WERKSTATT UNTERNEHMENSSOFTWARE KARLSRUHE (WUSKAR)

CASE STUDY

INTEGRATION OF SAP CAMPUS MANAGEMENT INTO A UNIVERSITY SOA

Heiko Schandua
Tomas Stiller

Christian Emig
Sebastian Abeck

Cooperation & Management (C&M)
Institute for Telematics
Universität Karlsruhe (TH)
wuskar@cm-tm.uka.de

Abstract

This case study depicts the integration of a legacy university resource planning system (URP) into a Service-oriented Architecture (SOA). The idea is to add the functionality of generating so-called Bologna-conforming Transcript of Records to SAP Campus Management (SAP CM). This is done by adding Web service interfaces to SAP CM that are orchestrated in the SOA using the Business Process Execution Language (BPEL). User Interaction is handled via a central University Portal.

Keywords

SAP Campus Management (SAP CM), Service-oriented Architecture (SOA), WUSKAR, ABAP, SAP Java Connector (JCo), Transcript of Records (ToR), Web service, University SOA (USOA), ToR Service, XML, UML, BPMN, BPEL, Java, SOAP, WSDL, BAPI, RFM, Apache Jakarta Tomcat, Apache Axis, Oracle Process Manager

Learning Goals

1. Understand how to add functionality to a legacy university resource planning system, especially to SAP Campus Management (SAP CM).
2. Obtain an idea how to integrate functionality from a legacy URP system into a Service-oriented Architecture (SOA).
3. Obtain a basic knowledge about ABAP and the SAP Java Connector (JCo) as technologies needed for the integration process.
4. Learn about tools for the orchestration of Web services.
5. Understand how to integrate Business Process Management into a SOA-Portal.

Major Sources

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

[We05] Jochen Weisser: University SOA – Building a Transcript of Records Service, Studienarbeit, Universität Karlsruhe, 2005.

Table of Contents

0 INTRODUCTION

0.1 Motivation

Through the "Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)", students of the Universtität Karlsruhe (TH) should have the possibility to work with various actual software products within the scope of this case study.

This case study deals with the European Higher Education Area and the so-called Bologna-Process it initiated. In the past years political and economical events led to a change in the German university landscape. On the one hand this is concerned with the coalescence of Europe to a European Higher Education Area. With the introduction of a European Credit Transfer System (ECTS) students should be enabled to easily study a semester abroad or completely change universities without confronting problems with the accreditation of their achieved study results. On the other hand this change has to deal with the business situation of the last years causing an increasing cost pressure on the German university system. Possibilities of optimizing the actual business processes without loosing quality of teaching and research have to be found. Savings potentials therefore are, for example, located in the administration. SAP Campus Management (SAP CM) is a system that supports universities' managing of their students in a high quality but cost-effective way.

Some functionality of this legacy University Resource Planning system (URP system) should be accessible not only for administrative staff but also for students and advisers. Due to security and convenience reasons, the URP system should not be accessed directly in this case. The idea to provide the necessary functionality is to integrate SAP CM in a Service-oriented Architecture (SOA). This is done by adding Web service interfaces to SAP CM that are orchestrated in the SOA using the Business Process Execution Language (BPEL). User Interaction is handled via a central University Portal.

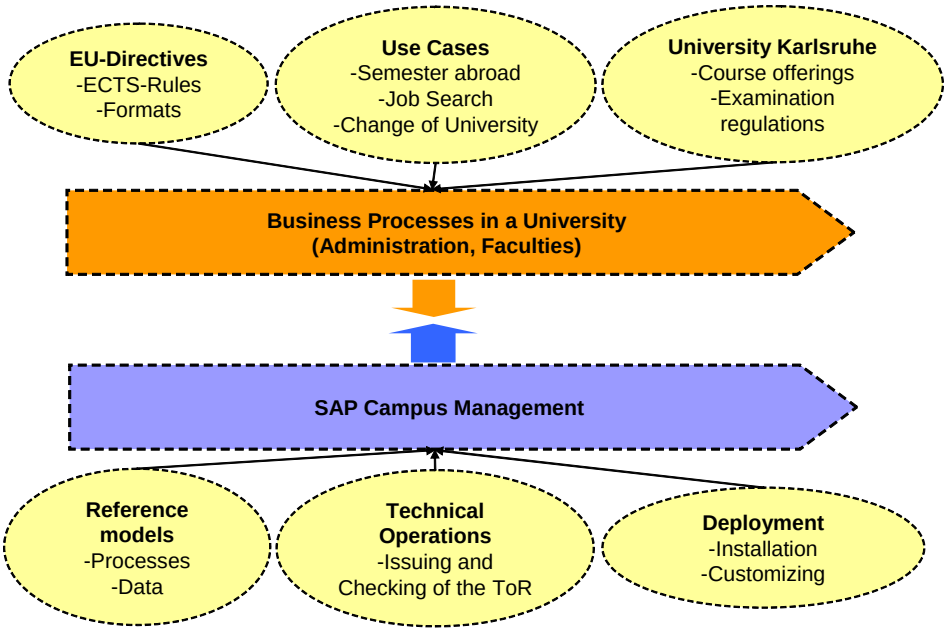
0.2 Progression of the case study

The reader of this WUSKAR case study will receive a basic overview about business processes in a university and how they are supported by SAP Campus Management. To preserve the clearness and to limit the complexity, the focus will be put on one central functionality: the provision of a Bologna-conforming Transcript of Records (ToR).

- Orientation in Bologna subjects and ECTS
- Understanding the needs for a Transcript of Records
- Orientation in SAP Campus Management
- Overview about the Architecture (University SOA)
- Creation of ABAP Interfaces, Core Web services and User Portal
- Final Deployment and Usage

Information 1: Roadmap of the Case Study

Information 1 shows the progression of the case study. Each of these steps can be found in a chapter below. The Orientation in Bologna subjects, the ECTS and the needs for a Transcript of Records will be discussed in the analysis of the business area. The next headword is related to the analysis of the system area. Therein the possibilities and restrictions of SAP CM will be discussed in depth. With an overview of the University SOA it follows the Design Phase. The creation of ABAP Interfaces, the Core Web services and the User Portal can be understood as the Implementation Phase. Finally, this case study is topped off by the Deployment and Usage.



Information 2: Concrete Challenges of the Case Study University SOA

The concrete challenge one has to cope with is depicted in Information 2. The upper part shows the business processes in a university. The used URP system - SAP CM - is shown in the lower part of Information 2. Bringing together these two parts, business and system, is the major goal of every WUSKAR case study and therefore also of this document. Not the concrete solution

stands in the foreground but rather the methodic and structured way to reach this solution. At the end a participant of this case study should be able to master similar challenges on his own.

1 ANALYSIS

The analysis phase gives an overview of the business and the system area. First, in the business area, the focus is on the universities. Then after the business area is introduced the legacy system SAP CM plays an important role. SAP CM is a URP system and possibly fulfills the needs and constraints resulting from the business area.

1.1 Business Area

In 1999 29 European ministers in charge of higher education decided to create a European higher education area. The goal was to improve the mobility of the students and to raise the transferability of qualifications.

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

Sorbonne 1998

- Unification of course units and degrees
- Enhancing mobility of students
- Comparison and equivalence of students
- Achieving flexibility by the use of credits

Bologna 1999

- Establishing the European higher education area by 2010
  - Bachelor / master (two cycle degree)
  - No more obstacles for semesters abroad

Prag 2001 and Berlin 2003

- Reinforcement of the intentions
- Exposing the importance of transferability of qualifications

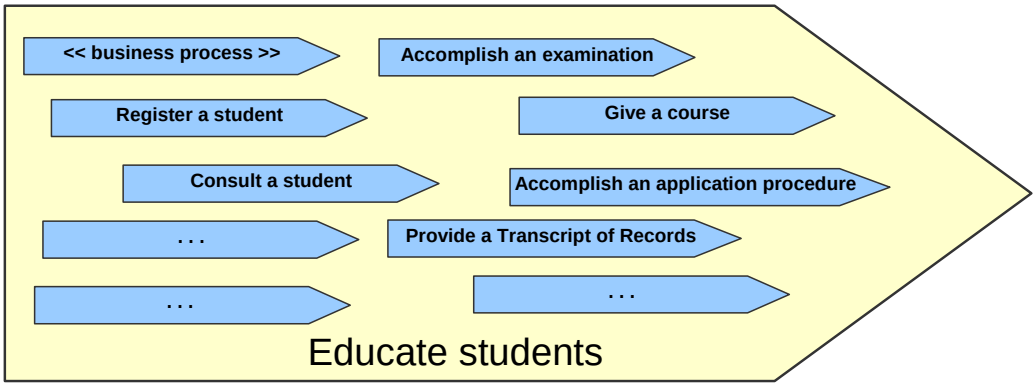
Information 3: Milestones of the Bologna Process

One close central aspect is the mobility of students. There should be no obstacles for a student to study at a foreign university and qualifications such as examination results should be easily transferred in both directions. The following parts analyze the business area to determine the needs and constraints concerning this development.

1.1.1 Involved Business Processes and Partners

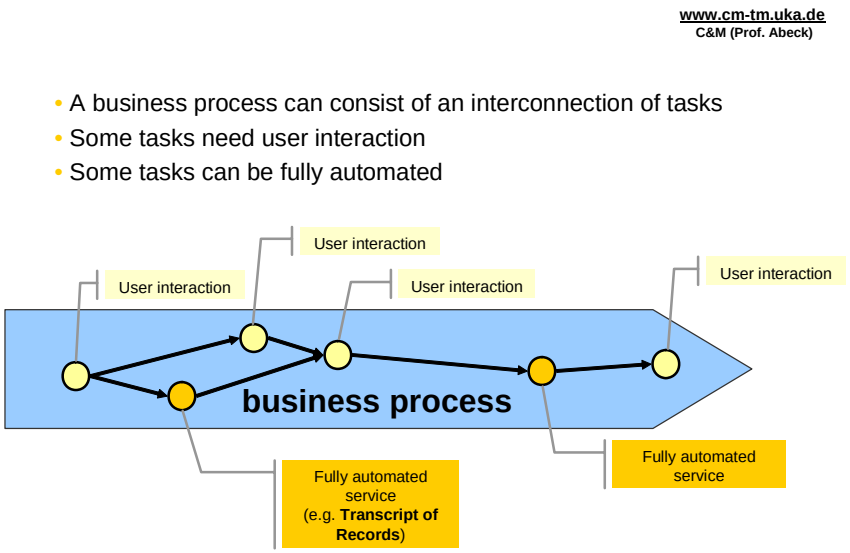
The university is responsible for many things concerning students' education.

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



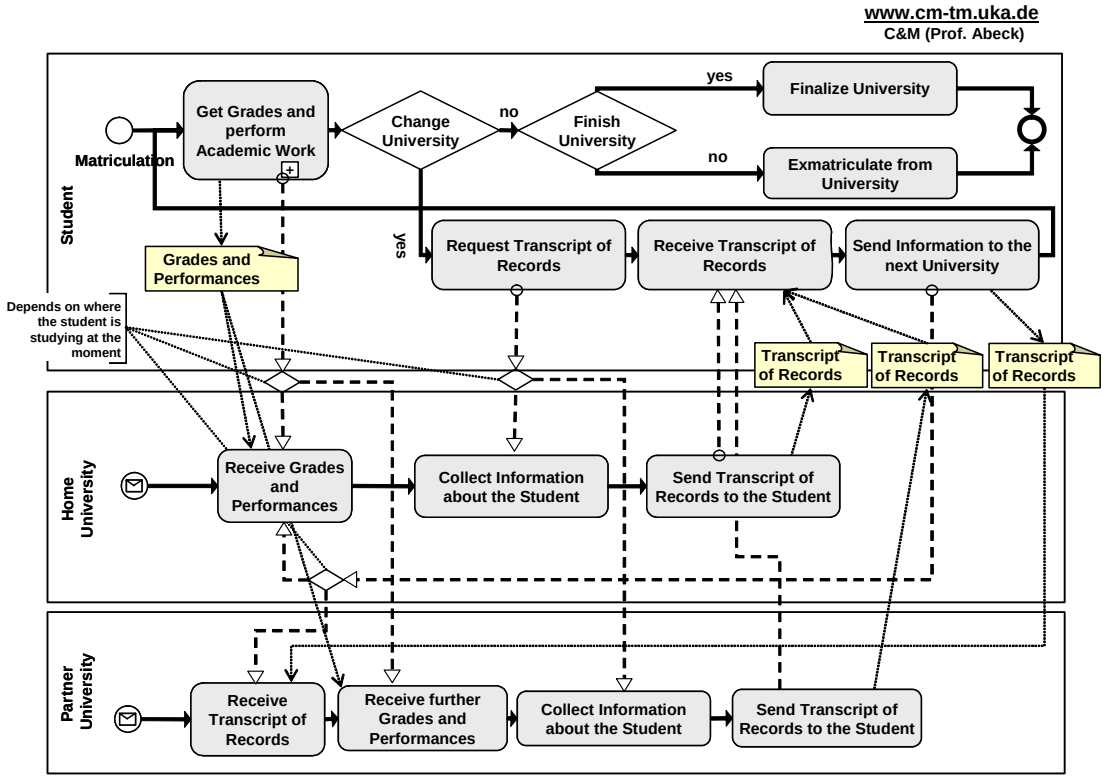
Information 4: Business Processes

Some of these tasks need explicit user interaction. Others can be fully automated. Providing a ToR can be implemented as a fully automated service. Once a person requests a ToR the needed parts of information are collected and displayed automatically.



