes in which this service is useful, it makes no sense to explore these business processes here. To keep the red thread in the design phase, where the ToRService is created, the modelling of the business processes is shifted into chapter 2, where a brief overview of an exemplary business process is given. BPMN is used to model the business process as well as the service itself.

2 DESIGN PHASE

In the design phase the realization of the new services is specified, but implementation details are blinded out as well as the used technologies. Of course in this chapter the customer is no longer aboard the ship.
Concerning the ToRService the whole design can be done by using BPMN, because in the next chapter a concept is introduced how the BPMN diagrams can be semi-automated transferred in BPEL code.

2.1 SOA Part Architecture

The specific layers of the ToRService as a SOA Part with its rough coherences are displayed in Figure 16. The ToRClient enables the user to interact with the ToRService because the USOA Portal does not exist up to time of realization. Beneath the ToRClient the relevant business processes are implied. The ToRService concatenates four core Web services and each of the several core Web services consists of some underlying components that enables access to the corresponding database.

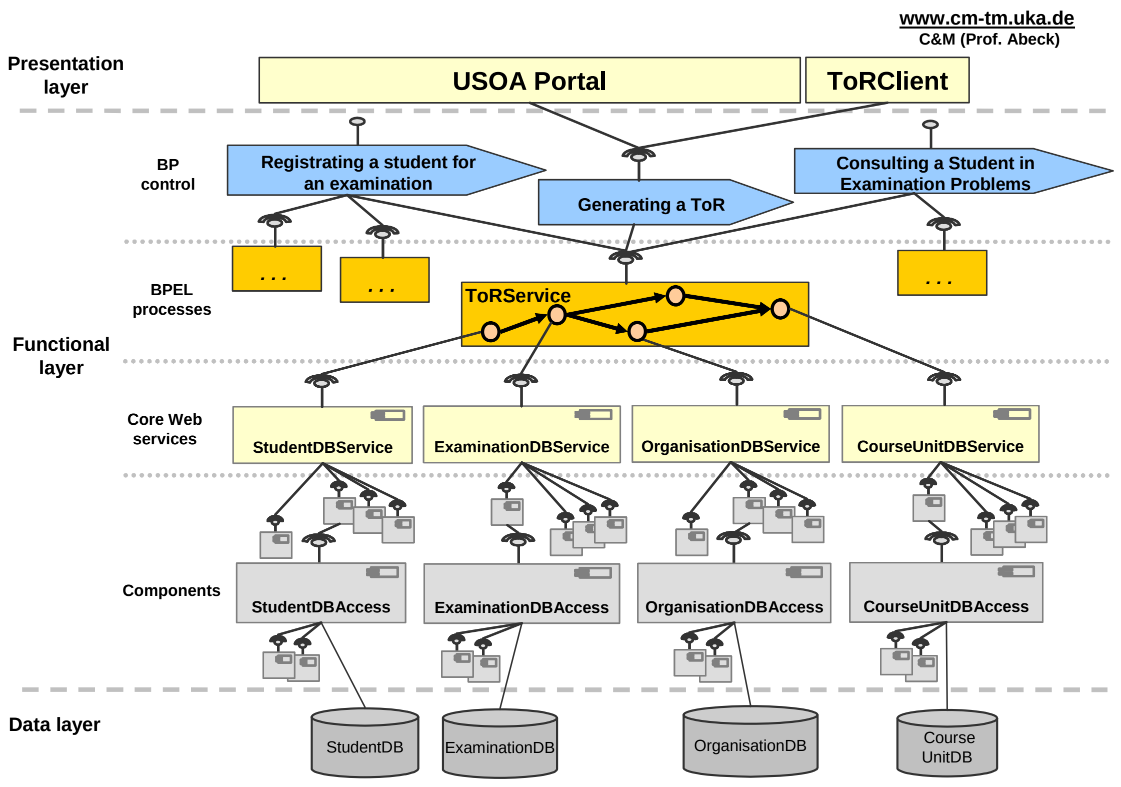


Figure 16: DESIGN PHASE - ToR Service as Part of the USOA

The main focus of the implementation phase lies on the business process ‘Generating a ToR’, even though other business processes are discussed in this chapter. The reason of looking at other business processes instead of the ‘Generating a ToR’ process is that the interesting design aspects can be better highlighted on them.

2.2 The ToRService

One of the business processes of Figure 8 and Figure 16 is ‘Consulting a student in examination problems’. Considering this business process as example, this chapter demonstrates how the ToRService works. Figure 17 shows the procedure in which a student as well as a student counsellor is involved. Some parts of this process can be supported by an IT system. First of all a student communicates a consultation request to the student counsellor or the student counsellor's secretary. Then the student gets an appointment. At the arranged date the student tells the counsellor about his problems. When the counsellor needs further information like the ToR of the student and information about current courses he collects them either manually or system supported and at the end he gives the student an advice.

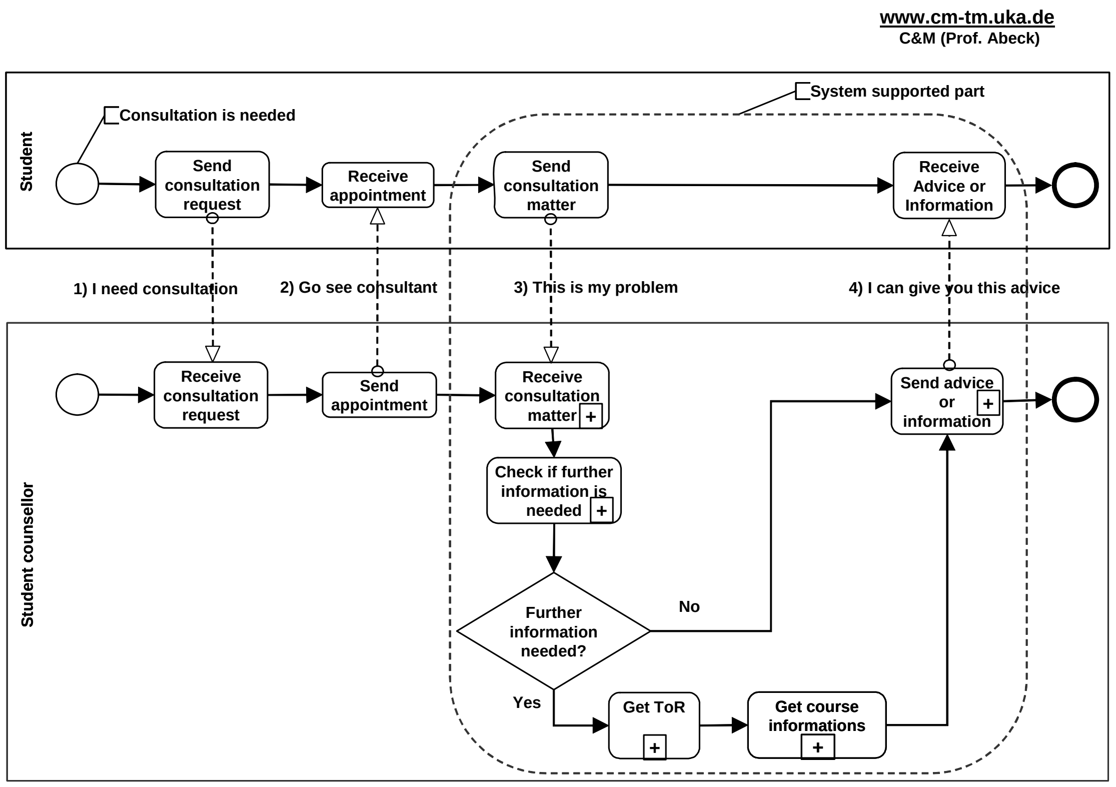


Figure 17: BPMN Collaborative Process - Consulting a Student in Examination Problems

Getting to the point where the student counsellor needs the ToR of the student, the ToRService comes into play and it is x-rayed in Figure 18.

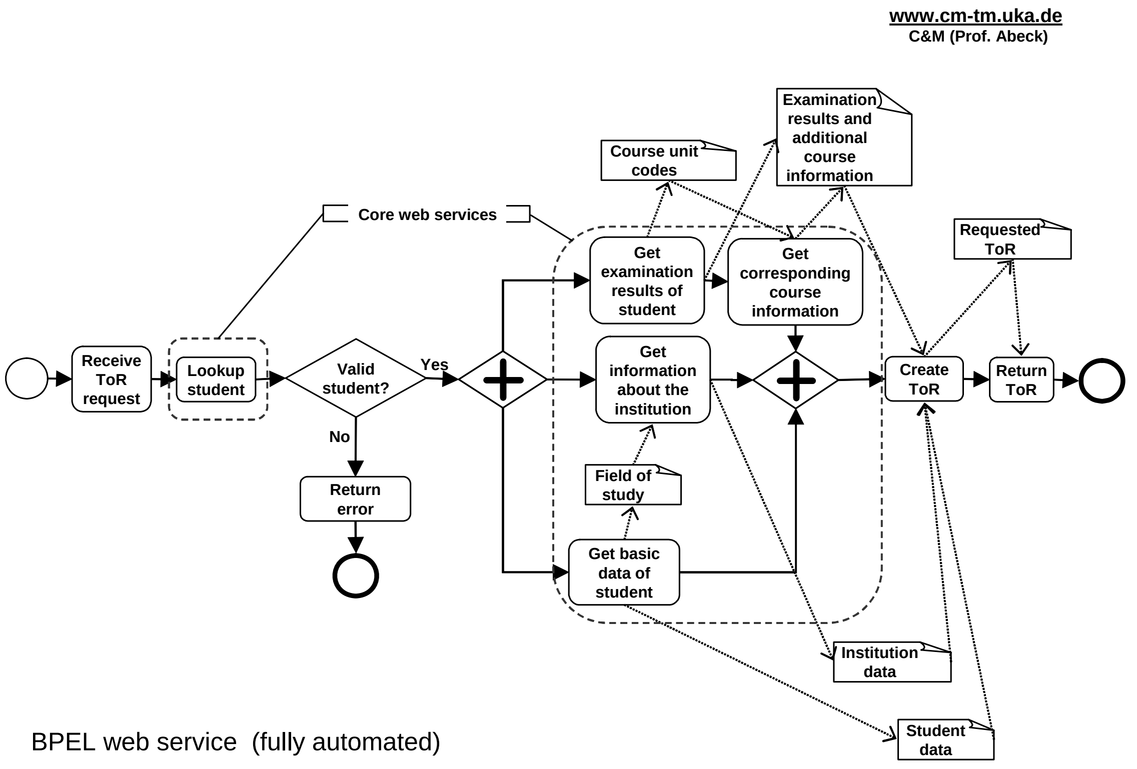
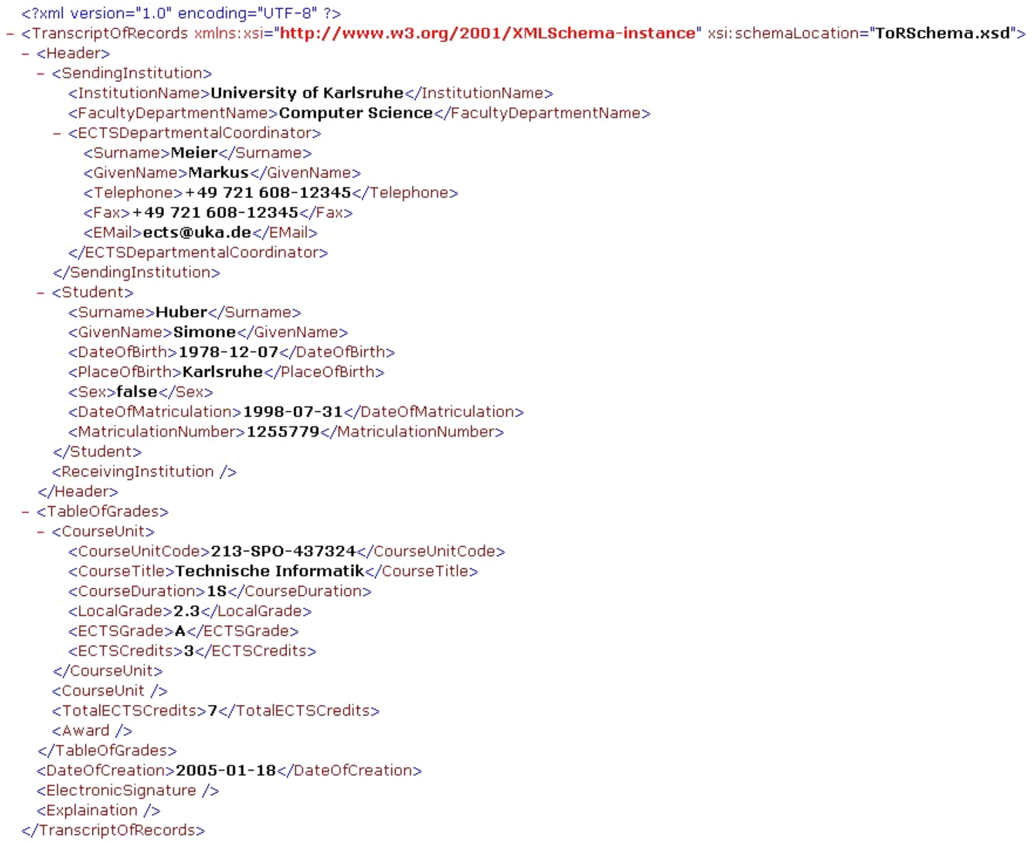

www.cm-tm.uka.de
C&M (Prof. Abeck)
Examination results and additional course information
Course unit codes
Core web services
Requested ToR
Get examination results of student
Get corresponding course information
Receive ToR request
Lookup student
Valid student?
Yes
No
Get information about the institution
Create ToR
Return ToR
Return error
Field of study
Get basic data of student
Institution data
Student data
BPEL web service (fully automated)