Information 5: Consult a Student as a concrete Business Process

While remaining at the home university a student participates in course units, receives grades and performs academic work. The home university collects information about these accomplished courses and grades for the corresponding student. For some reasons it can become necessary to have a list of actually fulfilled courses and grades, e.g. a student wants to do a semester abroad or completely change to another university. Then the student requests a ToR at his home university. This progress is modeled in Information 6.

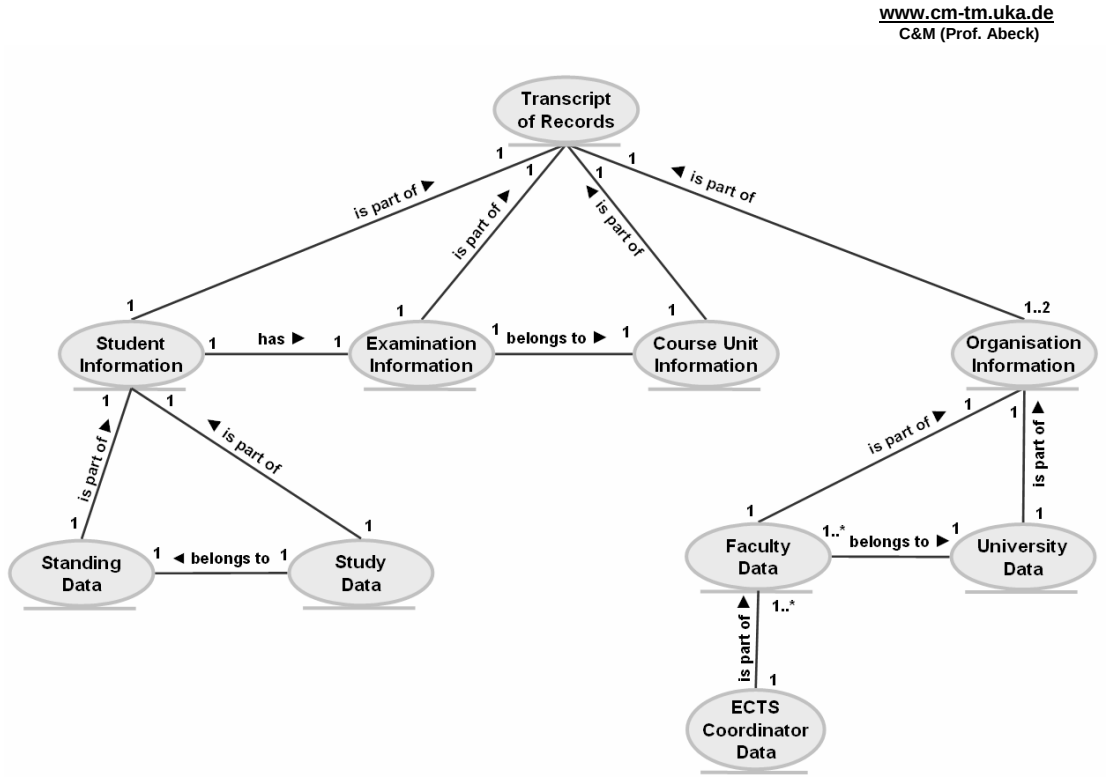


Information 6: Appearance and Usage of the Transcript of Records

At least the student is interested in the transferability of the achieved study results. Therefore the ToR has to record the individual achievements in a standardized form throughout the European Higher Education Area. Furthermore a ToR is meaningful in other contexts as well. E.g. a student consultant should have a ToR available to know the academic status of a student and to offer better guidance.

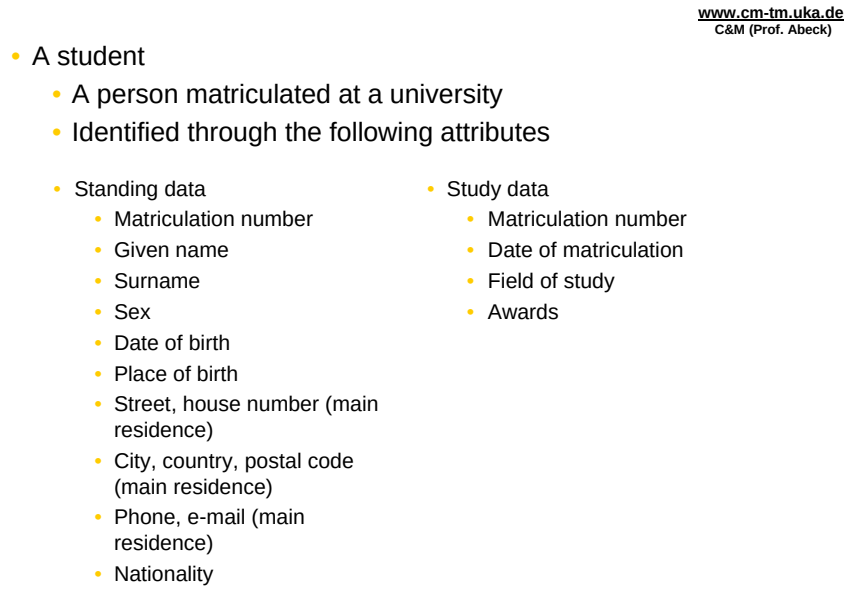
1.1.2 Business Objects

A ToR consists of all relevant information a student needs to change universities. There is no difference whether a student changes the university completely or does one semester abroad. The information about a student and his examination results belong to the ToR just as the course unit information and the organisation information do. Below the different business objects needed for a ToR are analyzed. Information 7 gives an overview of the business objects a ToR consists of.



Information 7: Business Object Diagram

The student information contains two parts of interest – the student's personal data and study related information.

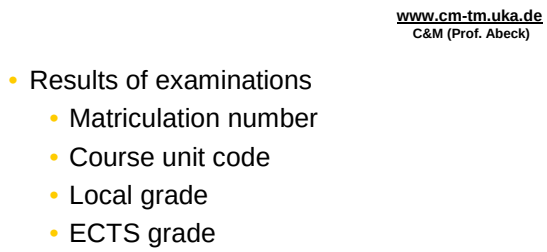


Information 8: Overview about the Student Information

A central aspect for later considerations will be the knowledge of the semantics of the single entities to identify the right objects and clarify what is meant by the entities. That is why the following listing is essential. Some identifiers of Information 8 are listed and explained as a showcase and demonstration of the methodology.

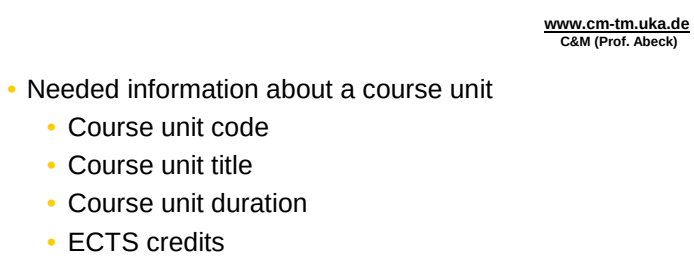
- Given Name
  - All given names should be mentioned here i.e. the first name as well as the middle name etc.
- Home university
  - The home university is the university where the student first matriculated. If a student decides to change the university for a period of time, a distinction is drawn between the home university and the partner university.
- Main residence
  - The main residence is the address where the student can be contacted.
- Matriculation number
  - A matriculation number identifies a person e.g. at a university. In Germany, the matriculation number simply consists of numbers.
- Partner university
  - Every university that wants its students to profit from ECTS establishes partnerships with other universities. Within this partnership much information is exchanged. Information about study programs, contents etc. Thus students can change to these partner universities easily and with the use of ECTS.
- Student
  - A student is someone matriculated at a university.

The examination information consists only of one set named Results of examinations in which the examination results of each student can be found. Information 9 embodies this set in detail. Because a student may have written more than one examination, there is possibly more than one object containing a special result for one student. Each object represents one written examination.



Information 9: Overview about the Examination Information

The course unit information consists of one part which contains all information about a special course. In this context the further information course unit duration and its ECTS credits are sufficient.



Information 10: Overview about the Course Unit Information

A further description of the single points in Information 9 and Information 10 is depicted in Chapter 1.1.4. The course unit code is a unique identification number of a special course. Last but

not least all necessary institution information containing university information and faculty information are required for a ToR to determine where the student's university is, and how his ECTS coordinator can be contacted.

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

- The organisation information again is split into two parts
  - University information
    - University name
    - City
    - Post code
    - State
  - Faculty information
    - University name
    - Faculty name
    - Given name / surname of the ECTS coordinator
    - Telefon of the ECTS coordinator
    - Email / fax of the ECTS coordinator

Information 11: Overview about the Organisation Information

1.1.3 Bologna-conforming Transcript of Records

A showcase body structure of a Transcript of Records is given in Information 12. The previously analyzed object information recurs.

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

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 noticeable 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

Information 12: An empty Transcript of Records

The second page contains solely information about the local grading system and no personalized information. Therefore it is not taken into further consideration. The decomposition of the first page is revealed in the following listing.

- Organisation information
  - Name of the sending / receiving institution
  - Faculty
  - ECTS coordinator
    - maintained by employees of the university

- Student information
  - Personal information
  - Contact information
  - Awards
    - maintained by a student consultant

- Course unit information
  - Course unit code
  - Title and duration
    - maintained by the administration of a faculty

- Examination information
  - Grades of the course units
    - stored by the examination office

1.1.4 Needs and Constraints

It follows from the analysis of the business area that the ToR is of particular importance. A student should be able to receive his own ToR with little effort. To ensure access to a ToR the analyzed object information has to be available and centralized accessible for a student's request. A ToR must fulfil special conventions. Therefore information included about courses and grades are of interest. The grading scale and explanations on the reverse side have no individual use; rather they point out the central specialty, the usage of the European Credit Transfer and Accumulation System (ECTS). This is a student-centered system based on the student workload required to achieve the objectives of a programme - objectives preferably specified in terms of the learning outcomes and competences to be acquired. This system has been successfully tested and is used across Europe. Thus study programmes can easily be compared for all students - local and/or foreign.. The main key features of the credit system are outlined in Information 13.

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

- 60 Credits measure Workload of a one academic Year
- Get Credits after successful completion of the Work
- Workload consists of
  - Lectures
  - Seminars
  - Independent an private Study
  - Preparation of Projects
  - Examinations
  - …

Information 13: What is ECTS – key features

The performance of the students is documented by a local and national grade. The ECTS grading scale ranks the students on a statistical basis. Therefore, statistical data on student performance is a prerequisite for applying the ECTS grading system. Grades are assigned among students with a pass grade as follows in Information 14.

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

- A best 10%
- B next 25%
- C next 30%
- D next 25%
- E next 10%

- Distinction between the grades ‚FX' and ‚F'
  - FX : some more work required
  - F : considerable further work required
- Inclusion of failure rates in the Transcript of Records is optional

Information 14: What is ECTS – Grades

1.2 System Area

In the following a special University Resource Planning system (URP system) – SAP CM will be analyzed. It is supposed to support the business processes from above. Often there are heterogeneous software solutions at a university. Legacy systems are then combined to reach widespread functionality. A single URP system that is extensible due to its modularity and includes up to now a plurality of modules for various business processes will now be looked at. Thus it is flexible for further needs.

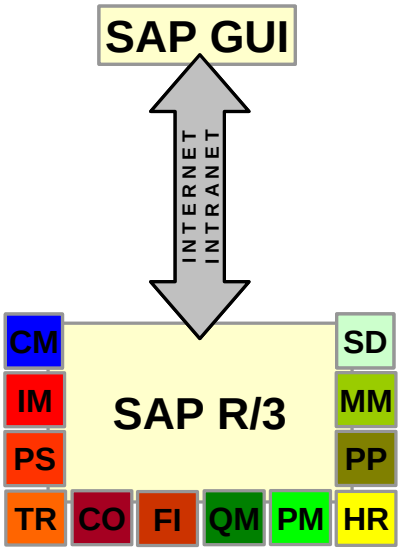
To understand the requirements and design decisions, the technologies and existing developments in the context of SAP CM are analyzed. Therefore technologies supporting interfaces or extensibility are important.