Figure 18: BPMN Internal Business Process – Get ToR

Every time the ToRService is called, which can happen out of various business processes, the following activities are executed:
As it can be seen after an incoming ToR request the ToRService executes a plausibility check by looking up the matriculation number of the request in the StudentDB. If the student who corresponds to the matriculation number is a valid student the core Web services are invoked to collect the necessary information, otherwise an error is returned. The ToR is generated after all information is available to the process. If every activity prospers a ToR in XML format is returned to the ToRClient.
Now it is time to take a close look at the XML schema of the ToR as shown in Figure 7. Because the second page of the ToR does not depend on the student information and is kind of static, this part of the ToR is suppressed in the following considerations. Figure 19 shows a shortened cut-out of an example ToR in XML format.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <TranscriptOfRecords xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ToRSchema.xsd">
  - <Header>
    - <SendingInstitution>
        <InstitutionName>University of Karlsruhe</InstitutionName>
        <FacultyDepartmentName>Computer Science</FacultyDepartmentName>
      - <ECTSDepartmentalCoordinator>
          <Surname>Meier</Surname>
          <GivenName>Markus</GivenName>
          <Telephone>+49 721 608-12345</Telephone>
          <Fax>+49 721 608-12345</Fax>
          <EMail>ects@uka.de</EMail>
        </ECTSDepartmentalCoordinator>
      </SendingInstitution>
    - <Student>
        <Surname>Huber</Surname>
        <GivenName>Simone</GivenName>
        <DateOfBirth>1978-12-07</DateOfBirth>
        <PlaceOfBirth>Karlsruhe</PlaceOfBirth>
        <Sex>false</Sex>
        <DateOfMatriculation>1998-07-31</DateOfMatriculation>
        <MatriculationNumber>1255779</MatriculationNumber>
      </Student>
      <ReceivingInstitution />
    </Header>
  - <TableOfGrades>
    - <CourseUnit>
        <CourseUnitCode>213-SPO-437324</CourseUnitCode>
        <CourseTitle>Technische Informatik</CourseTitle>
        <CourseDuration>1S</CourseDuration>
        <LocalGrade>2.3</LocalGrade>
        <ECTSGrade>A</ECTSGrade>
        <ECTSCredits>3</ECTSCredits>
      </CourseUnit>
      <CourseUnit />
      <TotalECTSCredits>7</TotalECTSCredits>
      <Award />
    </TableOfGrades>
    <DateOfCreation>2005-01-18</DateOfCreation>
    <ElectronicSignature />
    <Explaination />
  </TranscriptOfRecords>
```

Figure 19: XML Representation of a ToR

The corresponding XML schema can be found at APPENDIX A and at $ARCHIV\usoa\tor\bpel\ToRSchema.xsd.

2.3 The Core Web Services

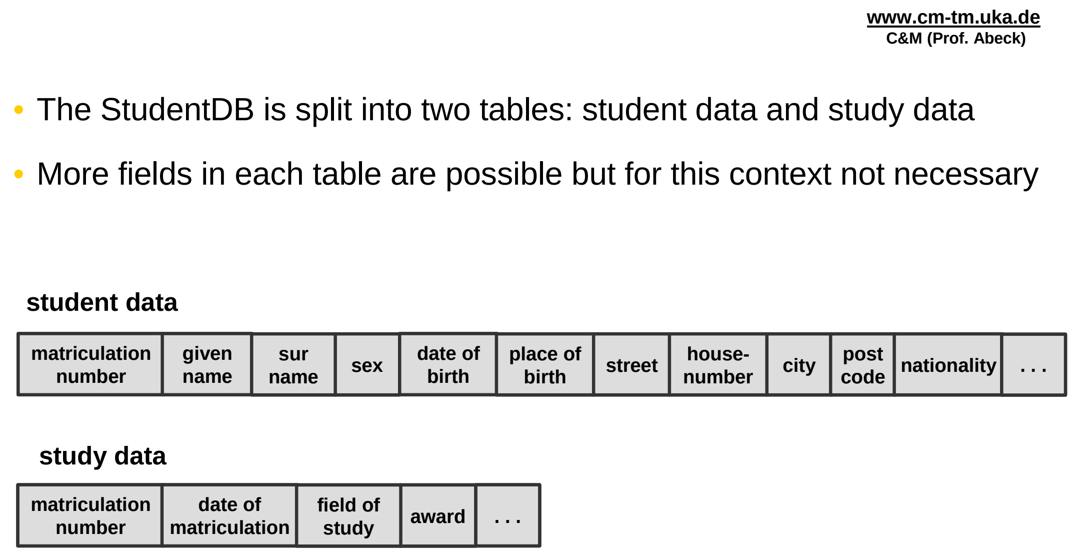
This subchapter deals with the core Web services which encapsulate the access to heterogeneous database management systems. Because these services are closely-coupled with the corresponding databases, they are also specified here. The superior aims when designing these services were:

- Re-usability of the core Web services, so that it makes sense to use them in other services than the ToRService as well
- Enabling uniform access to different database management systems
- Modeling the scenario of distributed databases all over the campus which seems to be very close to reality

Because all four services are very similar the following four subchapters are also very similar. Most of the classes that realize a Web service can be generated automatically out of the WSDL using appropriate tools. So in design phase only the WSDL file for each core Web service has to be specified. In the implementation phase the Apache Axis tool WSDL2Java is used to generate executable code. The complex types that represent a row in the corresponding database are also described in the WSDL file. An example of such a WSDL is printed out in APPENDIX B.

2.3.1 The StudentDBService

The StudentDB, which is the underlying database of the StudentDBService, is divided into the two tables ‘standingdata’ and ‘studydata’. One table contains all standing data of the student and the other contains all study relevant data. The complete database schema is shown in Figure 20.


www.cm-tm.uka.de
C&M (Prof. Abeck)

- The StudentDB is split into two tables: student data and study data
- More fields in each table are possible but for this context not necessary

student data

matriculation number	given name	sur name	sex	date of birth	place of birth	street	house-number	city	post code	nationality	. . .

study data

matriculation number	date of matriculation	field of study	award	. . .

Figure 20: Overview about the StudentDB

The component SetOfStudentDB is displayed in Figure 21, which represents a complete row of the StudentDB. It consists of two other components which themselves represent a complete row of the tables ‘studydata’ and ‘standingdata’.

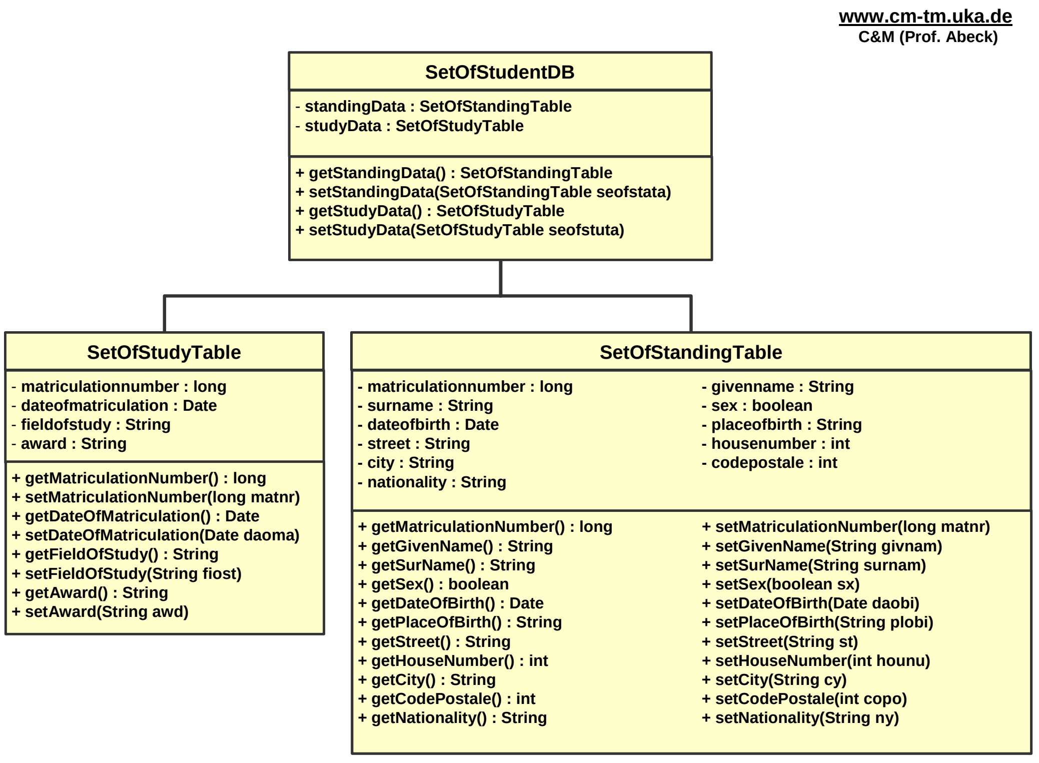

www.cm-tm.uka.de
C&M (Prof. Abeck)
SetOfStudentDB
- standingData : SetOfStandingTable
- studyData : SetOfStudyTable
+ getStandingData() : SetOfStandingTable
+ setStandingData(SetOfStandingTable seofstata)
+ getStudyData() : SetOfStudyTable
+ setStudyData(SetOfStudyTable seofstuta)
SetOfStudyTable
- matriculationnumber : long
- dateofmatriculation : Date
- fieldofstudy : String
- award : String
+ getMatriculationNumber() : long
+ setMatriculationNumber(long matnr)
+ getDateOfMatriculation() : Date
+ setDateOfMatriculation(Date daoma)
+ getFieldOfStudy() : String
+ setFieldOfStudy(String fiost)
+ getAward() : String
+ setAward(String awd)
SetOfStandingTable
- matriculationnumber : long
- surname : String
- dateofbirth : Date
- street : String
- city : String
- nationality : String
- givenname : String
- sex : boolean
- placeofbirth : String
- housenumber : int
- codepostale : int
+ getMatriculationNumber() : long
+ getGivenName() : String
+ getSurName() : String
+ getSex() : boolean
+ getDateOfBirth() : Date
+ getPlaceOfBirth() : String
+ getStreet() : String
+ getHouseNumber() : int
+ getCity() : String
+ getCodePostale() : int
+ getNationality() : String
+ setMatriculationNumber(long matnr)
+ setGivenName(String givnam)
+ setSurName(String surnam)
+ setSex(boolean sx)
+ setDateOfBirth(Date daobi)
+ setPlaceOfBirth(String plobi)
+ setStreet(String st)
+ setHouseNumber(int hounu)
+ setCity(String cy)
+ setCodePostale(int copo)
+ setNationality(String ny)

Figure 21: Components which encapsulate a whole Row of the StudentDB

Another component is displayed in Figure 22. It embodies the connection to the database and offers all needed methods to query information of the database.

- This is the sole component of the StudentDBService that has to be manually programmed
- Only those methods that are relevant for the ToRService are displayed

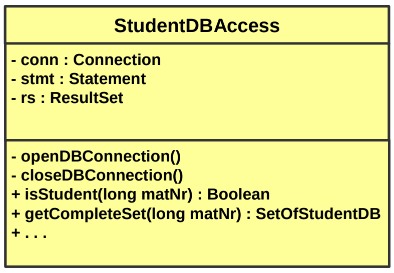


Figure 22: Component that encapsulates the Access to the StudentDB

2.3.2 The ExaminationDBService

The ExaminationDB consists only of one table named ‘resultdata’ in which the examination results of all students are saved. Figure 23 shows this table in detail.

- The ExaminationDB contains only one table
- More fields in this table are possible but for this context not necessary

result data

matriculation number	course unit code	local grade	ects grade	. . .

Figure 23: Overview about the ExaminationDB

The class SetOfExaminationDB is displayed in Figure 24, which represents a complete row of the ExaminationDB. Because a student may have written more than one examination the component SetOfExaminationDBArray is needed. Each SetOfExaminationDB represents a written examination.

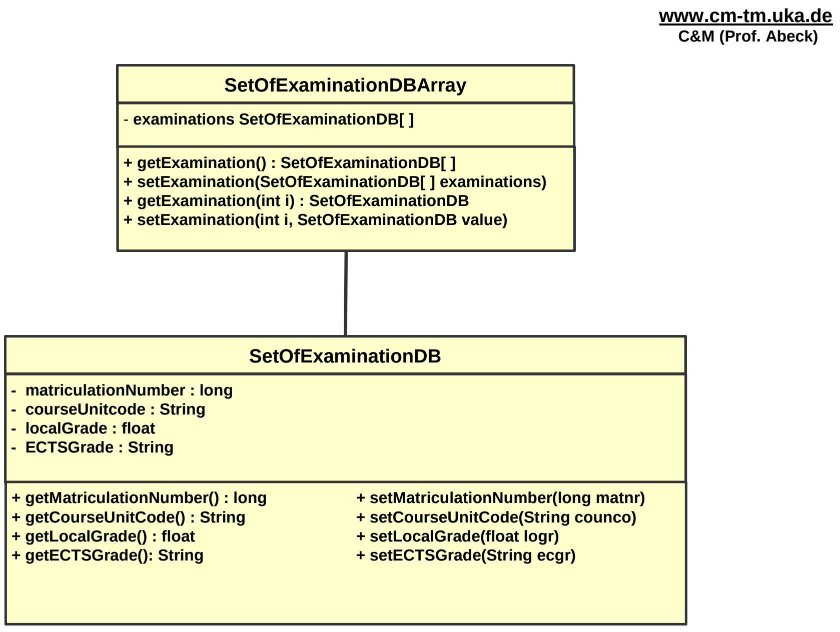
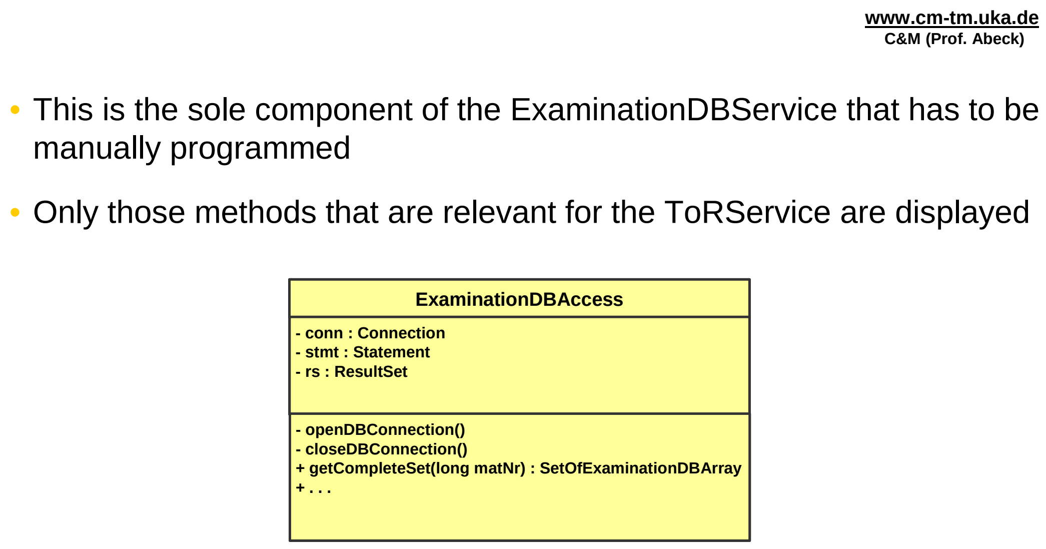

www.cm-tm.uka.de
C&M (Prof. Abeck)
SetOfExaminationDBArray
- examinations SetOfExaminationDB[ ]
+ getExamination() : SetOfExaminationDB[ ]
+ setExamination(SetOfExaminationDB[ ] examinations)
+ getExamination(int i) : SetOfExaminationDB
+ setExamination(int i, SetOfExaminationDB value)
SetOfExaminationDB
- matriculationNumber : long
- courseUnitcode : String
- localGrade : float
- ECTSGrade : String
+ getMatriculationNumber() : long
+ getCourseUnitCode() : String
+ getLocalGrade() : float
+ getECTSGrade(): String
+ setMatriculationNumber(long matnr)
+ setCourseUnitCode(String counco)
+ setLocalGrade(float logr)
+ setECTSGrade(String ecgr)

Figure 24: Components which encapsulate a whole Row of the ExaminationDB

Another component is displayed in Figure 25. It embodies the connection to the database and offers all needed methods to query information of the database.



- This is the sole component of the ExaminationDBService that has to be manually programmed
- Only those methods that are relevant for the ToRService are displayed

Figure 25: Component that encapsulates the Access to the ExaminationDB

2.3.3 The CourseUnitDBService

As displayed in Figure 26 the information about all the course units is stored into the table ‘coursedata’ of the CourseUnitDB.

- The CourseUnitDB contains only one table
- More fields in each table are possible but for this context not necessary

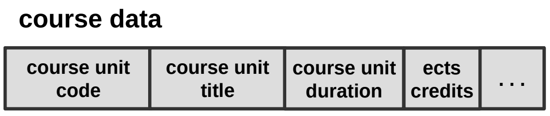


Figure 26: Overview about the CourseUnitDB

The class SetOfCourseUnitDB is displayed in Figure 27, which represents a complete row of the CourseUnitDB.

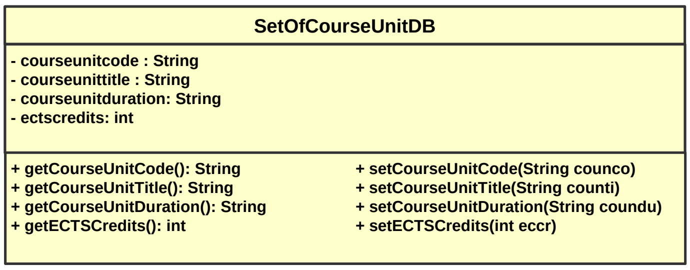

SetOfCourseUnitDB
- courseunitcode : String
- courseunittitle : String
- courseunitduration: String
- ectscredits: int
+ getCourseUnitCode(): String
+ getCourseUnitTitle(): String
+ getCourseUnitDuration(): String
+ getECTSCredits(): int
+ setCourseUnitCode(String counco)
+ setCourseUnitTitle(String counti)
+ setCourseUnitDuration(String coundu)
+ setECTSCredits(int eccr)

Figure 27: Component which encapsulates a whole Row of the CourseUnitDB

Another component is displayed in Figure 28. It embodies the connection to the database and offers all needed methods to query information of the database.

- This is the sole component of the CourseUnitDBService that has to be manually programmed
- Only those methods that are relevant for the ToRService are displayed

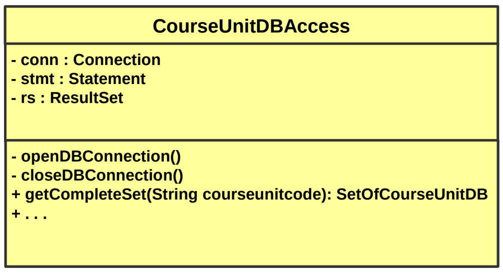
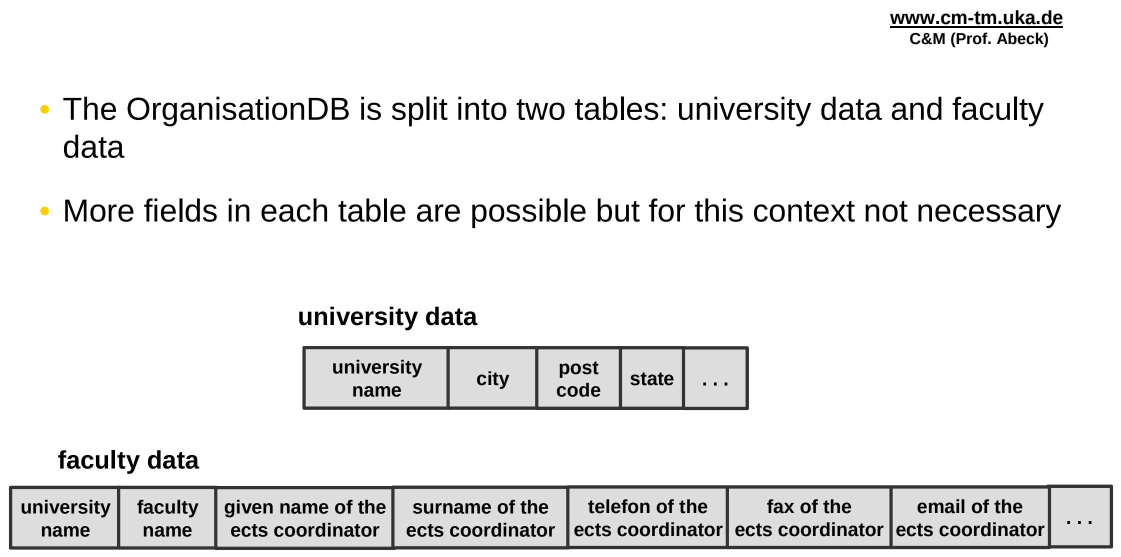


Figure 28: Component that encapsulates the Access to the CourseUnitDB

2.3.4 The OrganisationDBService

Last but not least all necessary institution information is stored into two tables of the OrganisationDB: ‘universitydata’ and ‘facultydata’.


www.cm-tm.uka.de
C&M (Prof. Abeck)

- The OrganisationDB is split into two tables: university data and faculty data
- More fields in each table are possible but for this context not necessary

university data

university name	city	post code	state	. . .

faculty data

university name	faculty name	given name of the ects coordinator	surname of the ects coordinator	telefon of the ects coordinator	fax of the ects coordinator	email of the ects coordinator	. . .

Figure 29: Overview about the OrganisationDB

The class SetOfOrganisationUnitDB is displayed in Figure 30, which represents a complete row of the OrganisationDB. It consists of two other components which themselves represent a complete row of the tables ‘universitydata’ and ‘facultydata’.

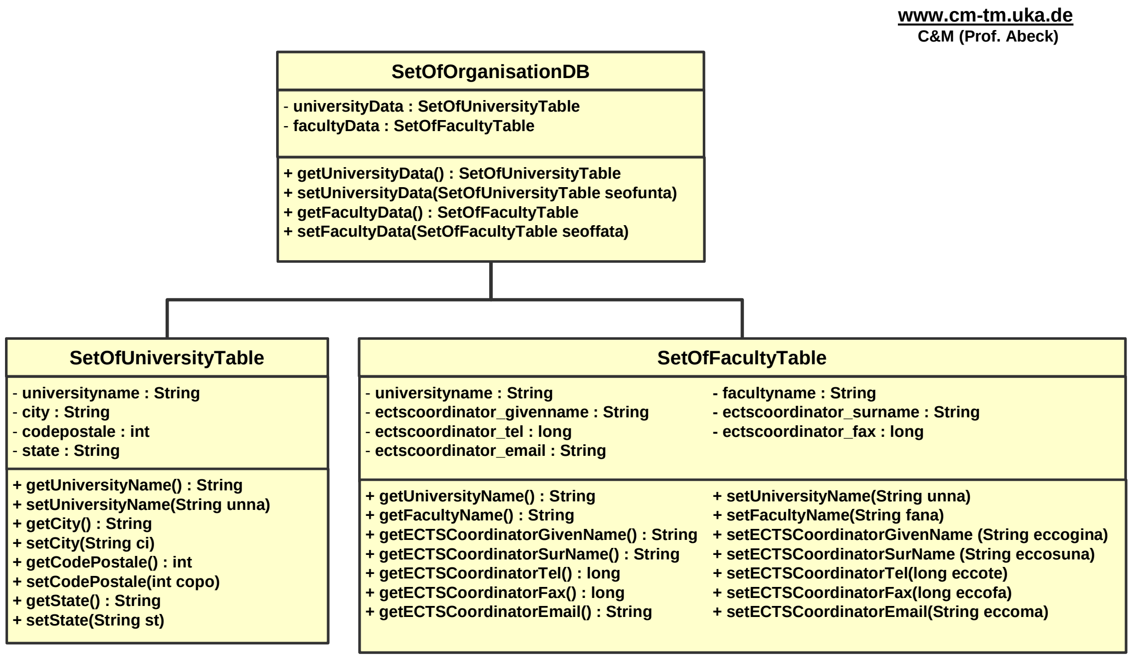

www.cm-tm.uka.de
C&M (Prof. Abeck)
SetOfOrganisationDB
- universityData : SetOfUniversityTable
- facultyData : SetOfFacultyTable
+ getUniversityData() : SetOfUniversityTable
+ setUniversityData(SetOfUniversityTable seofunta)
+ getFacultyData() : SetOfFacultyTable
+ setFacultyData(SetOfFacultyTable seoffata)
SetOfUniversityTable
- universityname : String
- city : String
- codepostale : int
- state : String
+ getUniversityName() : String
+ setUniversityName(String unna)
+ getCity() : String
+ setCity(String ci)
+ getCodePostale() : int
+ setCodePostale(int copo)
+ getState() : String
+ setState(String st)
SetOfFacultyTable
- universityname : String
- ectscoordinator_givenname : String
- ectscoordinator_tel : long
- ectscoordinator_email : String
- facultyname : String
- ectscoordinator_surname : String
- ectscoordinator_fax : long
+ getUniversityName() : String
+ getFacultyName() : String
+ getECTSCoordinatorGivenName() : String
+ getECTSCoordinatorSurName() : String
+ getECTSCoordinatorTel() : long
+ getECTSCoordinatorFax() : long
+ getECTSCoordinatorEmail() : String
+ setUniversityName(String unna)
+ setFacultyName(String fana)
+ setECTSCoordinatorGivenName (String eccogina)
+ setECTSCoordinatorSurName (String eccosuna)
+ setECTSCoordinatorTel(long eccote)
+ setECTSCoordinatorFax(long eccofa)
+ setECTSCoordinatorEmail(String eccoma)

Figure 30: Components which encapsulate a whole Row of the OrganisationDB

Another component is displayed in Figure 31. It embodies the connection to the database and offers all needed methods to query information of the database.

- This is the sole component of the OrganizationDBService that has to be manually programmed
- Only those methods that are relevant for the ToRService are displayed

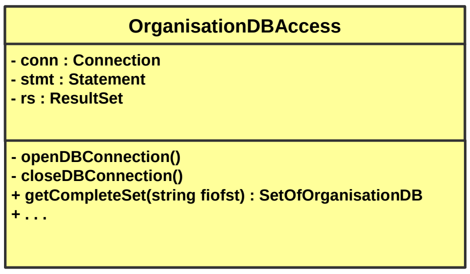


Figure 31: Component that encapsulates the Access to the OrganizationDB

3 IMPLEMENTATION PHASE

3.1 The ToRService

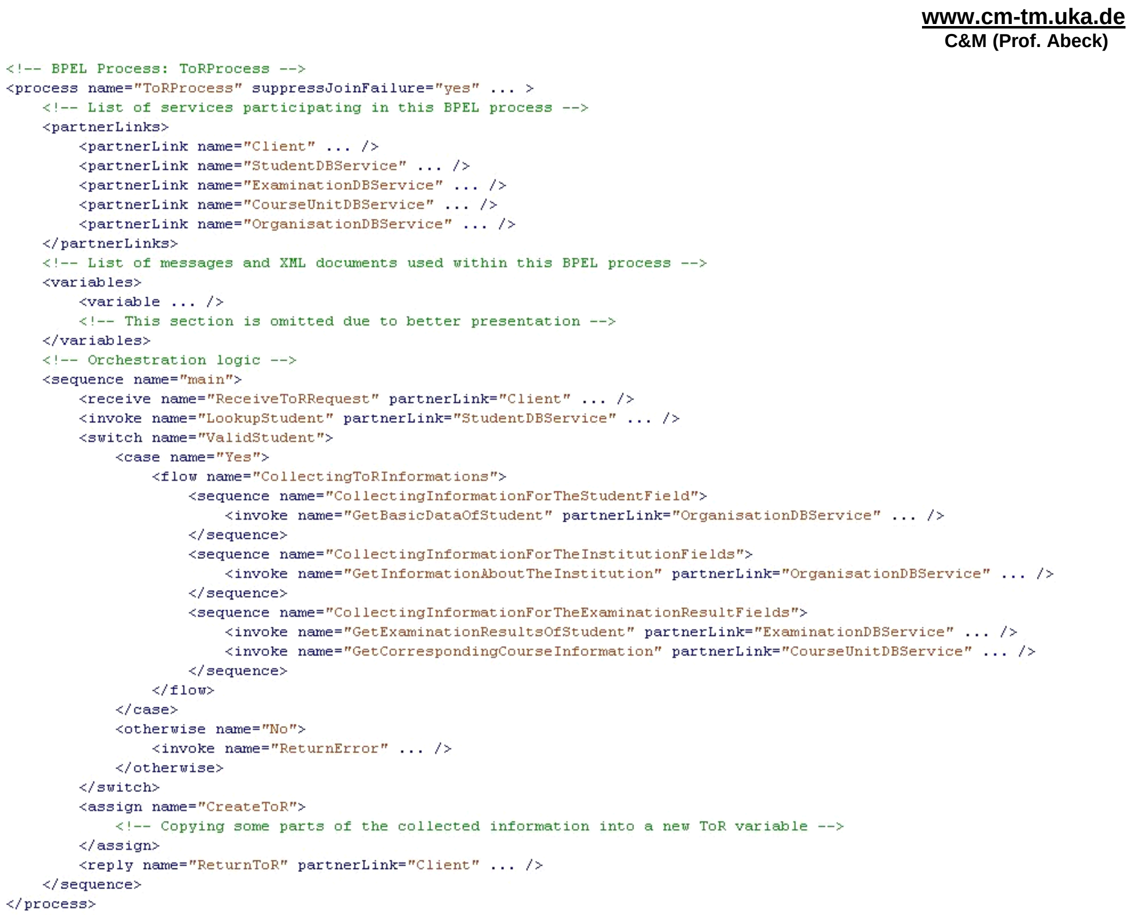
The ToRService uses several Web services that have been designed in Chapter 2 to collect all information it needs to generate a ToR. The necessary Web services offer their functionalities as described in the corresponding WSDL documents using SOAP interfaces. The process internal message flow is also done via SOAP. And last but not least the ToRService should provide the requested ToR at its own SOAP interface.
Of course it is possible to realize the ToRService in Java, .NET or any other conventional programming language. But in this project BPEL was used because of several reasons:

- BPEL programming is done in XML, which is a platform independent standard that can be used easily to interchange data between heterogeneous applications
- Variables within a BPEL process are represented by XML objects
- Message passing within a BPEL process is realized in SOAP so that at this point no converting overhead arises
- It is possible to systematically generate BPEL code out of BPMN diagrams
- One of the specifications for realizing this service was to use BPEL

Some parts of the analysis and design phases were done using BPMN. Reaching the implementation phase a mechanism or concept is needed to generate BPEL code for the business processes out of a BPMN representation in a mostly automatic way, because one main idea behind BPMN is to reduce the gap between the business process design and the implementation. The rest of this subchapter illustrates how the BPMN graphs can be converted to BPEL code under the precondition to maximize the automatically executable parts. The lack of available software products, which can translate BPMN to BPEL in a way so that this code can be deployed or even efficiently developed, causes a work-around.

First of all it should be mentioned that there are two different categories of relevant tools for this procedure: BPMN modelling tools to create the BPMN graph and BPEL design tool for managing the specific BPEL aspects and concerns. Let’s begin with our ToR process that is shown in Figure 18, which was developed with the BPMN modelling tool Popkin’s System Architect. At this point it is possible to map the information represented in the BPMN graph by hand using the mapping rules defined in [BPMN1.0]. Some BPMN modelling tools like Popkin’s System Architect offer functions to generate BPEL code out of BPMN automatically. But there is one hitch. The generated code contains almost no information about the SOAP interface of the orchestrated elements in it. This work can be done machine supported because all of the needed information is listed in the Web service description if we are talking about Web services that already exist.

That is the point where a BPEL design tool like the Oracle BPEL Designer comes into play. The alternative of using such a tool is to add the remaining information manually. This is not efficiently practicable. Most of the design tools allow importing existing BPEL code which can be generated by Popkin’s System Architect as described. The BPEL code is parsed and then displayed as a graph. Unfortunately this graph is not in BPMN notation but in a proprietary one. Now the WSDL documents of the participating core Web services can be imported and missing information like variables and messages can be added to the BPEL process. Furthermore functionalities for testing and validating the created BPEL process as well as a function for deploying the process to the used BPEL engine are available.


www.cm-tm.uka.de
C&M (Prof. Abeck)