Up to now SAP CM does not provide a ToR conforming to the Bologna requirements. However the business processes pointed out the importance of a ToR.

1.2.1 SAP R/3 and Campus Management

Within the R/3 system SAP uses a client/server architecture. The presentation layer is the interface to the user. It is achieved via the SAP GUI and required for all SAP applications. The SAP GUI directly connects to a SAP R/3 server system. SAP R/3 itself is an application software written in ABAP/4, a SAP native language. Roughly speaking it is related to COBOL and SQL. Various modules can expand the functionality of the central component SAP R/3.

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

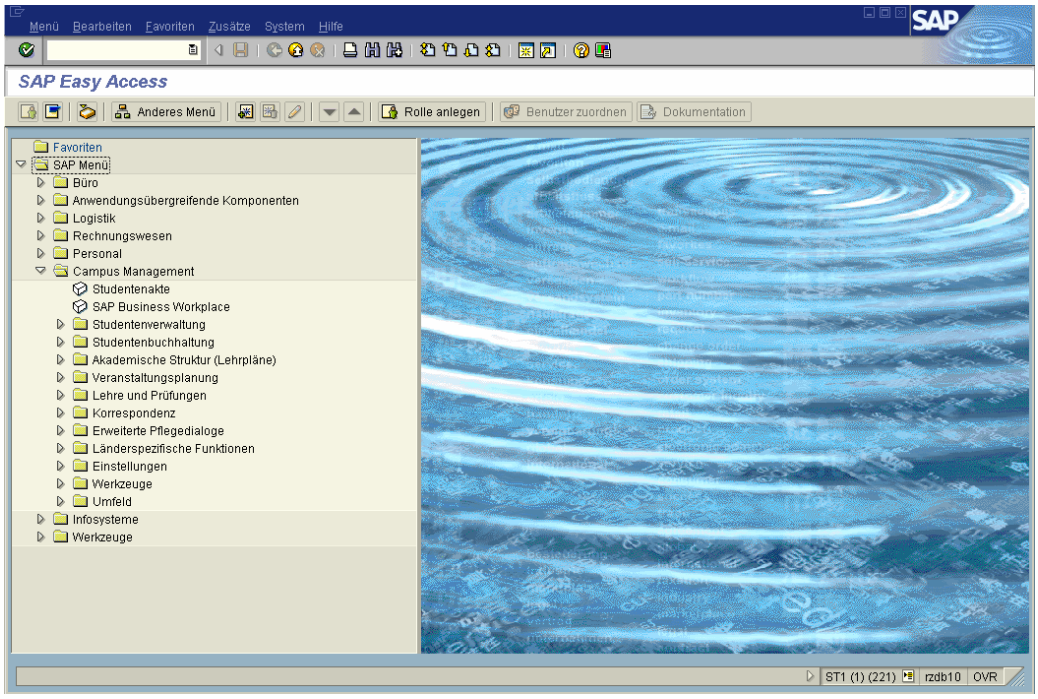


(1) Campus Management (CM)
(2) Sales and Distribution (SD)
(3) Materials Management (MM)
(4) Production Planning (PP)
(5) Quality Management (QM)
(6) Maintenance (PM)
(7) Human Resources (HR)
(8) Financial Accounting (FI)
(9) Controlling (CO)
(10) Treasury (TR)
(11) Project System (PS)
(12) Investment Management(IM)

Information 15: Client-Server-Architecture of SAP R/3

Now the focus will be on the SAP GUI. This is the interface for every user interaction.

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

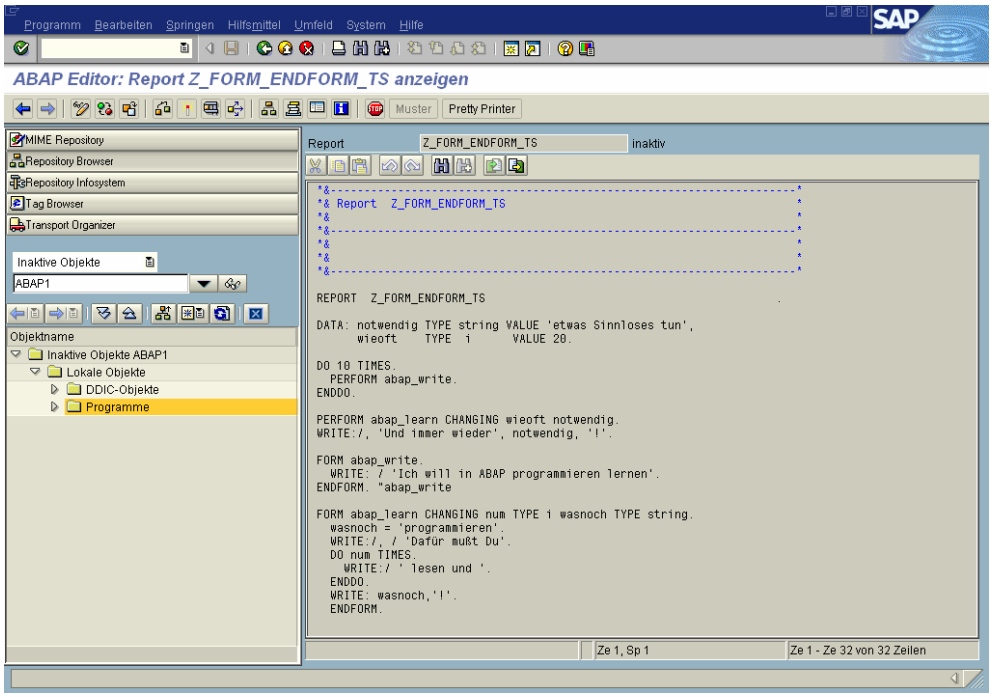


Information 16: SAP GUI

All modules and each function can be used within the GUI from any client connected to the internet and any user with a login. Furthermore the development workbench is integrated into the GUI the starting point to develop new reports and function modules. Reports are programmes written in ABAP. A function module can be created from a report very easily.

Therefore a special client administration assigns permission to single users or groups. Thus only special accounts are authorized to develop new function modules.

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



Information 17: Development Workbench

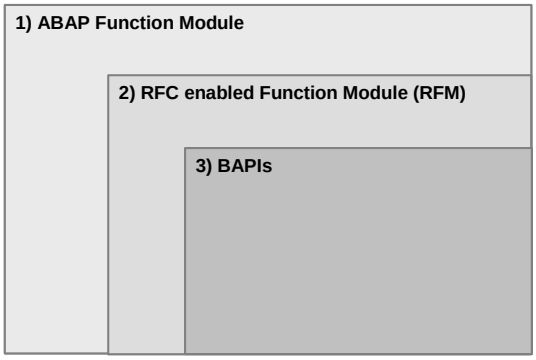
In the workbench one can access every function module and every database available on the server.

1.2.2 ABAP, RFMs and BAPIs

In general, function modules in SAP can be classified according to the following model.

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

- Function Modules in SAP can be divided into three nested Classes



Information 18: Classification

1) The super class of all modularized, procedural ABAP code is presented by the ABAP function modules which have to be created according to a strict encapsulation policy.

2) Function modules written in ABAP which can be used reasonably by other SAP systems or various different applications are created as RFC enabled function modules (RFM).

3) The BAPIs are a certain form of these RFC enabled function modules. Besides the detailed technical requirements, they have to satisfy further requirements regarding nomenclature and the documentation of their interfaces and their operation methods.

Since it is possible that an existing BAPI cannot provide certain functionality, there is the possibility of developing appropriate new BAPIs. In contrast to other developments, the development of BAPIs has to meet high demands in order to fulfil the requirements. Besides a BAPI Programming Guide, the creation is assisted by a wizard function, which simplifies the creation of a BAPI from a function module.

1.2.3 Business Framework and Business Object Types

SAP developed the business framework to enable technical integration and integration of business objects between different SAP components and non-SAP components.
The business framework provides an object oriented structure of the functionality in the business components. Structuring of data and processes is ensured by the business object types. BAPIs - figured in Information 19 - are central components of the business framework. They form a standardized interface to the business objects.

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

- BAPI is an acronym for Business Application Programming Interface
- BAPIs are visible interfaces to other components (software)
- BAPIs allow integration of SAP functions
- BAPIs provide integration on a non-technical layer
  - Stability of coupling
  - Independence of used communication technologie

Information 19: Business Application Programming Interface

1.2.4 Business Object Repository (BOR)

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

(1) SAP business object types and their methods are identified and explained in the Business Object Repository (BOR)

(2) The BOR creates instances of business object types

Information 20: SAP Business Object Repository (BOR)

The BOR is a good starting point for developing new applications or functions. Information about the business object types, interfaces, key fields and BAPI methods can be found here. The runtime environment of the BOR is able to receive requests from client applications and therefore creating instances just in time.

1.2.5 Requirements

There are different methods of generating new functionality to the SAP system. The different methods correspond to the functionality's purpose.

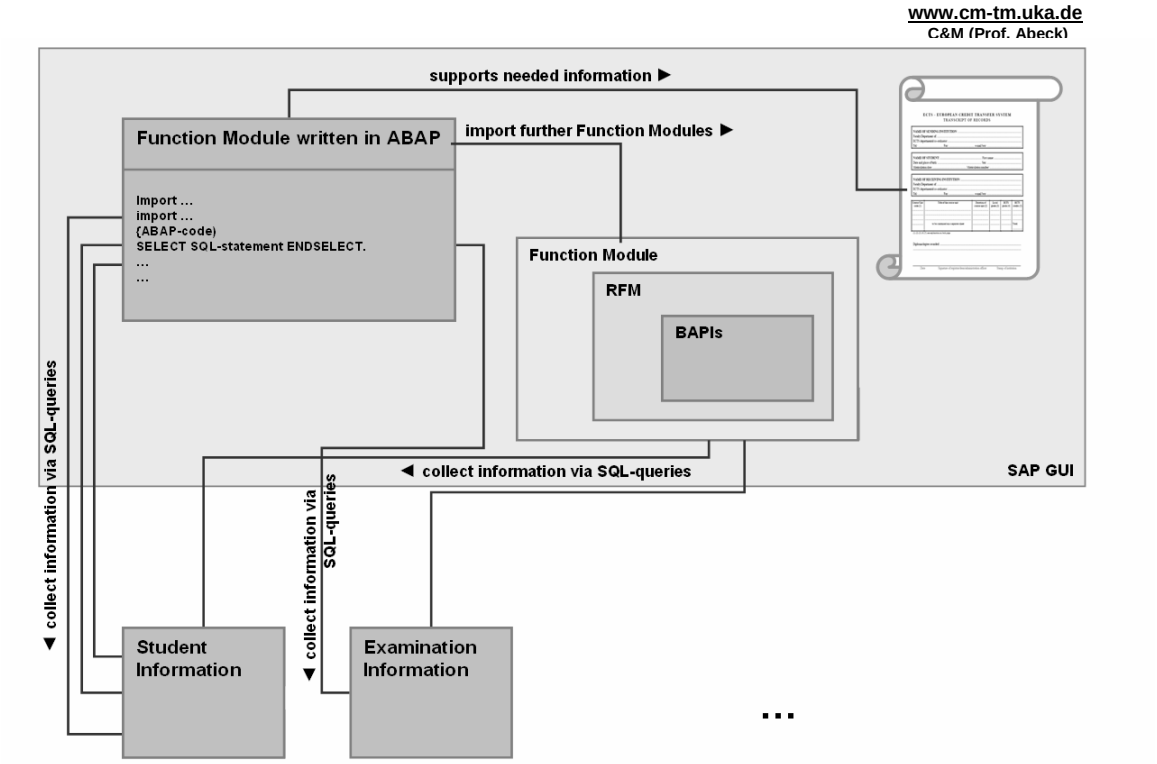
On principle this is done via a report. A report is nothing else than a small program written in ABAP. Existing functionality can be used within a report. Furthermore new Functionality and abstract data types can be developed as well as new modules. Another possibility is to directly create a new function module. This function module has defined input and output parameters. Thus the use of a function module is restricted to interacting with other programs and modules.

Some information is already available as business objects. Thus their interfaces are well defined making the use of such business objects rather simple. This is because the input and output syntax is also well defined for business objects. These interfaces are accessible via function modules or BAPIs. Thus it is recommended to first search for existing BAPIs which cope with the business objects and have increased integrity in further system releases. Further information important in the context of a ToR can be collected, orchestrated or handed over by the use of a report or self generated function module.

So, there are several possibilities to get the needed information:

1. Via SQL query directly accessing the database
2. Using existing function modules
3. Using existing BAPIs
4. Using business objects

Information 21 depicts these different possibilities. The way of retaining special information in the SAP system depends on the kind of information needed i.e. whether there are existing function modules or BAPIs or not. As an example Information 21 deals with student information and examination information representing all business objects. The abstract about student information and examination information is a showcase for the other objects as well. Deriving from the analyzed possibilities the requirement is to find the appropriate way of developing the missing functionality and of finding the needed information.