```
<!-- BPEL Process: ToRProcess -->
<process name="ToRProcess" suppressJoinFailure="yes" ... >
    <!-- List of services participating in this BPEL process -->
    <partnerLinks>
        <partnerLink name="Client" ... />
        <partnerLink name="StudentDBService" ... />
        <partnerLink name="ExaminationDBService" ... />
        <partnerLink name="CourseUnitDBService" ... />
        <partnerLink name="OrganisationDBService" ... />
    </partnerLinks>
    <!-- List of messages and XML documents used within this BPEL process -->
    <variables>
        <variable ... />
        <!-- This section is omitted due to better presentation -->
    </variables>
    <!-- Orchestration logic -->
    <sequence name="main">
        <receive name="ReceiveToRRequest" partnerLink="Client" ... />
        <invoke name="LookupStudent" partnerLink="StudentDBService" ... />
        <switch name="ValidStudent">
            <case name="Yes">
                <flow name="CollectingToRInformations">
                    <sequence name="CollectingInformationForTheStudentField">
                        <invoke name="GetBasicDataOfStudent" partnerLink="OrganisationDBService" ... />
                    </sequence>
                    <sequence name="CollectingInformationForTheInstitutionFields">
                        <invoke name="GetInformationAboutTheInstitution" partnerLink="OrganisationDBService" ... />
                    </sequence>
                    <sequence name="CollectingInformationForTheExaminationResultFields">
                        <invoke name="GetExaminationResultsOfStudent" partnerLink="ExaminationDBService" ... />
                        <invoke name="GetCorrespondingCourseInformation" partnerLink="CourseUnitDBService" ... />
                    </sequence>
                </flow>
            </case>
            <otherwise name="No">
                <invoke name="ReturnError" ... />
            </otherwise>
        </switch>
        <assign name="CreateToR">
            <!-- Copying some parts of the collected information into a new ToR variable -->
        </assign>
        <reply name="ReturnToR" partnerLink="Client" ... />
    </sequence>
</process>
```

Figure 32: IMPLEMENTATION PHASE - BPEL Code for the ToRService

The result of this approach is the in Figure 32 displayed BPEL process for the ToRService. It should be mentioned that the displayed code is fragmentary. Most of the attributes, variables and assigns are omitted for better readability. Manifesting by the surppressJoinFailure attribute, the complete error handling is also left out.

As mentioned above to top the BPEL code off, which was generated out of BPMN, and to make it executable the Oracle BPEL Designer was used. The whole BPEL process is shown in Figure 33 with its four scopes.

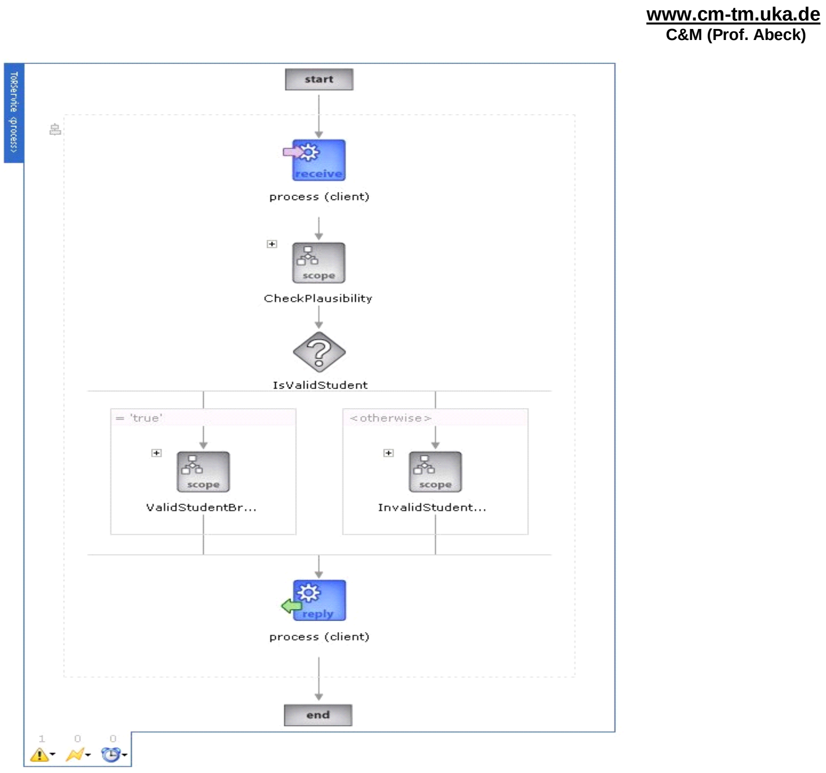


Figure 33: The ToRService modelled with the Oracle BPEL Designer

Scopes are helpful to handle errors individually in subject to the location they occurs. An example for an easy error handling is shown in Figure 34. The scope in which an error emerges is aborted. A catchAll causes the error value to be written to a specific variable.

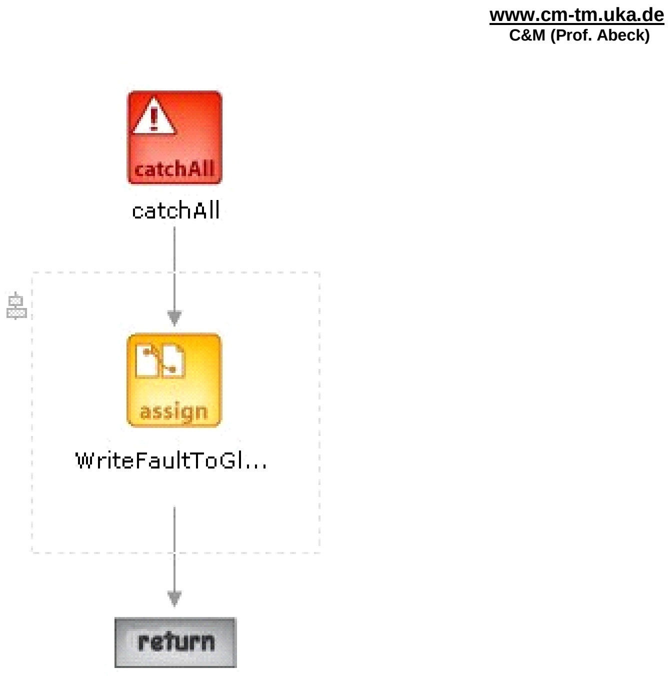


Figure 34: Example of a simple Error Handling

After receiving a ToR request from the ToRClient a plausibility check is performed. This is the first scope. Basically the StudentDBService is invoked which tells the ToRService if the assigned matriculation number corresponds to a valid student. After that the result is written in a variable that is visible in other scopes, too.

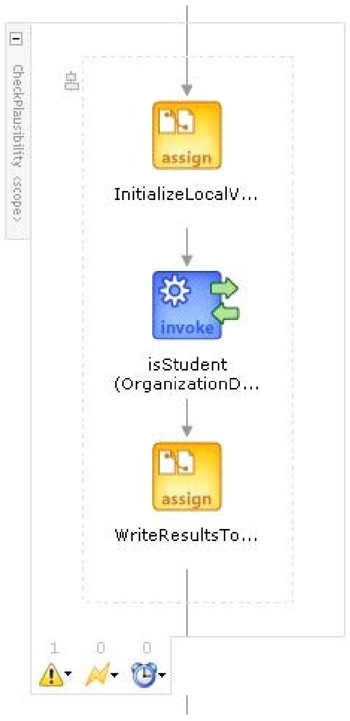


Figure 35: The Scope in which the Plausibility Check is performed

As it can be seen in Figure 33 if the plausibility check returns false the scopes named InvalidStudentBranch is executed, which only contains an empty tag because nothing has to be done. In the other case the ValidStudentBranch is executes and the necessary information is collected.

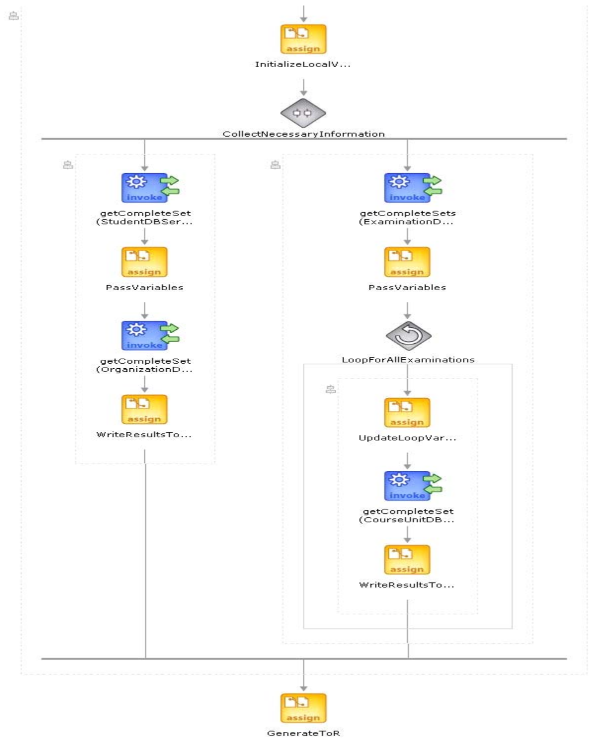

InitializeLocalV...
CollectNecessaryInformation
getCompleteSet (StudentDBSer...
PassVariables
getCompleteSet (OrganizationD...
WriteResultsTo...
getCompleteSets (ExaminationD...
PassVariables
LoopForAllExaminations
UpdateLoopVar...
getCompleteSet (CourseUnitDB...
WriteResultsTo...
GenerateToR

Figure 36: The Scope in which the ToR-relevant Information is collected

First of all some scope internal variables are initialized. Then two parallel sequences are executed. In contrast to the corresponding BPMN graph of Figure 18 messages can not be easily passed from one branch to another. So the OrganizationDBService is executed after the StudentDBService because it needs the information about the field of study to collect information about the considered faculty. After that both results are written to a global variable.
The second sequence of Figure 36 at first queries all the performed examinations of the student by invoking the ExaminationDBService. Then it iterates through all these results to gain the corresponding course information using the CourseUnitDBService. Of course the results are also written to a global variable.
Concluding this scope a ToR has to be generated out of the gained information. At this point various approaches are possible. The first approach was to do this using the assign tag of BPEL. Because the BPEL specification says that in the TO tags inside of assign tags cannot contain XPath expressions this approach is impossible. So it is impossible to handle dynamical indexing into arrays which means that the content of the course unit tags of the XML ToR cannot be filled out at runtime. But this had to be done at runtime because the number of examination results is not known until runtime.
So another approach has to be taken. One idea is to do the transformation using another Web service. But to stay in the XML technology in this context XSLT is used. A XSLT program can be invoked using a special XPath expression, so it is not necessary to invoke the XSLT processor like an external Web services. When the ToR is generated it is returned to the caller and the process terminates.
The whole BPEL process to the ToRService has an amount of about 300 LoC which includes the files ToRService.bpel, ToRGenerator.xslt and ToRService.wsdl. The folder were this files can be found is $ARCHIV\usoa\tor\bpel\ToRService.
As a BPEL engine, which is needed to run this BPEL process, the Oracle Process Manager was installed. The deployment and use is described in Chapter 4.

3.2 The Core Web Services

The implementation of the core Web services is done in Java. It also may be possible to implement them in .NET or other conventional programming languages as long as they provide a SOAP interface.
All core Web services are Java Servlets. The usage of Apache Axis as SOAP engine in addition to Apache Jakarta Tomcat as a Servlet container simplifies the implementation of the services. Regular Java classes with at least one public method are automatically transformed into Web services with SOAP interfaces. All as public declared methods together form the SOAP interface of the service. But this procedure also causes problems, because BPEL and the Oracle Process Manager are very prudish with the WSDL files and the SOAP messages. It is advisable to create the WSDL first and then auto-generate the Java stub and skeletons with a tool like WSDL2Jave of the Apache Axis framework. This tool takes a description of a Web service written in WSDL and emits Java artefacts used to access the Web service. The WSDL files can be found at the corresponding subfolders of $Archiv\usoa\tor\java\src\de\cm\services.
Because the procedure is done for all four core Web services and the results are very similar, the generic class model which is displayed in Figure 37 gives a good overview about the implementation of all four web services.

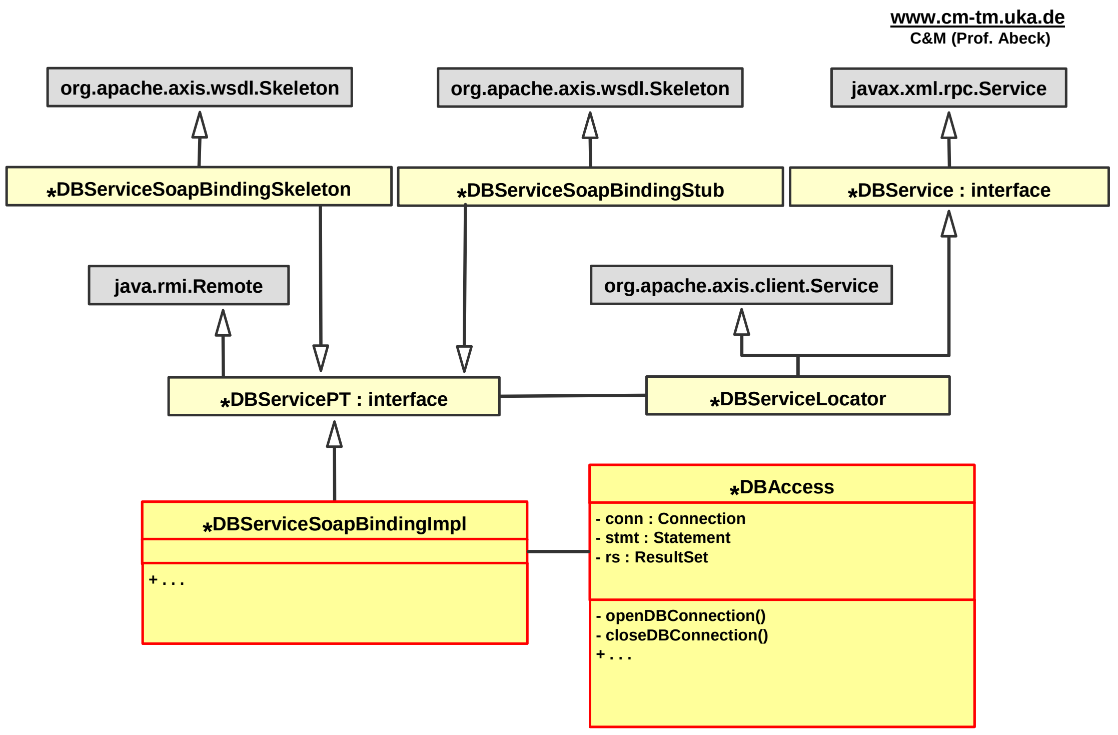


Figure 37: Generic Class Model for any of the four core Web Services

Only these classes that have red-dyed borders have to be manually programmed. The JavaBeans that encapsulate the query results of each service can be automatically generated as well. But in these figures they are omitted. The term *DBAccess for example indicates that this class could be the StudentDBAccess, the ExaminationDBAccess, the CourseUnitDBAccess or the OrganizationDBAccess.
The underlying databases are realized with the database management system MySQL. The schemas are displayed in Figure 38 to Figure 41.

www.cm-tm.uka.de
C&M (Prof. Abeck)

```
CREATE DATABASE studentdb;
USE studentdb;
CREATE TABLE standingdata (
                matriculationnumber INT ZEROFILL,
                givenname VARCHAR(64) CHARACTER SET utf8,
                surname VARCHAR(64) CHARACTER SET utf8,
                sex BOOLEAN,
                dateofbirth DATE,
                placeofbirth VARCHAR(64) CHARACTER SET utf8,
                street VARCHAR(64) CHARACTER SET utf8,
                housenumber SMALLINT,
                city VARCHAR(64) CHARACTER SET utf8,
                codepostale INT,
                nationality CHAR(64) CHARACTER SET utf8,
                PRIMARY KEY (matriculationnumber)
);
CREATE TABLE studydata (
                matriculationnumber INT ZEROFILL,
                dateofmatriculation DATE,
                fieldofstudy VARCHAR(64) CHARACTER SET utf8,
                award VARCHAR(64) CHARACTER SET utf8,
                PRIMARY KEY (matriculationnumber)
);
INSERT INTO standingdata VALUES ('1','Rahn','Helmut','1','1929-08-16','Wuppertal','Porscheplatz','1','Essen','45121','deutsch');
INSERT INTO standingdata VALUES ('2','Luxemburg','Rosa','0','1871-03-05','Zamost','Heinrich-Heine Strasse','23','Weimar','99401','polnisch');
INSERT INTO standingdata VALUES ('3','Zuse','Konrad','1','1910-07-22','Berlin','Mittelstrasse','9','Hünfeld','99406','deutsch');
INSERT INTO studydata VALUES ('1','1944-03-29','Sport',NULL);
INSERT INTO studydata VALUES ('2','1957-03-29','Staatswissenschaft',NULL);
INSERT INTO studydata VALUES ('3','1925-03-29','Informatik',NULL);
```

Figure 38: MySQL Database Schema for the StudentDB

www.cm-tm.uka.de
C&M (Prof. Abeck)