Information 21: Possibilities for getting a ToR within the SAP System

Some further requirements have to be met to fulfil the business analysis. It is not satisfactory to supply the ToR in the SAP GUI. Presenting the ToR within the SAP GUI presents the following problems:

1. Access Management
   - Every student needs a license / account
2. Rights Management
   - Special limited rights required for a student
3. Functional Restriction
   - Student wants more functionality possibly supported by other systems
4. Thin Clients
   - No installation of client software required. Only an internet browser is mandatory.

The solution to these problems can be found in a higher level, more flexible architecture. This architecture uses the SAP system as an underlying (legacy) software system. Providing functionality of the SAP system, orchestrating different functions to a new aggregated functionality and supporting a user portal as well as various other possibilities are fulfilled within this architecture. Furthermore the future integration of new functionality is simplified. This is supported by a Service-oriented Architecture (SOA) and in our case a University SOA (USOA). This is vital for further considerations.

1.2.6 Conclusion

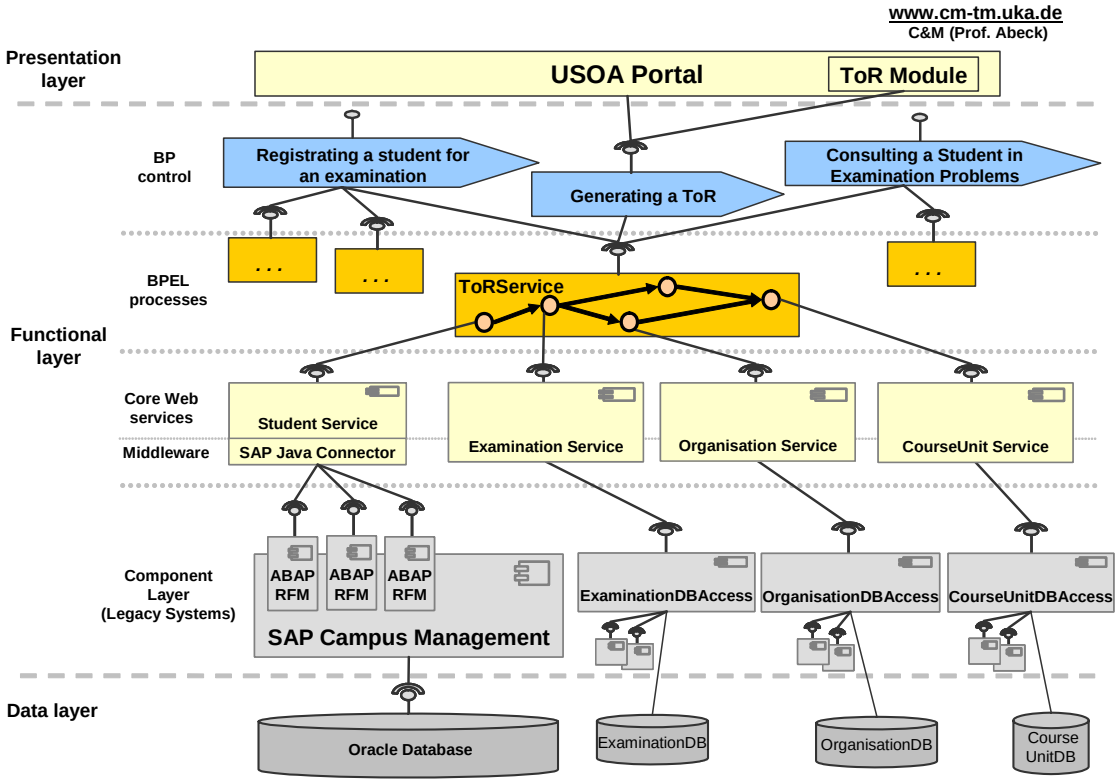
The analysis of the system area depicted the approaches to develop missing functionality inside SAP CM. Thus it seems basically possible to fulfil the needs and constraints from the business area by the use of a SAP CM system. Further considerations how this can be done in particular will be exposed in Chapter 2.

2 DESIGN

The design of the University SOA can be divided into two major parts, the legacy system and the SOA. These two architectural parts will be discussed in detail in this chapter. But in a first step, one has to obtain a general idea about the whole architecture that will be used to achieve a USOA with SAP Campus Management (SAP CM) as the underlying Legacy System.

2.1 General Architecture

Roughly speaking, knowledge of the technologies which will be worked with is known. Information 22 displays all specific layers of the Service-oriented Architecture. Since the system to be used as a source of information is an SAP system, ABAP as SAP's programming language will certainly be used to build the interfaces for SAP CM. Remote Function Modules (RFMs) will be essential to make these interfaces accessible for programs different to SAP, e.g. for Java systems.
Concerning Java, the interface to SAP is the SAP Java Connector (SAP JCo) that operates as a Middleware layer between the Core Web services and the ABAP Function Modules. Those Core Web services are orchestrated in the SOA using a BPEL process representing the ToR Service. Finally, the presentation layer and its user interaction is accomplished via a central University Portal.



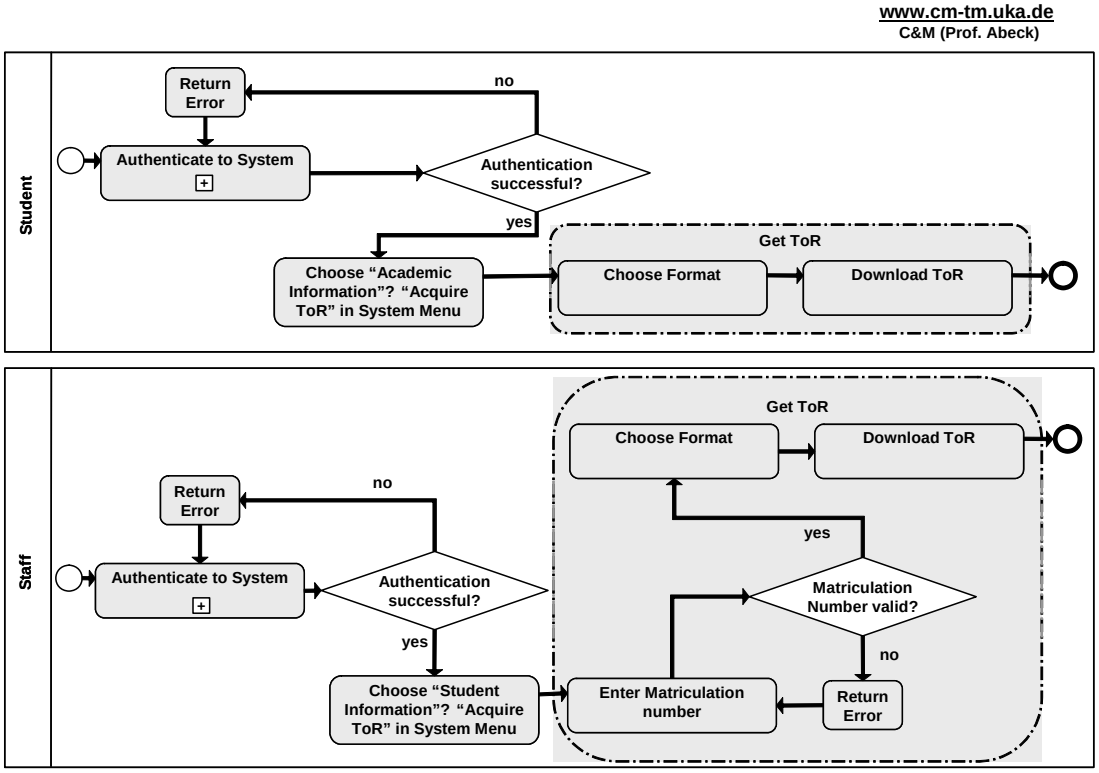
Information 22: DESIGN – General Architecture

The approach now used will be to make a top down design separated by the different SOA layers that are depicted in Information 22. This is very important because the business processes can be put in the foreground at first.

2.2 The SOA Part

2.2.1 User Portal

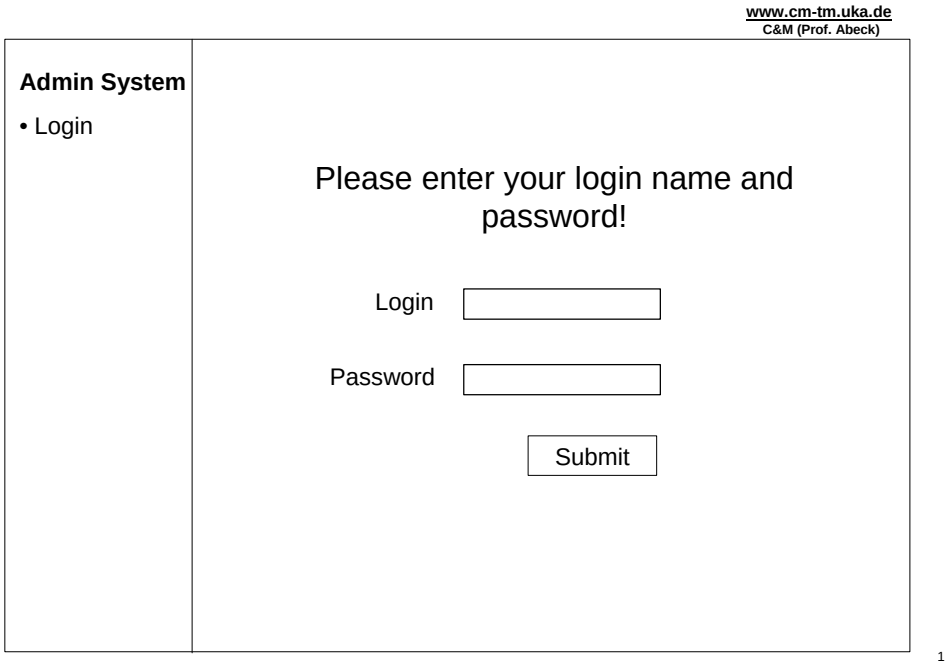
This subchapter describes the design of the university portal and also introduces the technologies which are used to run such a portal. The portal itself should be able to support various business processes such as obtaining a ToR, administrating personal information, printing a study timetable, applying for examinations and so on. To reduce the complexity and the bulk of information in this chapter, the focus will only lay on the support of one business process namely getting a ToR.



Information 23: BPMN Process Model of the GUI for ToR queries

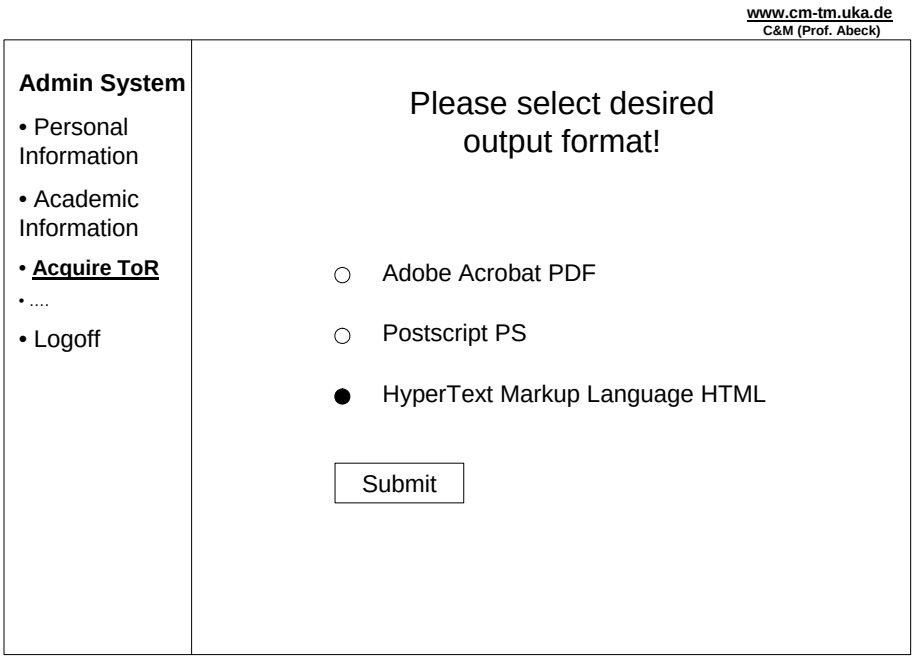
As can be seen in Information 23, the process of obtaining the ToR is embedded in a larger framework where the retrieval of the ToR is only a part of the functionality. There is a distinction between users, namely students and staff. This distinction is necessary, since students can only obtain their own ToR while staff members, academic as well as non-academic, have access to various students' ToRs.

At first a user, no matter if a student or a staff member, has to authenticate to the system as shown below.