```
CREATE DATABASE examinationdb;
USE examinationdb;
CREATE TABLE resultdata (
                matriculationnumber INT ZEROFILL,
                courseunitcode VARCHAR(16),
                localgrade ENUM('0.7','1.0','1.3','1.7','2.0','2.3','2.7','3.0','3.3','3.7','4.0','5.0'),
                ectsgrade ENUM('A','B','C','D','E','FX','F'),
                PRIMARY KEY (matriculationnumber, courseunitcode)
);
INSERT INTO resultdata VALUES ('1','341-SPO-87653','2.3','A');
INSERT INTO resultdata VALUES ('1','873-SPO-9735','2.3','A');
INSERT INTO resultdata VALUES ('2','893-SWI-2243','1.7','C');
INSERT INTO resultdata VALUES ('2','27-SWI-2454','3.7','C');
INSERT INTO resultdata VALUES ('2','873-SPO-9735','4.0','B');
INSERT INTO resultdata VALUES ('3','34-INFO-29775','3.0','E');
```

Figure 39: MySQL Database Schema for the ExaminationDB

www.cm-tm.uka.de
C&M (Prof. Abeck)

```
CREATE DATABASE courseunitdb;
USE courseunitdb;
CREATE TABLE courseunitdata (
                courseunitcode VARCHAR(16),
                courseunittitle VARCHAR(64),
                courseunitduration ENUM('1S','2S','3S','4S','1T','2T','3T','4T','5T','6T'),
                ectscredits TINYINT,
                PRIMARY KEY (courseunitcode)
);
INSERT INTO courseunitdata VALUES ('341-SPO-87653','Leichter Lauf','1S','8');
INSERT INTO courseunitdata VALUES ('873-SPO-9735','Elfmeterschiessen','2S','6');
INSERT INTO courseunitdata VALUES ('893-SWI-2243','Revolutionslehre','2T','4');
INSERT INTO courseunitdata VALUES ('27-SWI-2454','Durch eine Labyrinth von 2 hoch 30 Reformmöglichkeiten','3T','8');
INSERT INTO courseunitdata VALUES ('34-INFO-29775','Internet-Systeme und Web-Applikationen','1S','6');
```

Figure 40: MySQL Database Schema for the CourseUnitDB

```
CREATE DATABASE organisationdb;
USE organisationdb;
CREATE TABLE universitydata (
		universityname VARCHAR(64),
		city VARCHAR(64),
		codepostale INT,
		state VARCHAR(64),
		PRIMARY KEY (universityname)
);
CREATE TABLE facultydata (
		universityname VARCHAR(64),
		facultyname VARCHAR(64),
		ectscoordinator_givenname VARCHAR(64),
		ectscoordinator_surname VARCHAR(64),
		ectscoordinator_tel BIGINT,
		ectscoordinator_fax BIGINT,
		ectscoordinator_email VARCHAR(64),
		PRIMARY KEY (universityname, facultyname)
);
INSERT INTO universitydata VALUES ('Universität Karlsruhe (TH)','Karlsruhe','76128','Deutschland');
INSERT INTO universitydata VALUES ('Università di Bologna','Bologna','40126','IT');
INSERT INTO facultydata VALUES ('Universität Karlsruhe (TH)',
		'Informatik','Max','Mustermann','07218456324865','07218456324865','mustermann@ira.uka.de');
```

Figure 41: MySQL Database Schema for the OrganisationDB

They are also available at $Archiv\usoa\tor\dbs.
All four services have together a total amount of about 3000 LOC which does not include the files that contains the MySQL schemas. The source code for the core Web services can be found in the subfolder of $ARCHIV\usoa\tor\java\src.

3.3 The ToRClient

As mentioned in many places when creating this SOA Part the surrounding USOA did not exist. This means that of course the USOA Portal did not exist too. Now the problem occurs that a potential user needs some GUIs to interact with the ToRService and these GUIs have to be embedded in the SOA Portal. So the decision was taken that a standalone application has to be realized to bypass the time period until the ToRService can be integrated into the USAO. Its realization was only done in a prototypic manner. The ToRClient helps the user to send his ToR request to the ToRService. The ToRClient consists of two components. One component encapsulates the whole business functionality which means that it transforms the user request to a SOAP request and translates the arriving SOAP response to an output at the GUI. The other component is called ToRClientGUI and it is responsible for the control of the necessary GUIs. As in the next chapter displayed the GUI consists of three section: a section at the top for managing the persistent storage of the XML ToRs and for entering the request parameters, a section in the middle to output process information and a bottom section with buttons to start the execution.

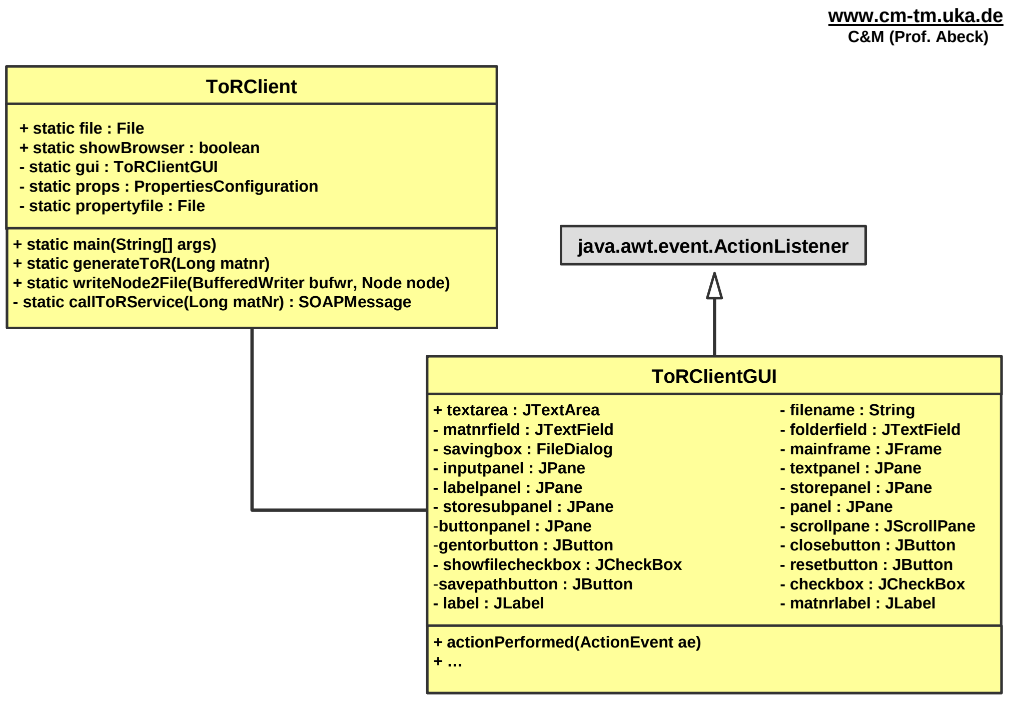

www.cm-tm.uka.de
C&M (Prof. Abeck)
ToRClient
+ static file : File
+ static showBrowser : boolean
- static gui : ToRClientGUI
- static props : PropertiesConfiguration
- static propertyfile : File
+ static main(String[] args)
+ static generateToR(Long matnr)
+ static writeNode2File(BufferedWriter bufwr, Node node)
- static callToRService(Long matNr) : SOAPMessage
java.awt.event.ActionListener
ToRClientGUI
+ textarea : JTextArea
- matnrfield : JTextField
- savingbox : FileDialog
- inputpanel : JPane
- labelpanel : JPane
- storesubpanel : JPane
-buttonpanel : JPane
-gentorbutton : JButton
- showfilecheckbox : JCheckBox
-savepathbutton : JButton
- label : JLabel
- filename : String
- folderfield : JTextField
- mainframe : JFrame
- textpanel : JPane
- storepanel : JPane
- panel : JPane
- scrollpane : JScrollPane
- closebutton : JButton
- resetbutton : JButton
- checkbox : JCheckBox
- matnrlabel : JLabel
+ actionPerformed(ActionEvent ae)
+ ...

Figure 42: Class Model for the ToRClient

The usage and appearance of the ToRClient is described in chapter 4.4.

3.4 Conducted Tests and Benchmarks

With the help of the Oracle Process Manager a stress test can be performed to test the behaviour of a deployed service under definable circumstances. This was also done for the ToRService.
This subchapter does not contain extensive test of the ToRService. It should rather give an idea of the order of magnitude of the ToRService's response time, because most parts of the application software were not tested in such a scenario.
The test was performed with two configurations to showcase how the BPEL process reacts under certain conditions. Both Tests were performed on the same host system.

Information about the host system	
Operating System	Microsoft Windows XP Professional SP2
Software	Oracle Process Manager; Apache Jarkata Tomcat; Apache Axis; MySQL Database Server
Processor	Intel Pentium 4; 3 GHz; 16 KByte L1 Cache; 1 MByte L2 Cache
Main Memory	1GByte main memory; Dual Channel; 400 MHz FSB
Secondary Storage	7200 rpm; SATA; 60 GBytes
Installation	All core Web services as well as the ToRService and the MySQL databases are running on the same machine

3.4.1 Test Configuration 1

The first configuration should benchmark how the ToRService manages low loads.

Test Parameters	
Number of concurrent threads	5
Number of loops per thread	100
Total amount of invocations	500
Constant delay between each invocation	10 ms

Test Results	
Average execution time per invocation	1247.64 ms
Rate	3.79 instructions/second
Minimum of execution time per thread	688 ms
Maximum of execution time per thread	2203 ms

3.4.2 Test Configuration 2

The second configuration benchmarks how the ToRService manages high loads.

Test Parameters	
Number of concurrent threads	30
Number of loops per thread	100
Total amount of invocations	3000
Constant delay between each invocation	10 ms

Test Results	
Average execution time per invocation	6318.98 ms
Rate	4.6 instructions/second
Minimum of execution time per thread	2844 ms
Maximum of execution time per thread	17531 ms

In both test cases the measured results allude to a configuration where the ToRService as well as the core Web services run on the same machine. The time it takes to forward a request from the ToRClient to the ToRService and back is also not considered. But despite of these facts, it can be asserted that the ToRService fulfills all needs of such a Web service in a USOA as regards response time.
Of course further tests are necessary to determinate if this service is scalable or not and how it deals with the system resources.

4 DEPLOYMENT AND USE

This chapter describes how to install the ToRService on a system with Microsoft Windows XP Home Edition / Professional and how it can be accessed via the ToRClient.

4.1 Software Requirements

To run the ToRService and the ToRClient the following software has to be installed:

- Java 2 Platform Standard Edition SDK (Version 1.4.2)
- Apache Jakarta Tomcat (Version 5.0.28)
- Apache Axis (Version 1.1)
- MySQL Database Server (Version 4.1.9) and MySQL Connector/J (Version 3.0.16)
- Process Manager from Oracle (Version 2.1.1)

4.2 Installing the Core Web Services

This subchapter contains a step-by-step description that guides potential users as well as system administrators through the installation process of the ToRService.

Let's start with the necessary databases. After downloading the MySQL Database Server at [MYSQL] it can be unpacked to the desired installation directory. In this document the installation path is C:\mysql. Changing to the folder C:\mysql\bin and executing the command mysqld --console causes the database server to start. The following command creates the four databases of chapter 2.3 and fills in some test values: mysql < $ARCHIV\usoa\tor\dbs\create_dbs.sql.

Now the Java 2 Platform Standard Edition SDK has to be downloaded from [SUN JAVA] and installed. With the aid of the installer the J2SE SDK can easily be installed. As installation folder in this document C:\java\j2sdk1.4.2_07\ is used.

The Apache Jakarta Tomcat installer can be downloaded under [APACHE TOMCAT]. The installation path C:\tomcat is used in this document. The installation can be validated by entering http://localhost:8080 into a browser. Of course the corresponding service to Tomcat has to be started. Because Tomcat needs to process JSPs some JVM tools either the JAVA_HOME variable has to be set correctly or the package tools.jar has to be placed in C:\tomcat\common\lib. Additionally the jars at $ARCHIV\usoa\tor\java\libs should be placed in C:\tomcat\common\lib.

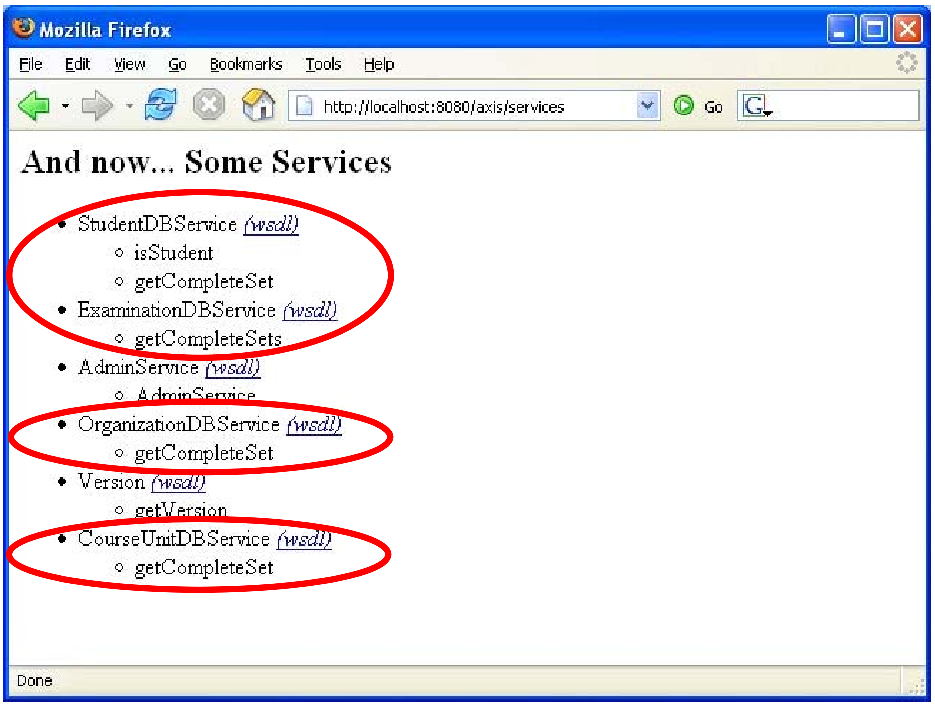
After downloading Apache Axis under [APACHE AXIS] it can be deployed using the installation description of the respective version. The folder webapps\axis of the Axis distribution has to be placed to C:\tomcat\webapps\. The packages jaxrpc.jar and saaj.jar has to be copied from C:\tomcat\webapps\axis\WEB-INF\lib to C:\tomcat\common\lib and after that Tomcat has to be restarted. This installation can also be validated by entering http://localhost:8080/axis/happyaxis.jsp into a browser. If some packages are missing Axis shows on this page where they can be found. Last but not least the value of the parameter sendMultiRefs has to be set to false in the configuration file C:\tomcat\webapps\axis\WEB-INF\server-config.wsdd to enable a good communication with the BPEL engine.

Finish the installation of the core Web services by deploying them to Axis. A detailed description how to deploy Web services to Axis can be found under [EJWS] or [EAA]. The Web services are placed in $ARCHIV\usoa\tor\java\classes\de\cm\services. With the corresponding deployment / undeployment descriptors deploy.wsdd / undeploy.wsdd in the core Web services can be easily deployed into Axis. The content of the folder $ARCHIV\usoa\tor\java\classes has to be removed to a special place in the working directory of Axis. This is done by copying the content to C:\tomcat\webapps\axis\WEB-INF\classes. Concluding the deploy.wsdd for each service has to be executed in the directory C:\tomcat\webapps\axis\WEB-INF\classes by typing the following commands into a shell:

```
set AXIS_HOME=C:\tomcat\webapps\axis

set AXIS_LIB=%AXIS_HOME%\WEB-INF\lib

set AXISCLASSPATH=%AXIS_LIB%\axis.jar;%AXIS_LIB%\commons-
discovery.jar;%AXIS_LIB%\commons-
logging.jar;%AXIS_LIB%\jaxrpc.jar;%AXIS_LIB%\saaj.jar;%AXIS_LIB%\log4j-
1.2.8.jar;%AXIS_LIB%\xml-apis.jar;%AXIS_LIB%\xercesImpl.jar

java -cp "%AXISCLASSPATH%" org.apache.axis.client.AdminClient -
lhttp://localhost:8080/axis/services/AdminService deploy.wsdd
```

Now the four core Web services should be seen when typing http://localhost:8080/axis/services into your browser.

www.cm-tm.uka.de
C&M (Prof. Abeck)


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/axis/services
Go
Done

And now... Some Services

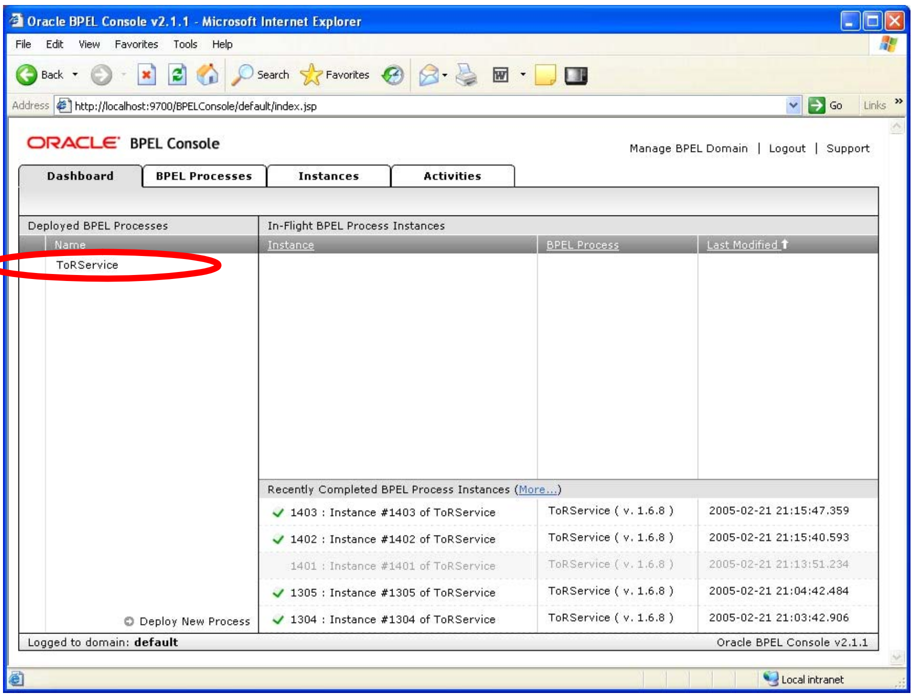
- StudentDBService (wsdl)
  - isStudent
  - getCompleteSet
- ExaminationDBService (wsdl)
  - getCompleteSets
- AdminService (wsdl)
  - AdminService
- OrganizationDBService (wsdl)
  - getCompleteSet
- Version (wsdl)
  - getVersion
- CourseUnitDBService (wsdl)
  - getCompleteSet

Figure 43: USE - The deployed core Web services

The service URLs of the basic configuration are http://localhost:8080/axis/services/StudentDBServiceP, http://localhost:8080/axis/services/ExaminationDBServiceP, http://localhost:8080/axis/services/CourseUnitDBServiceP and http://localhost:8080/axis/services/CourseUnitDBServiceP.
To start the RemoteClient and test the core Web services the batch file $ARCHIV\usoa\tor\RunRemoteClient.bat has to be executed into a shell and in the property file at $ARCHIV\usoa\tor\java\src\de\cm\clients the service URLs of the core Web services have to be set correctly. Furthermore in this property file all the request parameters can be adjusted.

4.3 Installing the ToRService

Now that the core Web services are running the installation of the ToRService can begin.
After downloading the Oracle Process Manager at [ORACLE] the installation can be executed using the installer program. In this document the installation path is C:\orabpel. Go to the Start Menu and click on Programs>Oracle BPEL Process Manager>Start BPEL PM to start the Process Manager. The installation can be validated by connecting to the BPELConsole. This is done by entering http://localhost:9700/BPELConsole into the Internet Explorer. A login is done with the default password bpel. In the BPELConsole a click at Deploy new Process and the selection of the ToRService at $ARCHIV\usoa\tor\bpel\bpel_ToRService_1.6.8.jar deploys the ToRService.


Oracle BPEL Console v2.1.1 - Microsoft Internet Explorer

Address http://localhost:9700/BPELConsole/default/index.jsp

ORACLE BPEL Console — Manage BPEL Domain | Logout | Support

Dashboard | BPEL Processes | Instances | Activities

Deployed BPEL Processes

Name
ToRService

Deploy New Process

In-Flight BPEL Process Instances

Instance	BPEL Process	Last Modified

Recently Completed BPEL Process Instances (More...)

✓ 1403 : Instance #1403 of ToRService	ToRService (v. 1.6.8)	2005-02-21 21:15:47.359
✓ 1402 : Instance #1402 of ToRService	ToRService (v. 1.6.8)	2005-02-21 21:15:40.593
1401 : Instance #1401 of ToRService	ToRService (v. 1.6.8)	2005-02-21 21:13:51.234
✓ 1305 : Instance #1305 of ToRService	ToRService (v. 1.6.8)	2005-02-21 21:04:42.484
✓ 1304 : Instance #1304 of ToRService	ToRService (v. 1.6.8)	2005-02-21 21:03:42.906

Logged to domain: **default** — Oracle BPEL Console v2.1.1

Figure 44: The deployed ToRService in the Oracle Process Manager

The service URL is http://localhost:9700/orabpel/default/ToRService.
Concluding it should be mentioned that of course the ToRService can be run on any other BPEL engine too. But this requires adaptations in the package structure and respectively in the deployment descriptor.

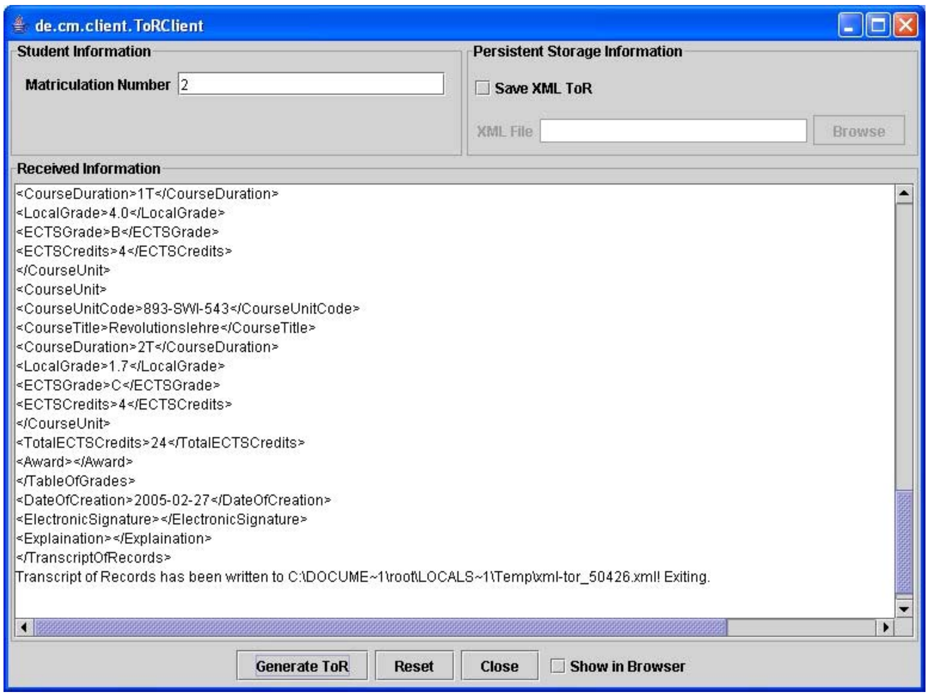
4.4 Using the ToRService

In Figure 45 the graphical user interface of the ToRClient is displayed. As shown it is divided into three sections:

- the head section where the input information can be entered
- the body section where all information that was received from the ToRService is printed out
- the foot section where all the available buttons are

The only necessary information is the matriculation number of the student. Despite of this information a file name can be entered to generate the received ToR in this file.

The big text area in the middle of the GUI displays all status information of the ToRService and the ToRClient and is needed for communication with the customer. At the bottom the user can close or reset the ToRClient as well as start the generation process of the ToR. Furthermore there exists a checkbox. If this checkbox is marked the received ToR is opened in the browser.



de.cm.client.ToRClient

Student Information

Matriculation Number: 2

Persistent Storage Information

☐ Save XML ToR

XML File:

Received Information

```
<CourseDuration>1T</CourseDuration>
<LocalGrade>4.0</LocalGrade>
<ECTSGrade>B</ECTSGrade>
<ECTSCredits>4</ECTSCredits>
</CourseUnit>
<CourseUnit>
<CourseUnitCode>893-SWI-543</CourseUnitCode>
<CourseTitle>Revolutionslehre</CourseTitle>
<CourseDuration>2T</CourseDuration>
<LocalGrade>1.7</LocalGrade>
<ECTSGrade>C</ECTSGrade>
<ECTSCredits>4</ECTSCredits>
</CourseUnit>
<TotalECTSCredits>24</TotalECTSCredits>
<Award></Award>
</TableOfGrades>
<DateOfCreation>2005-02-27</DateOfCreation>
<ElectronicSignature></ElectronicSignature>
<Explaination></Explaination>
</TranscriptOfRecords>
Transcript of Records has been written to C:\DOCUME~1\root\LOCALS~1\Temp\xml-tor_50426.xml! Exiting.
```

Figure 45: The GUI of the ToRClient

To start the ToRClient the file $ARCHIV\usoa\tor\RunToRClient has to be double clicked and in the property file at $ARCHIV\usoa\tor\java\src\de\cm\clients the service URL of the ToRService has to be set correctly.

APPENDIX A XML schema for the ToR

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ToR.xsd -->
<!-- @author weisser@cm-tm.uka.de -->

<xsd:schema targetNamespace="urn:schema.tor.services.cm.de"
                      xmlns:tns="urn:schema.tor.services.cm.de"
                      xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!-- Transcript of Records -->
<xsd:complexType name="ToR">
   <xsd:sequence>
      <xsd:element name="TranscriptOfRecords" type="tns:ToRStructure"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Structure of the Transcript of Records -->
<xsd:complexType name="ToRStructure">
   <xsd:sequence>
      <xsd:element name="Header" type="tns:Head"/>
      <xsd:element name="TableOfGrades" type="tns:Table"/>
      <xsd:element name="DateOfCreation" type="xsd:dateTime"/>
      <xsd:element name="ElectronicSignature" type="xsd:string"/>
      <xsd:element name="Explaination" type="tns:Foot"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Back of the ToR -->
<xsd:complexType name="Foot">
   <xsd:sequence>
      <xsd:element name="InstitutionalGradingSystem" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Header -->
<xsd:complexType name="Head">
   <xsd:sequence>
      <xsd:element name="SendingInstitution" type="tns:Organization"/>
      <xsd:element name="Student" type="tns:Person"/>
      <xsd:element name="ReceivingInstitution" type="tns:Organization"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Personal Data of Header -->
<xsd:complexType name="Person">
   <xsd:sequence>
      <xsd:element name="Surname" type="xsd:string"/>
      <xsd:element name="GivenName" type="xsd:string"/>
      <xsd:element name="DateOfBirth" type="xsd:dateTime"/>
      <xsd:element name="PlaceOfBirth" type="xsd:string"/>
      <xsd:element name="Sex" type="xsd:boolean"/>
      <xsd:element name="DateOfMatriculation" type="xsd:dateTime"/>
      <xsd:element name="MatriculationNumber" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Instiotutional Data of Header -->
<xsd:complexType name="Organization">
   <xsd:sequence>
      <xsd:element name="InstitutionName" type="xsd:string"/>
      <xsd:element name="FacultyDepartmentName" type="xsd:string"/>
      <xsd:element name="ECTSDepartmentalCoordinator">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Surname" type="xsd:string"/>
               <xsd:element name="GivenName" type="xsd:string"/>
               <xsd:element name="Telephone" type="xsd:string"/>
               <xsd:element name="Fax" type="xsd:string"/>
               <xsd:element name="EMail" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<!-- TableOfGrades -->
<xsd:complexType name="Table">
   <xsd:sequence>
      <xsd:element name="CourseUnit" maxOccurs="unbounded" type="tns:Row"/>
      <xsd:element name="TotalECTSCredits" type="xsd:int"/>
      <xsd:element name="Award" maxOccurs="unbounded" type="xsd:string"/>
   </xsd:sequence>
```

```
</xsd:complexType>

<!-- Row of TableOfGrades -->
<xsd:complexType name="Row">
   <xsd:sequence>
      <xsd:element name="CourseUnitCode" type="xsd:string"/>
      <xsd:element name="CourseTitle" type="xsd:string"/>
      <xsd:element name="CourseDuration" type="xsd:string"/>
      <xsd:element name="LocalGrade" type="xsd:string"/>
      <xsd:element name="ECTSGrade" type="xsd:string"/>
      <xsd:element name="ECTSCredits" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

APPENDIX B WSDL of a core Web service as a Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- StudentDBService.wsdl -->
<!-- @author weisser@cm-tm.uka.de -->

<definitions name="urn:StudentDBService"
         targetNamespace="urn:stud.services.cm.de" xmlns:tns="urn:stud.services.cm.de"
         xmlns:typns="urn:stud.services.cm.de"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns="http://schemas.xmlsoap.org/wsdl/">

<!-- type definitions -->
<types>
   <xsd:schema targetNamespace="urn:stud.services.cm.de"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="SetOfStandingTable">
         <xsd:all>
            <xsd:element name="matriculationNumber" type="xsd:long"/>
            <xsd:element name="surName" type="xsd:string"/>
            <xsd:element name="givenName" type="xsd:string"/>
            <xsd:element name="sex" type="xsd:boolean"/>
            <xsd:element name="placeOfBirth" type="xsd:string"/>
            <xsd:element name="dateOfBirth" type="xsd:dateTime"/>
            <xsd:element name="nationality" type="xsd:string"/>
            <xsd:element name="street" type="xsd:string"/>
            <xsd:element name="houseNumber" type="xsd:int"/>
            <xsd:element name="codePostale" type="xsd:int"/>
            <xsd:element name="city" type="xsd:string"/>
         </xsd:all>
      </xsd:complexType>
      <xsd:complexType name="SetOfStudyTable">
         <xsd:all>
            <xsd:element name="matriculationNumber" type="xsd:long"/>
            <xsd:element name="dateOfMatriculation" type="xsd:dateTime"/>
            <xsd:element name="fieldOfStudy" type="xsd:string"/>
            <xsd:element name="award" type="xsd:string"/>
         </xsd:all>
      </xsd:complexType>
      <xsd:complexType name="SetOfStudentDB">
         <xsd:all>
            <xsd:element name="standingData" type="typns:SetOfStandingTable"/>
            <xsd:element name="studyData" type="typns:SetOfStudyTable"/>
         </xsd:all>
      </xsd:complexType>
   </xsd:schema>
</types>

<!-- message declarations -->
<message name="IsStudentRequestMessage">
   <part name="matriculationNumber" type="xsd:long"/>
</message>
<message name="IsStudentResponseMessage">
   <part name="validFlag" type="xsd:boolean"/>
</message>
<message name="GetCompleteSetRequestMessage">
   <part name="matriculationNumber" type="xsd:long"/>
</message>
<message name="GetCompleteSetResponseMessage">
   <part name="studentSet" type="typns:SetOfStudentDB"/>
</message>

<!-- port type declarations -->
<portType name="StudentDBServicePT">
   <operation name="isStudent">
      <input message="tns:IsStudentRequestMessage"/>
      <output message="tns:IsStudentResponseMessage"/>
   </operation>
   <operation name="getCompleteSet">
      <input message="tns:GetCompleteSetRequestMessage"/>
      <output message="tns:GetCompleteSetResponseMessage"/>
   </operation>
</portType>

<!-- binding declarations -->
<binding name="StudentDBServiceSoapBinding" type="tns:StudentDBServicePT">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="isStudent" >
      <soap:operation soapAction=""/>
      <input>
```

```
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="urn:stud.services.cm.de" use="encoded"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="urn:stud.services.cm.de" use="encoded"/>
      </output>
   </operation>
   <operation name="getCompleteSet">
      <soap:operation soapAction=""/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="urn:stud.services.cm.de" use="encoded"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="urn:stud.services.cm.de" use="encoded"/>
      </output>
   </operation>
</binding>

<!-- service declaration -->
<service name="StudentDBService">
   <port name="StudentDBServiceP" binding="tns:StudentDBServiceSoapBinding">
      <soap:address location="http://localhost:8080/axis/services/StudentDBService"/>
   </port>
</service>

</definitions>
```

APPENDIX C WSDL of the ToRService

```
<?xml version="1.0"?>
<!-- ToRService.wsdl -->
<!-- @author weisser@cm-tm.uka.de -->

<definitions name="ToRService" targetNamespace="urn:tor.services.cm.de"
         xmlns:tns="urn:tor.services.cm.de"
         xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
         xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
   <schema attributeFormDefault="qualified" elementFormDefault="qualified"
         targetNamespace="urn:tor.services.cm.de"
         xmlns="http://www.w3.org/2001/XMLSchema"
         xmlns:ts="urn:schema.tor.services.cm.de">
      <import namespace="urn:schema.tor.services.cm.de" schemaLocation="ToRSchema.xsd"/>
      <element name="ToRServiceRequest">
         <complexType>
            <sequence>
               <element name="matriculationNumber" type="long"/>
            </sequence>
         </complexType>
      </element>
      <element name="Report">
         <complexType>
            <sequence>
               <element name="StatusReport">
                  <complexType>
                     <sequence>
                        <element name="validStudent" type="boolean"/>
                        <element name="error" type="string"/>
                     </sequence>
                  </complexType>
               </element>
               <element name="TempVars">
                  <complexType>
                     <sequence>
                        <element name="examCount" type="int"/>
                        <element name="counter" type="int"/>
                     </sequence>
                  </complexType>
               </element>
            </sequence>
         </complexType>
      </element>
      <element name="GeneratedToR" type="ts:ToR"/>
      <element name="RawInformation">
         <complexType>
            <sequence>
               <element name="Organization"/>
               <element name="Student"/>
               <element name="TableOfGrades"/>
               <element name="totalECTSCredits" type="int"/>
               <element name="creationDate" type="dateTime"/>
            </sequence>
         </complexType>
      </element>
   </schema>
</types>

<message name="ToRServiceRequestMessage">
   <part name="request" element="tns:ToRServiceRequest"/>
</message>
<message name="ToRServiceResponseMessage">
   <part name="payload" element="tns:GeneratedToR"/>
   <part name="attachment" element="tns:Report"/>
</message>

<portType name="ToRService">
   <operation name="process">
      <input  message="tns:ToRServiceRequestMessage" />
      <output message="tns:ToRServiceResponseMessage"/>
   </operation>
</portType>

<plnk:partnerLinkType name="ToRService">
   <plnk:role name="ToRServiceProvider">
      <plnk:portType name="tns:ToRService"/>
   </plnk:role>
</plnk:partnerLinkType>

</definitions>
```