Information 24: Login Screen

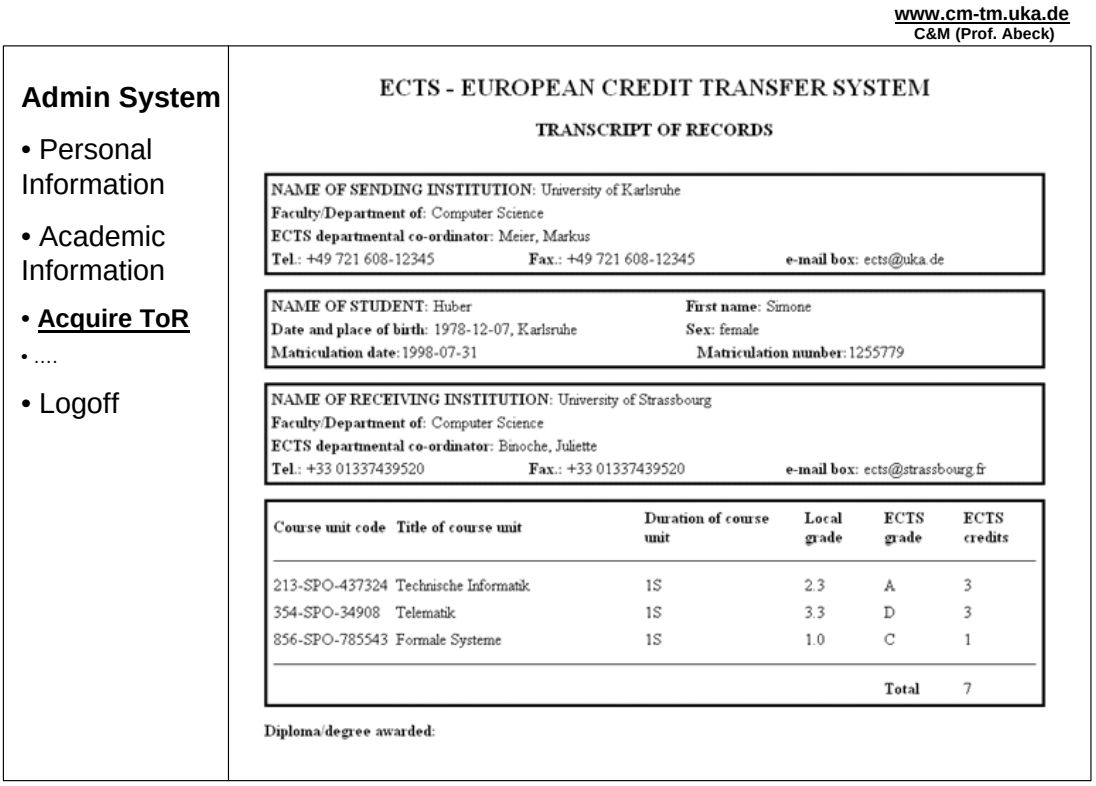
Naturally, this login can fail, so in this case the user has to provide a different login name or password (Information left out at this point). If the user is a student and chooses to obtain her/his ToR by following the links in the system to Acquire ToR, the only question she/he is asked is which output format is desired. This is more convenient for the user, in contrast to dictating the format at first hand.



Information 25: Choose output format

The choice is given between the most common formats, though this list can be expanded or reduced in the future. In case the user chooses, as it can be inferred in the Information,

"HyperText Markup Language HTML", she/he gets back an HTML page showing the complete ToR. This example of an output ToR can be seen in the following Information.



Information 26: Display ToR (HTML page in this case)

The process Get ToR has now ended (see BPMN model), since the user (a student in this case) could acquire the desired information. The case of the user not being a student, but a staff member, is slightly different since an additional step, namely the choosing of the student whose ToR should be retrieved, has to be considered. This can be accomplished by simply querying the user for the student's matriculation number, which is unique for a student at the whole university. This requires following the links to Acquire ToR in the first place of course.

The following passage describes the technologies used to implement the portal as well as the reasons which led to these choices.

(1) Apache Tomcat Servlet Container
  (1) Can be used as stand-alone Web Server
  (2) Holds and executes Servlets
  (3) Fits in existing Framework, widely used

(2) Servlets
  (1) Work according to Request-/Response Scheme
  (2) Can execute Java Code
  (3) Performance Advantages compared to CGI

(3) Java ServerPages JSP
  (1) Used when a lot of HTML is needed
  (2) Are compiled into Servlets

Information 27: Apache Jakarta Tomcat

Apache Jakarta Tomcat itself is a Servlet container. It implements the Servlet and Java Server Pages (JSP) specifications. It can be considered as a web server, as it does not need any additional web server such as Apache Web Server to function. The reasons for using Tomcat are that it fits into the existing framework of Java applications, its widespread utilization, its highly developed code and the license under which it is distributed. There are no licensing costs, either. Servlets are server-side Java programs which work according to a request-/response scheme. They are used as extensions of web servers and require a Servlet container such as Tomcat to run. While processing requests, any Java code can be executed, including e.g. database accesses. The advantage of Servlets compared to CGI scripting languages is performance, since they stay in the Java Virtual Machine's memory until the shutdown of the server.
Java Server Pages technology is an extension of the Java Servlet technology. Servlets are platform-independent, server-side modules that fit seamlessly into a Web server framework and can be used to extend the capabilities of a Web server with minimal overhead, maintenance, and support. Unlike other scripting languages, servlets involve no platform-specific consideration or modifications; they are application components that are downloaded, on demand, to the part of the system that needs them. Together, JSP technology and servlets provide an attractive alternative to other types of dynamic Web scripting/programming by offering: platform independence; enhanced performance; separation of logic from display; ease of administration; extensibility into the enterprise; and, most importantly, ease of use.

(1) Apache Axis
  (1) Open Source Framework
  (2) Based on Java and XML
  (3) Implements the Simple Object Access Protocol (SOAP)
  (4) Fits into Apache Tomcat Framework

(2) SOAP
  (1) Standard for exchanging XML based Messages
  (2) Mostly RPC Messages
  (3) Uses HTTP for Transportation

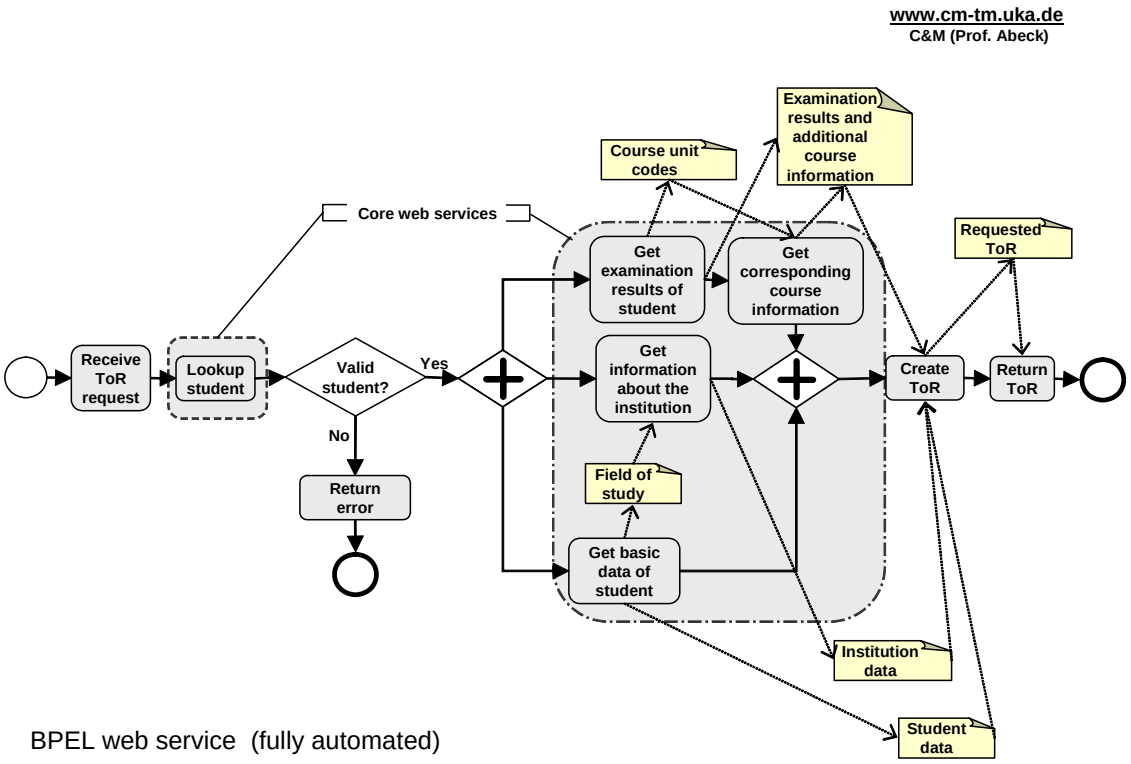
Information 28: Apache Axis

Apache Axis is an open source framework and is based on Java and XML. It is a technology enabling distributed communication among computer systems. To achieve this, the Simple Object Access Protocol (SOAP) is implemented. The highly developed interoperability between Axis and Tomcat justifies the use of this framework. SOAP is the state-of-the-art when exchanging messages between Web service clients and Web services in a distributed computer environment. It is based on a XML format and the most common message pattern is the remote procedure call (RPC), meaning that a client requests a response to this message. Communication is carried out using the hypertext transfer protocol (HTTP).

Another way to access Web services is to use the Web services Invocation Framework (WSIF). It is a simple Java API for invoking Web services, no matter how or where the services are provided. WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol (SOAP) APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed. WSIF allows stubless or completely dynamic invocation of a Web service, based upon examination of the meta-data about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime, and the calling service to defer choosing a binding until runtime. Finally, WSIF is heavily based upon WSDL, so it can invoke any service that can be described in WSDL [APACHE-WSIF].

2.2.2 BPEL Process

The BPEL process providing a Transcript of Records is described in detail in this subchapter. Based on the following BPMN diagram which reveals the necessary steps obtaining a ToR the BPEL process can be derived. The following paragraph describes this in detail.



Information 29: 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. 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 is successful a ToR in XML format is returned [We05].

The ToRService uses several Web services that will be designed in the next chapter to collect all information needed 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 implement 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
- A 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 operating this service was to use BPEL

Some parts of the analysis phases were done using BPMN. Reaching the design 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

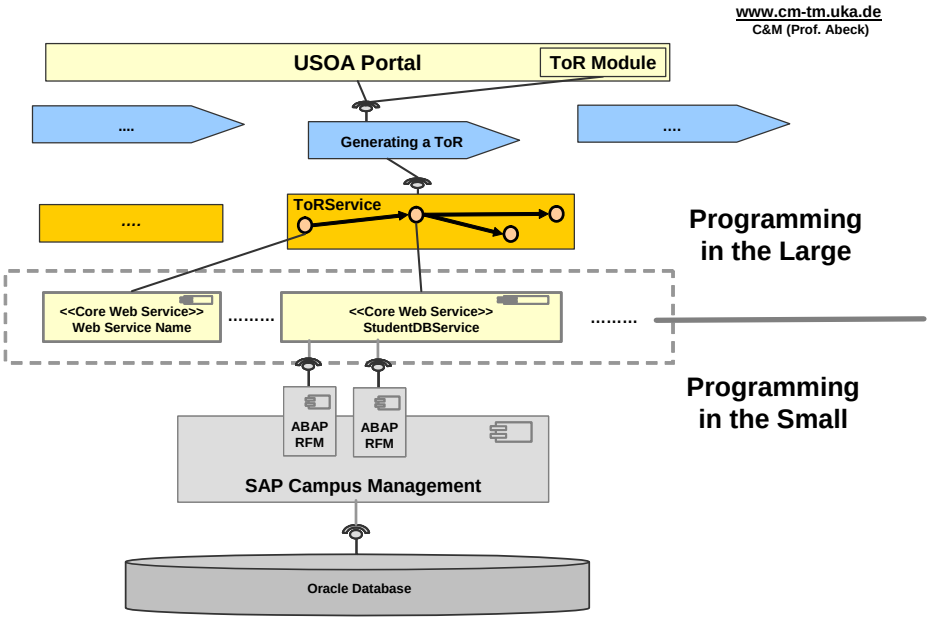
reduce the gap between the business process design and the implementation. The rest of this subchapter illustrates how the BPMN graphs can be converted to BPEL code under the precondition to maximize the automatically executable parts. The lack of available software products, which can translate BPMN to BPEL in a way so that this code can be deployed or even efficiently developed, causes a work-around [We05].

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 the ToR process that is shown in Information 29, 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 machine supported because all the needed information is listed in the Web service description if referring to Web services that already exist [We05].

That is the point where a BPEL design tool like the Oracle BPEL Designer comes into play. The alternative to 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 above. 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 [We05].