APPENDIX D   BPEL Code of the ToRService

```
<!-- ToRService.bpel -->
<!-- @author weisser@cm-tm.uka.de -->

<process name="ToRService" targetNamespace="urn:tor.services.cm.de"
         suppressJoinFailure="yes" xmlns:tns="urn:tor.services.cm.de"
         xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
         xmlns:ora="http://schemas.oracle.com/xpath/extension"
         xmlns:bpws=http://schemas.xmlsoap.org/ws/2003/03/business-process/
          xmlns:ns0="urn:stud.services.cm.de" xmlns:ns1="urn:org.services.cm.de"
         xmlns:ns2="urn:exam.services.cm.de" xmlns:ns3="urn:course.services.cm.de"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:ns4="http://schemas.xmlsoap.org/soap/encoding/">

<partnerLinks>
<partnerLink name="client" partnerLinkType="tns:ToRService"
         myRole="ToRServiceProvider"/>
<partnerLink name="StudentDBServicePL" partnerLinkType="ns0:StudentDBServicePLT"
         partnerRole="StudentDBServicePTProvider"/>
<partnerLink name="OrganizationDBServicePL"
         partnerLinkType="ns1:OrganizationDBServicePLT"
         partnerRole="OrganizationDBServicePTProvider"/>
<partnerLink name="ExaminationDBServicePL" partnerLinkType="ns2:ExaminationDBServicePLT"
         partnerRole="ExaminationDBServicePTProvider"/>
<partnerLink name="CourseUnitDBServicePL" partnerLinkType="ns3:CourseUnitDBServicePLT"
         partnerRole="CourseUnitDBServicePTProvider"/>
</partnerLinks>

<variables>
<variable name="InputVariable" messageType="tns:ToRServiceRequestMessage"/>
<variable name="OutputVariable" messageType="tns:ToRServiceResponseMessage"/>
</variables>

<faultHandlers>
<catchAll>
<assign name="WriteFaultToGloalVariable">
<copy>
   <from expression="'Unknown Fault occured while trying to execute the ToRService'">
   </from>
   <to variable="OutputVariable" part="attachment"
            query="/tns:Report/tns:StatusReport/tns:error"/>
</copy>
</assign>
</catchAll>
</faultHandlers>

<sequence name="main">

<receive name="receiveInput" partnerLink="client" portType="tns:ToRService"
         operation="process" variable="InputVariable" createInstance="yes"/>

<scope name="CheckPlausibility">
<variables>
   <variable name="IsStudentRequestVariable" messageType="ns0:isStudentRequest"/>
   <variable name="IsStudentResponseVariable" messageType="ns0:isStudentResponse"/>
</variables>
<faultHandlers>
   <catchAll>
      <sequence>
      <assign xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
               name="WriteFaultToGloalVariable">
         <copy>
            <from expression="'Fault occured while trying to execute the
                     plausibility check via the StudentDBServicePL'"></from>
            <to variable="OutputVariable" part="attachment"
                     query="/tns:Report/tns:StatusReport/tns:error"/>
         </copy>
      </assign>
      </sequence>
   </catchAll>
</faultHandlers>
<sequence name="sequence-1">
<assign name="InitializeLocalVariables">
   <copy>
      <from variable="InputVariable" part="request"
               query="/tns:ToRServiceRequest/tns:matriculationNumber">
      </from>
      <to variable="IsStudentRequestVariable" part="matriculationNumber"/>
   </copy>
   <copy>
      <from expression="'false'"></from>
      <to variable="OutputVariable" part="attachment"
               query="/tns:Report/tns:StatusReport/tns:validStudent"/>
   </copy>
   <copy>
      <from expression="'none'"></from>
      <to variable="OutputVariable" part="attachment"
               query="/tns:Report/tns:StatusReport/tns:error"/>
   </copy>
</assign>
<invoke name="LookupStudent" partnerLink="StudentDBServicePL"
         portType="ns0:StudentDBServicePT" operation="isStudent"
         inputVariable="IsStudentRequestVariable"
         outputVariable="IsStudentResponseVariable"/>
<assign name="WriteResultsToGlobalVariables">
   <copy>
      <from variable="IsStudentResponseVariable" part="validFlag"></from>
      <to variable="OutputVariable" part="attachment"
               query="/tns:Report/tns:StatusReport/tns:validStudent"/>
   </copy>
</assign>
</sequence>
</scope>

<switch name="IsValidStudent">
<case condition="bpws:getVariableData('OutputVariable','attachment',
         '/tns:Report/tns:StatusReport/tns:validStudent') = 'true'">
<scope name="ValidStudentBranch">
<variables>
   <variable name="SDBSGetCompleteSetRequestVariable"
            messageType="ns0:getCompleteSetRequest"/>
   <variable name="SDBSGetCompleteSetResponseVariable"
            messageType="ns0:getCompleteSetResponse"/>
   <variable name="ODBSGetCompleteSetRequestVariables"
            messageType="ns1:getCompleteSetRequest"/>
   <variable name="ODBSGetCompleteSetResponseVariables"
            messageType="ns1:getCompleteSetResponse"/>
   <variable name="EDBSGetCompleteSetsRequestVariable"
            messageType="ns2:getCompleteSetsRequest"/>
   <variable name="EDBSGetCompleteSetsResponseVariable"
            messageType="ns2:getCompleteSetsResponse"/>
   <variable name="CDBSGetCompleteSetRequestVariable"
            messageType="ns3:getCompleteSetRequest"/>
   <variable name="CDBSGetCompleteSetResponseVariable"
            messageType="ns3:getCompleteSetResponse"/>
   <variable name="RawInformationVariable" element="tns:RawInformation"/>
</variables>
<faultHandlers>
   <catchAll>
      <sequence>
         <assign xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
                  process/" name="WriteFaultToGloalVariable">
            <copy>
               <from expression="'Fault occured while trying to collect the
                        necessary information from the involved core web services'">
               </from>
               <to variable="OutputVariable" part="attachment"
                        query="/tns:Report/tns:StatusReport/tns:error"/>
            </copy>
         </assign>
      </sequence>
   </catchAll>
</faultHandlers>
<sequence name="sequence-2">
<sequence name="sequence-3">
<assign name="InitializeLocalVariables">
   <copy>
      <from variable="InputVariable" part="request"
               query="/tns:ToRServiceRequest/tns:matriculationNumber">
      </from>
      <to variable="SDBSGetCompleteSetRequestVariable" part="matriculationNumber"/>
   </copy>
   <copy>
      <from expression="ora:getCurrentDate(&quot;yyyy-MM-dd&quot;)"></from>
      <to variable="RawInformationVariable"
               query="/tns:RawInformation/tns:creationDate"/>
   </copy>
   <copy>
      <from variable="InputVariable" part="request"
               query="/tns:ToRServiceRequest/tns:matriculationNumber"></from>
      <to variable="EDBSGetCompleteSetsRequestVariable" part="matriculationNumber"/>
   </copy>
</assign>
<flow name="CollectNecessaryInformation">
   <sequence name="flow-sequence-1">
      <invoke name="CollectStudentInformation" partnerLink="StudentDBServicePL"
```

```
                portType="ns0:StudentDBServicePT" operation="getCompleteSet"
                inputVariable="SDBSGetCompleteSetRequestVariable"
                outputVariable="SDBSGetCompleteSetResponseVariable"/>
         <assign name="PassVariables">
            <copy>
               <from variable="SDBSGetCompleteSetResponseVariable" part="studentSet"
                      query="/studentSet/studyData/fieldOfStudy"></from>
               <to variable="ODBSGetCompleteSetRequestVariables" part="facultyName"/>
            </copy>
         </assign>
         <invoke name="CollectOrganizationInformation"
                partnerLink="OrganizationDBServicePL"
                portType="ns1:OrganizationDBServicePT" operation="getCompleteSet"
                inputVariable="ODBSGetCompleteSetRequestVariables"
                outputVariable="ODBSGetCompleteSetResponseVariables"/>
         <assign name="WriteResultsToGlabalVariable">
            <copy>
               <from variable="SDBSGetCompleteSetResponseVariable" part="studentSet">
               </from>
               <to variable="RawInformationVariable"
                      query="/tns:RawInformation/tns:Student"/>
            </copy>
            <copy>
               <from variable="ODBSGetCompleteSetResponseVariables" part="organizationSet">
               </from>
               <to variable="RawInformationVariable"
                      query="/tns:RawInformation/tns:Organization"/>
            </copy>
         </assign>
      </sequence>
      <sequence name="flow-sequence-2">
         <invoke name="CollectExaminationInformation" partnerLink="ExaminationDBServicePL"
                portType="ns2:ExaminationDBServicePT" operation="getCompleteSets"
                inputVariable="EDBSGetCompleteSetsRequestVariable"
                outputVariable="EDBSGetCompleteSetsResponseVariable"/>
         <assign name="PassVariables">
            <copy>
               <from expression="ora:countNodes('EDBSGetCompleteSetsResponseVariable',
                      'examinationSets', '/examinationSets/*')"></from>
               <to variable="OutputVariable" part="attachment"
                      query="/tns:Report/tns:TempVars/tns:examCount"/>
            </copy>
            <copy>
               <from expression="1"></from>
               <to variable="OutputVariable" part="attachment"
                      query="/tns:Report/tns:TempVars/tns:counter"/>
            </copy>
            <copy>
               <from expression="0"></from>
               <to variable="RawInformationVariable"
                      query="/tns:RawInformation/tns:totalECTSCredits"/>
            </copy>
         </assign>
         <while name="LoopForAllExaminations" condition="bpws:getVariableData(
                'OutputVariable','attachment','/tns:Report/tns:TempVars/tns:counter')
                &lt;= bpws:getVariableData('OutputVariable','attachment',
                '/tns:Report/tns:TempVars/tns:examCount')">
            <sequence name="sequence-5">
               <assign name="UpdateLoopVariables">
                  <copy>
                     <from expression="ora:getElement(
                            'EDBSGetCompleteSetsResponseVariable', 'examinationSets',
                            '/examinationSets/Examination/courseUnitCode',
                            bpws:getVariableData('OutputVariable', 'attachment',
                            '/tns:Report/tns:TempVars/tns:counter'))"></from>
                     <to variable="CDBSGetCompleteSetRequestVariable"
                            part="courseUnitCode"/>
                  </copy>
               </assign>
               <invoke name="CollectCourseInformation" partnerLink="CourseUnitDBServicePL"
                      portType="ns3:CourseUnitDBServicePT" operation="getCompleteSet"
                      inputVariable="CDBSGetCompleteSetRequestVariable"
                      outputVariable="CDBSGetCompleteSetResponseVariable"/>
               <assign name="WriteResultsToGlobalVariable">
                  <copy>
                     <from expression="ora:addChildNode(bpws:getVariableData(
                            'CDBSGetCompleteSetResponseVariable','courseUnitSet',
                            '/courseUnitSet'),ora:getElement('EDBSGetCompleteSetsResponse
                            Variable', 'examinationSets', '/examinationSets/Examination',
                            bpws:getVariableData('OutputVariable', 'attachment',
                            '/tns:Report/tns:TempVars/tns:counter')))"></from>
                     <to variable="CDBSGetCompleteSetResponseVariable" part="courseUnitSet"
                            query="/courseUnitSet"/>
                  </copy>
                  <copy>
                  <from expression="ora:addChildNode(bpws:getVariableData(
                           'RawInformationVariable',
                           '/tns:RawInformation/tns:TableOfGrades'),
                           bpws:getVariableData('CDBSGetCompleteSetResponseVariable',
                           'courseUnitSet'))"></from>
                  <to variable="RawInformationVariable"
                           query="/tns:RawInformation/tns:TableOfGrades"/>
               </copy>
               <copy>
                  <from expression="bpws:getVariableData('OutputVariable',
                           'attachment','/tns:Report/tns:TempVars/tns:counter') + 1">
                  </from>
                  <to variable="OutputVariable" part="attachment"
                           query="/tns:Report/tns:TempVars/tns:counter"/>
               </copy>
               <copy>
                  <from expression="bpws:getVariableData('RawInformationVariable',
                           '/tns:RawInformation/tns:totalECTSCredits') +
                           bpws:getVariableData('CDBSGetCompleteSetResponseVariable',
                           'courseUnitSet','/courseUnitSet/ECTSCredits')"></from>
                  <to variable="RawInformationVariable"
                           query="/tns:RawInformation/tns:totalECTSCredits"/>
               </copy>
            </assign>
         </sequence>
      </while>
   </sequence>
</flow>
</sequence>
<assign xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         name="GenerateToR">
   <copy>
      <from expression="ora:processXSLT('ToRGenerator.xslt',
               bpws:getVariableData('RawInformationVariable','/tns:RawInformation'))">
      </from>
      <to variable="OutputVariable" part="payload"
               query="/tns:GeneratedToR/TranscriptOfRecords"/>
   </copy>
</assign>
</sequence>
</scope>
</case>

<otherwise>
<scope name="InvalidStudentBranch">
   <assign name="DoNothing">
      <copy>
         <from expression="ora:getCurrentDate(&quot;yyyy-MM-dd&quot;)"></from>
         <to variable="OutputVariable" part="payload"
                  query="/tns:GeneratedToR/TranscriptOfRecords/DateOfCreation"/>
      </copy>
      <copy>
         <from expression="-1"></from>
         <to variable="OutputVariable" part="payload" query=
               "/tns:GeneratedToR/TranscriptOfRecords/TableOfGrades/TotalECTSCredits"/>
      </copy>
      <copy>
         <from expression="-1"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/ECTSCredits"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/ECTSGrade"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/LocalGrade"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/CourseDuration"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
               /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/CourseTitle"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/TableOfGrades/CourseUnit[1]/CourseUnitCode"/>
      </copy>
      <copy>
         <from expression="-1"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/MatriculationNumber"/>
      </copy>
      <copy>
         <from expression="ora:getCurrentDate(&quot;yyyy-MM-dd&quot;)"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/DateOfMatriculation"/>
      </copy>
      <copy>
         <from expression="'false'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/Sex"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/PlaceOfBirth"/>
      </copy>
      <copy>
         <from expression="ora:getCurrentDate(&quot;yyyy-MM-dd&quot;)"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/DateOfBirth"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/GivenName"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/Student/Surname"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /ECTSDepartmentalCoordinator/EMail"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /ECTSDepartmentalCoordinator/Fax"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /ECTSDepartmentalCoordinator/Telephone"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /ECTSDepartmentalCoordinator/GivenName"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /ECTSDepartmentalCoordinator/Surname"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution
                  /FacultyDepartmentName"/>
      </copy>
      <copy>
         <from expression="'none'"></from>
         <to variable="OutputVariable" part="payload" query="/tns:GeneratedToR
                  /TranscriptOfRecords/Header/SendingInstitution/InstitutionName"/>
      </copy>
      <copy>
         <from expression="-1"></from>
         <to variable="OutputVariable" part="attachment"
                  query="/tns:Report/tns:TempVars/tns:counter"/>
      </copy>
      <copy>
         <from expression="-1"></from>
         <to variable="OutputVariable" part="attachment"
```

```
                  query="/tns:Report/tns:TempVars/tns:examCount"/>
      </copy>
   </assign>
</scope>
</otherwise>

</switch>
<reply name="replyOutput" partnerLink="client" portType="tns:ToRService"
         operation="process" variable="OutputVariable"/>
</sequence>

</process>
```

APPENDIX E XSLT Code for Generating the XML ToR

```
<?xml version="1.0" encoding="utf-8"?>
<!-- GeneratingToR.xslt -->
<!-- @author weisser@cm-tm.uka.de -->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
            xmlns:ts="urn:tor.services.cm.de">

<xsl:output method="xml" indent="yes"/>

<xsl:template match="ts:RawInformation">

<TranscriptOfRecords xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="ToRSchema.xsd">

<Header>

<SendingInstitution>
   <InstitutionName>
      <xsl:value-of select="ts:Organization/universityData/universityName"/>
   </InstitutionName>
   <FacultyDepartmentName>
      <xsl:value-of select="ts:Organization/facultyData/facultyName"/>
   </FacultyDepartmentName>
   <ECTSDepartmentalCoordinator>
      <Surname>
         <xsl:value-of select="ts:Organization/facultyData/ECTSCoordinatorSurName"/>
      </Surname>
      <GivenName>
         <xsl:value-of select="ts:Organization/facultyData/ECTSCoordinatorGivenName"/>
      </GivenName>
      <Telephone>
         <xsl:value-of select="ts:Organization/facultyData/ECTSCoordinatorTel"/>
      </Telephone>
      <Fax>
         <xsl:value-of select="ts:Organization/facultyData/ECTSCoordinatorFax"/>
      </Fax>
      <E-Mail>
         <xsl:value-of select="ts:Organization/facultyData/ECTSCoordinatorEmail"/>
      </E-Mail>
   </ECTSDepartmentalCoordinator>
</SendingInstitution>

<Student>
   <Surname>
      <xsl:value-of select="ts:Student/standingData/surName"/>
   </Surname>
   <GivenName>
      <xsl:value-of select="ts:Student/standingData/givenName"/>
   </GivenName>
   <DateOfBirth>
      <xsl:value-of select="ts:Student/standingData/dateOfBirth"/>
   </DateOfBirth>
   <PlaceOfBirth>
      <xsl:value-of select="ts:Student/standingData/placeOfBirth"/>
   </PlaceOfBirth>
   <Sex>
      <xsl:value-of select="ts:Student/standingData/sex"/>
   </Sex>
   <DateOfMatriculation>
      <xsl:value-of select="ts:Student/studyData/dateOfMatriculation"/>
   </DateOfMatriculation>
   <MatriculationNumber>
      <xsl:value-of select="ts:Student/studyData/matriculationNumber"/>
   </MatriculationNumber>
</Student>

<ReceivingInstitution>
   <xsl:comment/>
</ReceivingInstitution>

</Header>

<TableOfGrades>
   <xsl:for-each select="ts:TableOfGrades/courseUnitSet">
      <CourseUnit>
         <CourseUnitCode>
            <xsl:value-of select="courseUnitCode"/>
         </CourseUnitCode>
         <CourseTitle>
            <xsl:value-of select="courseUnitTitle"/>
         </CourseTitle>
         <CourseDuration>
            <xsl:value-of select="courseUnitDuration"/>
         </CourseDuration>
         <LocalGrade>
            <xsl:value-of select="Examination/localGrade"/>
         </LocalGrade>
         <ECTSGrade>
            <xsl:value-of select="Examination/ECTSGrade"/>
         </ECTSGrade>
         <ECTSCredits>
            <xsl:value-of select="ECTSCredits"/>
         </ECTSCredits>
      </CourseUnit>
   </xsl:for-each>
   <TotalECTSCredits>
      <xsl:value-of select="ts:totalECTSCredits"/>
   </TotalECTSCredits>
   <Award>
      <xsl:comment/>
   </Award>
</TableOfGrades>

<DateOfCreation>
   <xsl:value-of select="ts:creationDate"/>
</DateOfCreation>

<ElectronicSignature>
   <xsl:comment/>
</ElectronicSignature>

<Explaination>
   <xsl:comment/>
</Explaination>

</TranscriptOfRecords>

</xsl:template>

</xsl:stylesheet>
```

TABLES

Abbreviations and Glossary

Abbreviation or Term	Full Name and/or Term Description
BPEL	Business Process Execution Language In computer science, the Business Process Execution Language (BPEL) is an XML language to describe business processes. A BPEL program is invoked as a Web service, and it can interact with the external world only by calling Web services. The standard that defines how BPEL is used in Web service transactions is BPEL4WS also known as WS-BPEL. BPEL is designed by IBM and Microsoft, based on their respective work on WSFL and XLANG, which are both superseded by BPEL. In April 2003, BPEL was submitted to OASIS and is now being standardized in the Web Services BPEL Technical Committee.
BPMN	Business Process Modeling Language Important Specification in the context of Business Process Modeling (BPM) developed by the Business Process Management Initiative (BPMI).
C&M	Cooperation & Management Name of a research group at the University of Karlsruhe
J2SE	Java-2-Platform, Standard Edition Variant of the Java framework for the development of client applications.
JDBC	Java Database Connectivity Java API for accessing databases
JDK	Java Development Toolkit Set of design tools (interpreter, compiler) and class libraries
SOA	Service Oriented Architecture A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties: • Logical view: The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation. • Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

	• Description orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description. • Granularity: Services tend to use a small number of operations with relatively large and complex messages. • Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement. • Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint
SOAP	Simple Object Access Protocol XML-based miscellaneously useable application protocol
USOA	University SOA Name of the SOA which supports business processes of universities.
Web service	A Web service is a software system designated to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the (Web) service is the resource characterized by the abstract set of functionality that is provided.
WSDL	Web Service Description Language XML format published for describing Web service interfaces
XML	eXtensible Markup Language W3C recommendation for creating special-purpose markup languages; it is a simplified subset of SGML, capable of describing many different kinds of data
XSLT	XML Transformation XML markup language used for transforming XML documents

References

[AE+04] S. Abeck, C. Emig, J. Weisser: Fallstudie Transcript of Records, Bericht zum Projekt „Werkstatt Unternehmenssoftware Karlsruhe“ (WUSKAR), Karlsruhe 2004.

[APACHE] Official Homepage of the Apache Project http://www.apache.org

[BPMN1.0] Business Process Management Initiative (BPMI): Business Process Modeling Notation (BPMN), Version 1.0, BPMI.org, May 2004.

[BPEL1.1]	Business Process Execution Language for Web Services (BPEL4WS), Version 1.1, May 2003 http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
[EJWS]	J. Weisser: Einführung in Java-Webservices, C&M 2005
[EAA]	E. Rull: Einführung in Apache Axis, C&M 2005
[MYSQL]	Official Homepage of MySQL http://www.mysql.com
[OASIS]	Official Homepage of the Organization for the Advancement of Structured Information Standards (OASIS) http://www.oasis-open.org
[ORACLE]	Official Homepage of Oracle http://www.oracle.com
[OR03]	Martin Owen, Jog Ray: BPMN and Business Process Management – Introduction to the New Business Modeling Standard, Popkin Software 2003.
[SUN JAVA]	Official Homepage of Sun Microsystems http://www.sun.com
[W3CXML]	Official Homepage of the World Wide Web Consortium http://www.w3c.org