2.2.3 Core Web services

This subchapter deals with the core Web services which encapsulate the access to SAP Campus Management. When seeing Information 30 as a SOA classification the core Web services can be considered as interface between the "Programming in the Large" and "Programming in the Small".



Information 30: Core Web services – SOA classification

The task of the core Web services is to offer a specific functionality supported by a well specified service interface. They are then being assembled to composite Web services in order to directly support business processes. The challenge for software developers is no longer the coding of the required functionality but the orchestration of already existing Web services to executable business processes. This approach has commonly been referred to in relevant literature as "Programming in the Large" whereas the development of core Web services is known as "Programming in the Small" [EM+05].

As already seen in the last subchapter there are four core Web services which have to be implemented:

- StudentDBService – Provides all standing data and all study relevant data of a student
- ExaminationDBService – Reads out all examination data of a student
- CourseUnitDBService – Offers all data corresponding to a specific course unit
- OrganisationDBService – Gives back all data of a specified organisation

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 or University resource planning systems
- Modeling the scenario of distributed databases or URP systems all over the campus which seems to be very close to reality

Most of the classes that implement a Web service can be generated automatically out of the WSDL using appropriate tools. So the WSDL file for each core Web service has to be specified in design phase only. 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 [We05]. An example of such a WSDL is found in 3.2.

The following part of this subchapter works out the design of one exemplary core Web service, the StudentDBService. The design of the other three services is described in detail in [We05]. Derived from the business analysis in Chapter 1 the StudentDBService should provide the following functionality:

- It receives a matriculation number as input and returns all relevant standing data (as defined in the business analysis in Chapter 1 before) and study data (also as defined in the business analysis in Chapter 1 before) of the corresponding student. The data selection is optimized for the composition of a Transcript of Records.

- This functionality is split into two operations, one does as described in the preceding paragraph, the other one checks whether a student exists in the system. This is necessary for generating error messages when using this Web service.

Both operations do need to be defined semantically or and syntactically. The following information slide illustrates this.

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

- Each of the two operations has the following input and output fields:

Operation	Syntax	Semantics
IsStudent	Input: long Output: boolean	Input: Matriculation number of the student Output: true: If student with this matriculation number exists in system false: If student with this matriculation number does not exist in system
GetCompleteSet	Input: long Output: SetofStudentDB	Input: Matriculation number of the student Output: Set of all standing data (as defined from the business view before) and study data (also as defined from the business view before) of the student with this matriculation number

Information 31: StudentDBService and its semantic and syntactical definition

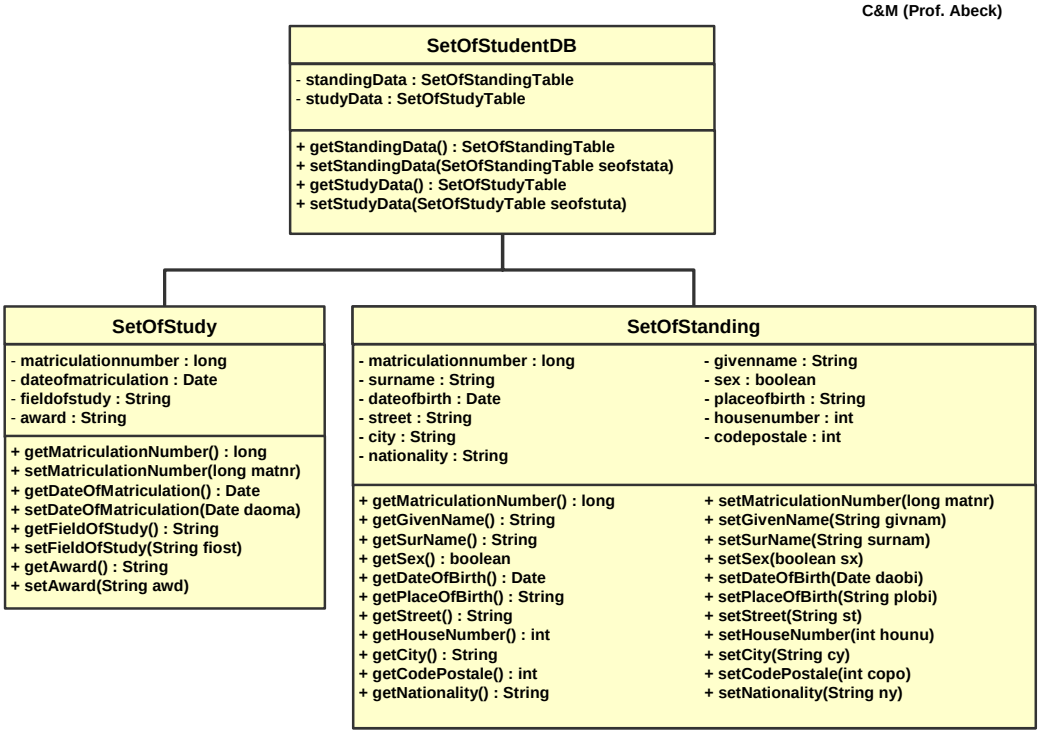
The operation GetCompleteSet returns the output type SetofStudentDB which is a structured type and has to be specified as well. Information 31 shows this in detail. This specification is definitely based on the results on the analysis of the business area in Chapter 1.

- The type SetofStudentDB is composed by two other structured types described below:

Type	Syntax	Semantics
SetofStandingTable	long	Matriculation number of the student
	string	Given name (all given names separated by spaces)
	string	Surname
	boolean	Sex (true is male, false is female)
	string	Place of birth (name of the city)
	dateTime	Day of birth
	string	Nationality
	string	Street (main residence, see definition from business view)
	int	House number (main residence, see above)
	int	Postal code (main residence, see above)
	string	City (main residence, see above)
SetofStudyTable	long	Matriculation number of the student
	dateTime	Date of matriculation
	string	Field of study
	string	Award

Information 32: StudentDBService – semantic and syntactical type definition

Now all necessary information is collected and the class model of the type SetofStudentDB can be designed as shown below.

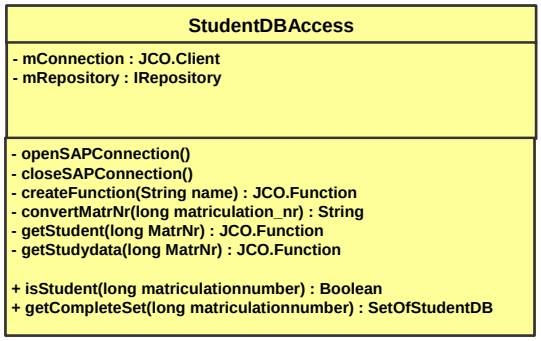


Information 33: Components which encapsulate one return set of the StudentDBService

Another component is the class StudentDBAccess that embodies the connection to SAP CM and provides all needed methods to query information of the SAP system. As one can see, some classes (e.g. JCO.Function) of the JCo already appear in the UML class diagram below.

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

- 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



Information 34: Component that encapsulates the Access to SAP Campus Management

Thus the next subchapter describes what the JCo is and does and how it can be used in this context.

2.3 The Legacy System Part

2.3.1 SAP Java Connector

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

(1) The SAP Java Connector (JCo) is a Middleware connecting ABAP and Java applications

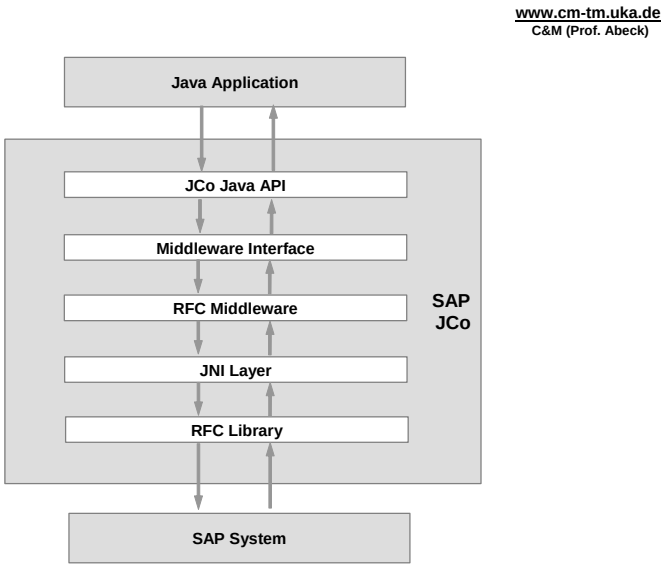
(2) JCo enables the interaction and communication between Java components and SAP applications in both directions

(3) JCo is a bundle of Java classes which have to be imported into a Java program

Information 35: SAP JAVA CONNECTOR – Overview

The SAP Java Connector (JCo) is a versatile solution for connecting to business partners which are running a Java and not a SAP system. It enables the readout of data from a SAP system for integrating this data into systems other than SAP as well as exhibiting the ability to develop a simplified front end for certain SAP functions. Further simplification of communication is

achieved by masking out codepages, data type conversions and details of RFC to a great extent. All of the JCo functions are optimized for communication with SAP. Moreover, JCo is platform independent since it is based on Java and may be used along with a valid SAP license.
The general architecture of JCo appears as follows.



Information 36: JCo Architecture

Starting from a
1) Java application, a Java method is routed to the
2) JCo API (Application Programming Interface) and another
3) Middleware interface. This interface is used to mediate between the JCo API and the
4) RFC middleware. At this point, the method is converted to a RFC (ABAP) call with the use of the
5) JNI (Java Native Interface) layer and the help of an
6) RFC Library and finally sent to the
7) SAP system.
The same steps in reverse are taken when a RFC call is converted to a Java method which is then routed to the Java application, so the JCo can be seen as bidirectional, not unidirectional.

Recapitulating, the JCo gives the Core Web services the ability to communicate with a SAP system. Hence the task of the JCo is to act as a middleware layer between the component layer (SAP R/3) and the Core Web services. In addition the switch-over to the following chapter can be done easily. It will describe in detail the ABAP components used by the JCo and for this reason used by the Core Web services too. To obtain further information about syntax, classes and methods of the JCo refer to [C&M-JCo].

2.3.2 ABAP

As already seen in Chapter 1 ABAP is the technology to work with when one wants to build interfaces for a SAP R/3 system. Those interfaces are called Remote Function Modules (RFMs) or Business Application Programming Interfaces (BAPIs) if they are already filled with business logic. Within the scope of this case study, focus will only be on the RFMs because they are easier to program and they do not need a very detailed documentation. Later on, every RFM can be upgraded to a BAPI, but this is not the task here.

The main entry point for the creation of RFMs within the SAP GUI is the so-called Function Builder that can be accessed via the transaction code SE37. It can be used to show, create or delete function modules and BAPIs.

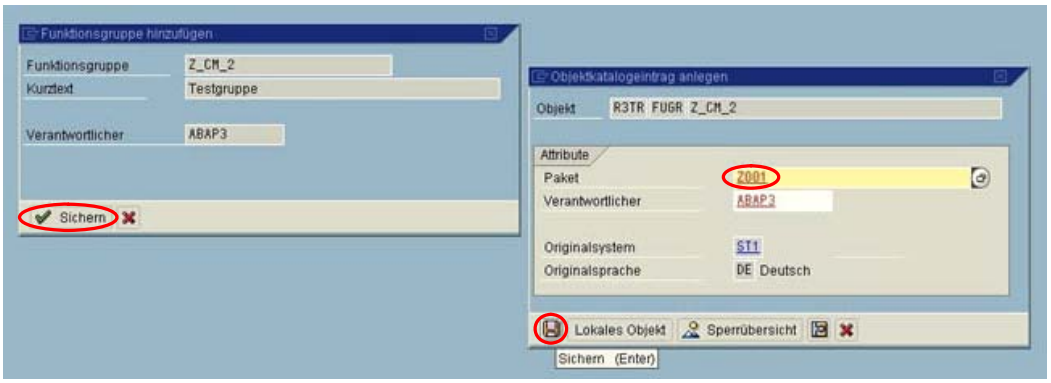
www.cm-tm.uka.de
C&M (Prof. Abeck)

Information 37: Creation of a function group

Before one can build one's own function modules a new function group has to be created. This is necessary for the hierarchical organization and the grouping within the SAP system. The following steps will lead to a successful creation:

- Select Springen → FGruppenverwaltung → Gruppe anlegen.
- Enter the name of the function group and a short description text. The name should always begin with the letter Z. This is the customer name space. Otherwise this ABAP program can get lost trough a release update.
- Then, choose Sichern.
- As the last step, asks to select the package name. In the WUSKAR environment at the Universität Karlsruhe (TH), this will always be Z001.

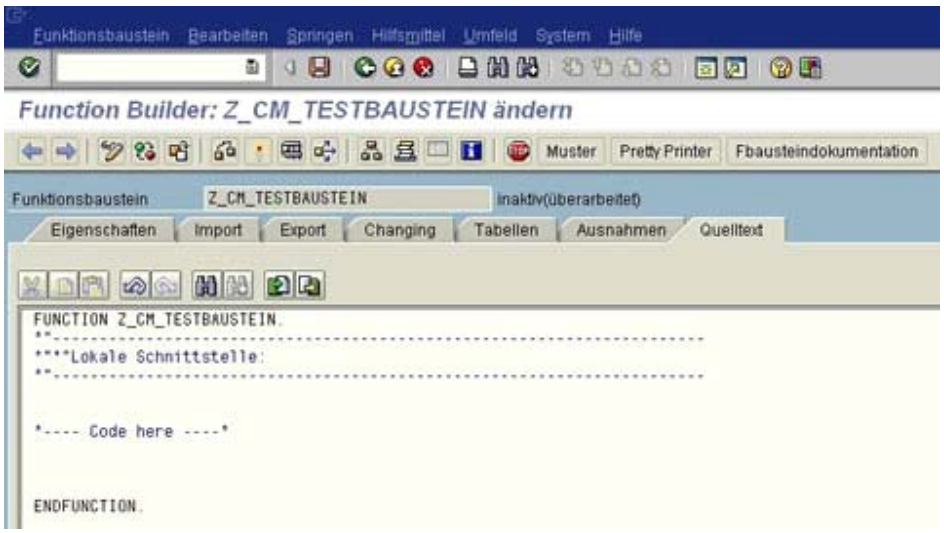
www.cm-tm.uka.de
C&M (Prof. Abeck)



Information 38: Creation of a function group – part two

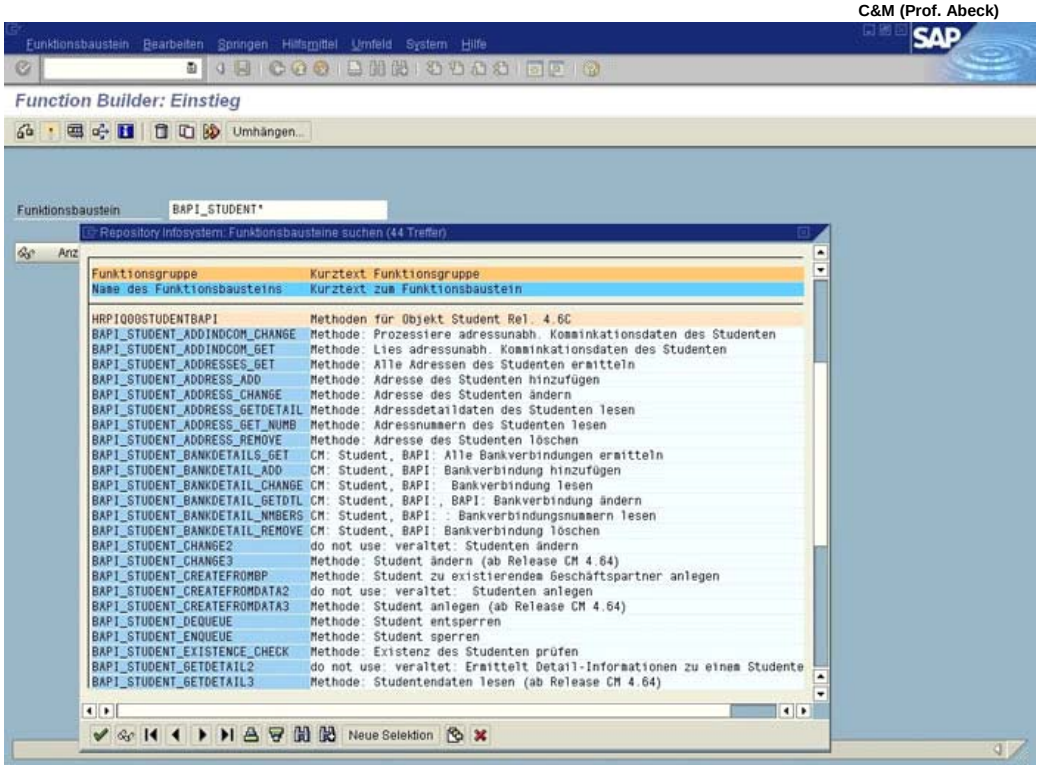
Information 22 and 23 depict all those steps to create an individual function group. Once the disk symbol for the finishing of this process is selected, the new function group exists and can be used when programming new ABAP function modules. To do so, the Function Builder is

again selected, a new function name (e.g. Z_CM_STUDY_DATA_GET) is entered and Anlegen is clicked. A screen very similar to the following will appear.



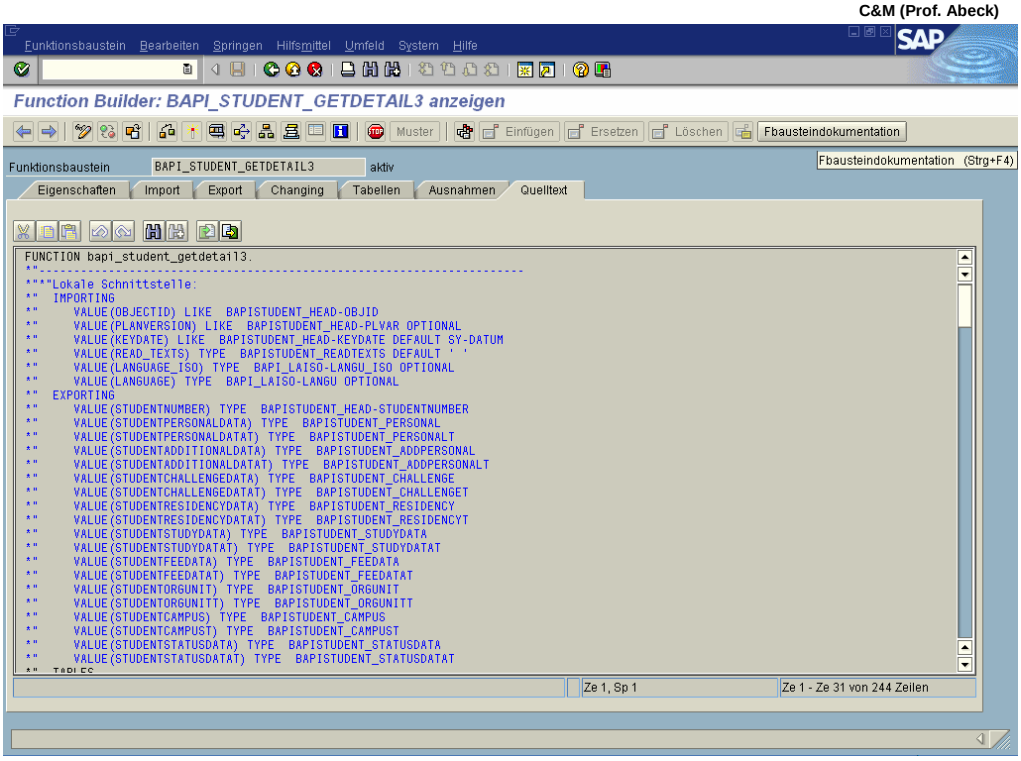
Information 39: Creation of a function module – ABAP source code editor

The first step will be to specify the import and export parameters of the function module. Therefore the corresponding tab pages must be selected and the needed parameters entered. There is the possibility of marking import parameters as “optional”, then they must not be filled when using the function module. Furthermore, if the module should have to export much structured information, the use of tables which can contain more than one row and are therewith a very flexible way to return complex result data is possible. In normal cases, it is strongly recommended to use already existing function modules and BAPIs within user function modules. The other way would be to directly access the databases of the SAP system. This is possible but may cause many problems. Database structures may for example change through a release update or especially writing data may cause inconsistency within the whole database. To prevent those entire problems one should be able to identify the right function modules. Again the Function Builder is crucial. It can be used to search for function modules and afterwards to analyze them.



Information 40: Searching for existing function modules

Nearly every BAPI that is available within SAP CM is named beginning with BAPI_STUDENT. Similarly the names of the normal function modules begin with HRIQ. Taking these two strings as search strings and clicking on the magnifier symbol, one will obtain a very long list of all those BAPIs or function modules. Every entry is endorsed by a short description text that can be utilized to acquire a first idea about which functionality this module provides. To receive a deeper view, double clicking the labelling of the relevant function module or BAPI will lead to a screen view corresponding to the following slide.



Information 41: Analyzing function modules – import and export parameters

The documentation of the function module as well as the import and export parameters and also the source code can be analyzed now. This is of utmost importance in realizing how a function module or BAPI works exactly.

Having this knowledge about existing function modules and BAPIs in mind, the new function modules necessary for the support of the StudentDBService can be designed now. As seen in the design of the StudentDBService in Chapter 2.2.3 this core Web service has two operations, isStudent and GetCompleteSet. The operation GetCompleteSet can logically be divided into two parts, one part for the standing data of the student and one part for the study relevant data. Three ABAP function modules can be designed as interfaces for SAP CM as the conclusion of this separation. In each of these function modules, the student is identified by his matriculation number as a unique identifier:

- One named Z_CM_IS_STUDENT providing the check if a specified student does exist in the system
- A second one named Z_CM_STUDENT_GET offering all standing data of a specified student
- And a last one named Z_CM_STUDY_DATA giving back all study relevant data of a student

While analyzing existing function modules of the SAP system, four modules can be identified which can provide a functionality that can used by the three new function modules mentioned above. They are named HRIQ_STUDENT_NUMBERS_GET, BAPI_STUDENT_GETDETAIL3, BAPI_STUDENT_ADDRESS_GETDETAIL and HRIQ_STUDENT_STUDIES_GET_RFC. They implement the following functionality within SAP CM:

- HRIQ_STUDENT_NUMBERS_GET reads the matriculation number of a student and returns the SAP-internal object ID of this student. This so-called object ID is a number value used to represent objects within SAP CM (and a student is an object). If the student does not exist in the system, the object ID as return value will be 0. This fact can be used to realize the function module Z_CM_IS_STUDENT.
- BAPI_STUDENT_GETDETAIL3 needs the SAP-internal object ID as input value and then returns most of all standing data of this student. This is very useful for Z_CM_STUDENT_GET.
- Also useful for Z_CM_STUDENT_GET is the function module BAPI_STUDENT_ADDRESS_GETDETAIL. As input value the object ID is necessary. The output will be all address data of a student.
- Last but not least the function module HRIQ_STUDENT_STUDIES_GET_RFC is of great importance for Z_CM_STUDY_DATA. It reads out the study-relevant data of a student such as course of studies, matriculation date and so on.

Summarizing all necessary information for the implementation of the ABAP function modules are collected now and a step-over to the next chapter can be triggered.

3 IMPLEMENTATION, DEPLOYMENT AND USAGE

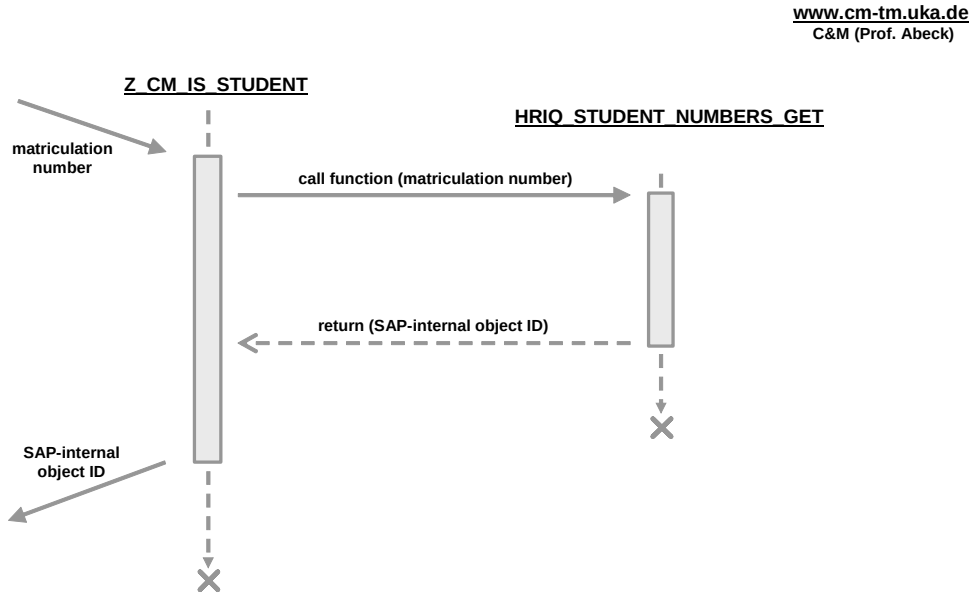
3.1 Implementation

The implementation phase as well as the design phase is fragmented into subchapters separated by the layers worked out in the general architecture. The approach in Chapter 2 was to do the design top down. This was motivated focusing on the business processes and their support from the topmost layer till inside deep ABAP code. In contrast the implementation will be done bottom up. Then the system view stands in the foreground and the implementation of the components as well as the implementation of the generic and aggregated Web services in each case can be used to build upon. This means, for example, to be able to implement the core Web services, the ABAP components have to be already implemented. It is the same with the aggregated BPEL Web service which uses the core Web service which should already be available.

3.1.1 Component Layer – ABAP Interfaces

The component layer is represented by the ABAP interfaces integrated and implemented within SAP CM. ABAP interface stands in this case for RFMs that are implemented with ABAP. It is not possible to use another programming language because extensions to the SAP module Campus Management are restricted to ABAP as the programming language. Contrarily other SAP modules like Human Resources (HR) in combination with SAP NetWeaver™ also support the Java Stack as a second programming environment.

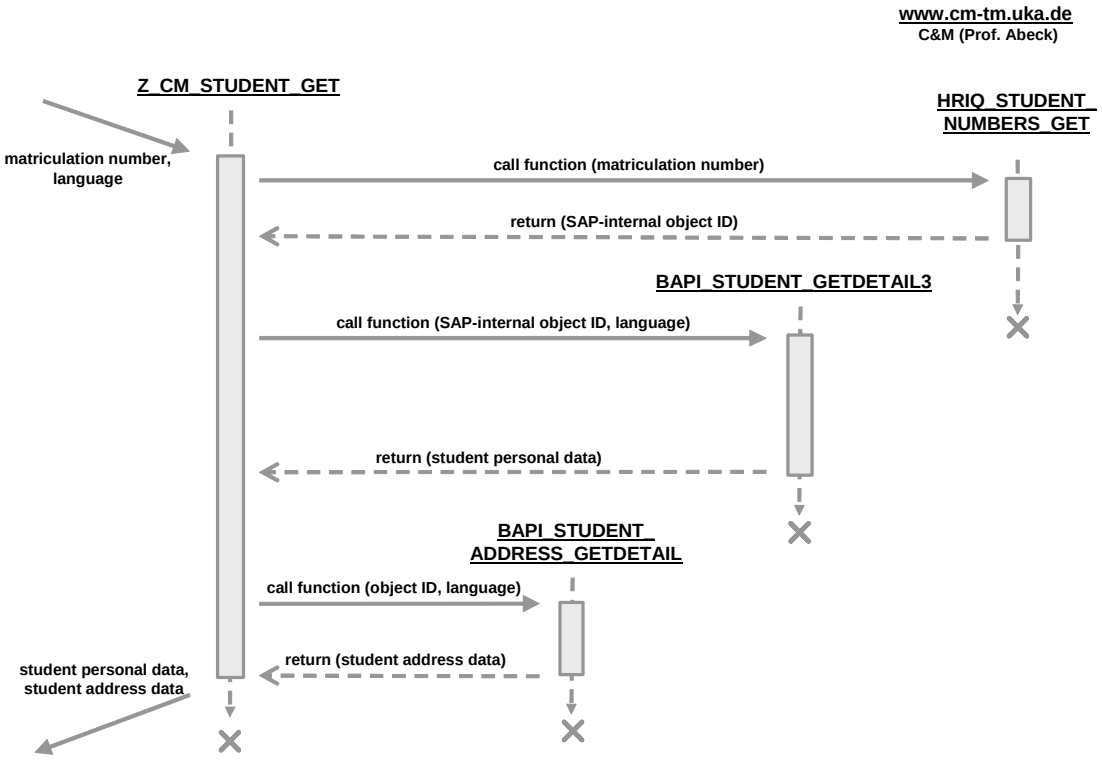
Because this is a WUSKAR case study and the "proof of concept" principle is ostensible, the implementation and design of the ABAP components was restricted to the integration of one core Web service (the StudentDBService) which has already been described in the last chapter. To appropriately support the functionality of this core Web service, three RFMs were designed and can be implemented as depicted below.



Information 42: IMPLEMENTATION – ABAP function module Z_CM_IS_STUDENT

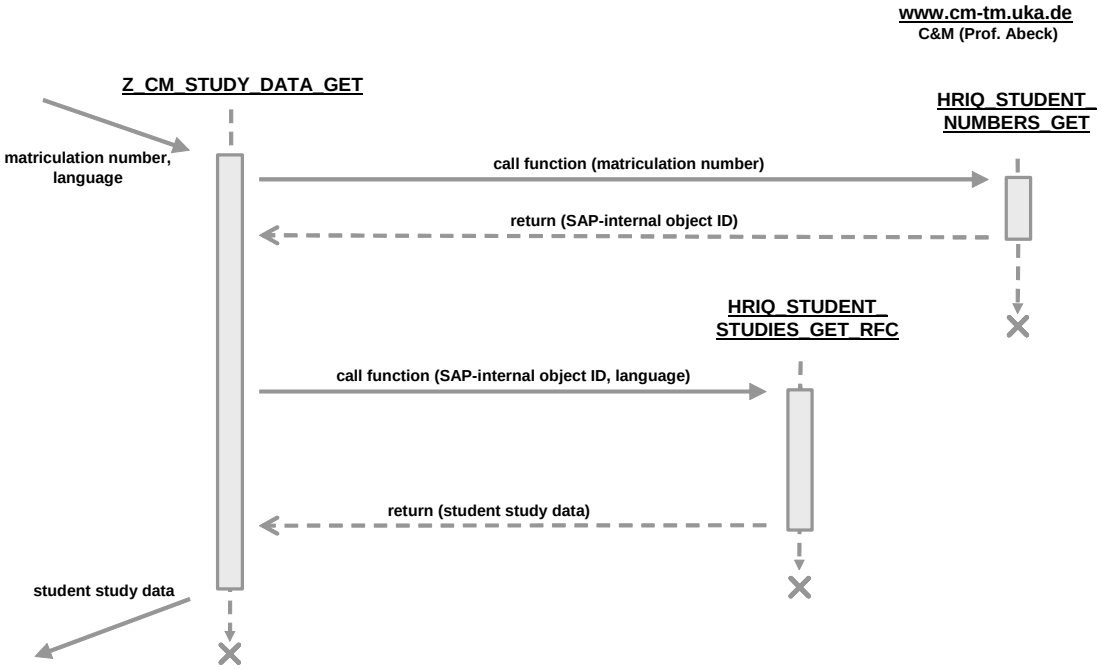
Information 42 shows the information flow within the function module Z_CM_IS_STUDENT which provides a check whether a student exists or not. This is done by calling the function

module HRIQ_STUDENT_NUMBERS_GET that maps the matriculation number of a student to a SAP-internal object ID (refer to Chapter 2.3.2 to get know what an object ID is). Thus Z_CM_IS_STUDENT does not really contain any own functionality, it simply encapsulates the functionality of HRIQ_STUDENT_NUMBERS_GET and makes it available for remote access.



Information 43: ABAP function module Z_CM_STUDENT_GET

The information flow of the second function module named Z_CM_STUDENT_GET is shown in Information 43. First the matriculation number as an input value is mapped to the SAP-internal object ID. Using this object ID, two BAPIs that deliver all standing data of a student are invoked and the adequate extract of their result data is returned.



Information 44: ABAP function module Z_CM_STUDY_DATA

Z_CM_STUDY_DATA provides the study-relevant data of a student. To accomplish this, the matriculation number is again translated into the object ID. Then the function module HRIQ_STUDENT_STUDIES_GET_RFC is invoked and the necessary data is returned. The concrete operational sequence is shown in Information 44 above.

As one can see, all three RFMs first use the same function module to map the matriculation number to the object ID. Then they use different function modules to implement their functionality. Hence they are independent and can be used alone/individually too.

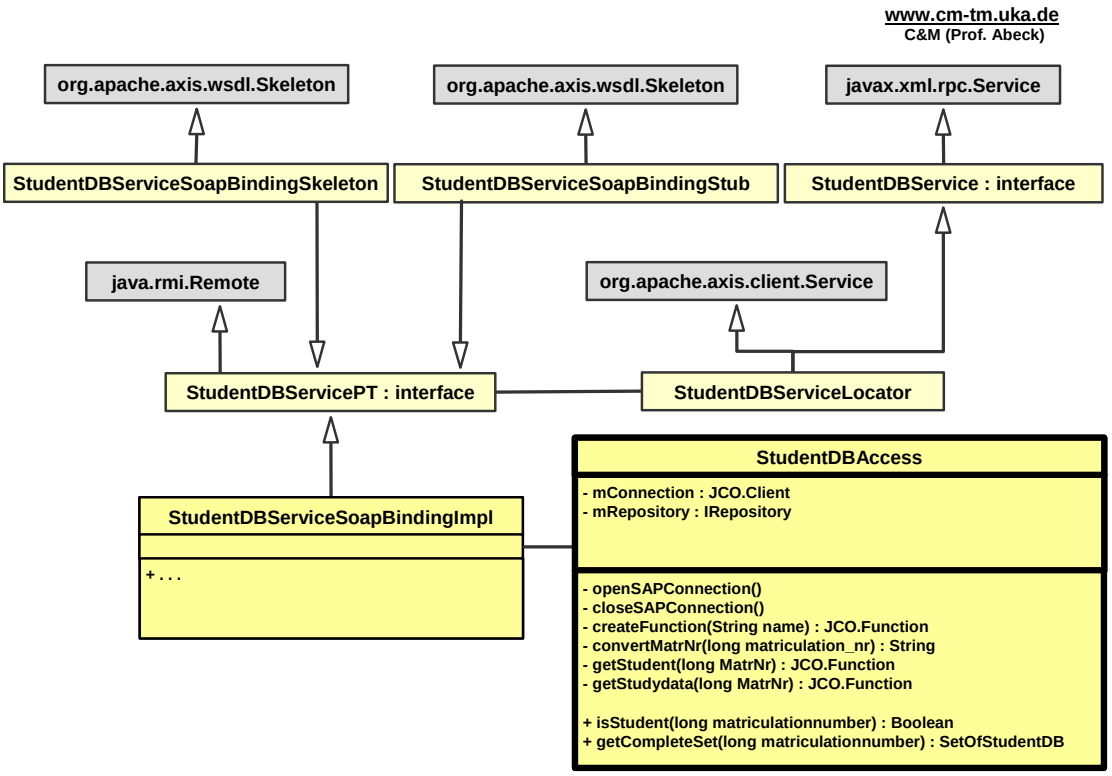
One important point during and after the implementation of the ABAP interfaces is the testing of their correctness. This can only be done by entering some test data into the accordant screens within the SAP GUI. Once these data are saved, the ABAP function modules can be tested by opening and running them in the Function Builder. As input values the just used test data could be taken. To also verify how the function modules react to incorrect input values, some non-existing data should be used. There is no other feasible way of testing the implementation of ABAP function modules that use existing functions of SAP CM.

All three function modules together have a total amount of about 80 LOC. Their source codes can be found in APPENDIX C.

3.1.2 Core Web services – JCo Web service

The implementation of the core Web services is achieved by using Java as the programming language. 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 a 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 WSDL2Java of the Apache Axis framework. This tool takes a description of a Web service written in WSDL and emits Java artifacts used to access the Web service [We05].
This procedure is done for all four core Web services. Three of them receive their data from underlying MySQL databases. APPENDIX D contains the SQL statements used to fill those databases. The fourth one takes the data from SAP CM using the JCo. It is named StudentDBService and was already designed in detail in Chapter 2. Information 45 displays the class model of this Web service. The other Web services use nearly the same class model; merely some classes are named differently. Thus Information 45 gives a good overview of the implementation of all four web services.



Information 45: Class Model of the StudentDBService

Only the class with the bold border, namely StudentDBAccess, has to be programmed manually. All other classes can be generated automatically by Apache Axis. The following three information slides depict some source code snippets taken from the StudentDBAccess class. Since it uses the JCo as its Middleware layer to interact with SAP CM, the focus will be put on the JCo-specific operations and classes.

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

(1) Source code to establish a connection to SAP Campus Management

```
import com.sap.mw.jco.*;                                   // Import JCo classes
[...]

private JCO.Client mConnection;                            // Define connection object
[...]

private void openSAPConnection() throws Exception {
        String clientSAP = "211";
        String login = "STUDENT";
        String passwort = "******";
        String lang = "DE";
        String server ="rzdb10.rz.uni-karlsruhe.de";
        String sysnum = "00";
        try     {
                System.out.print("Verbinde mit SAP ... ");
                mConnection = JCO.createClient (clientSAP, login, passwort, lang, server, sysnum);   // Create connection object
                mConnection.connect();
                System.out.println("verbunden !");                                                   // Connect this object to SAP CM
                System.out.println("---------------------------------");
        } catch (Exception e) {
                System.out.println(e.fillInStackTrace());
        }
}
```

Information 46: Source code snippet of StudentDBAccess – JCO.createClient()

Information 46 illustrates the method that establishes a connection to SAP CM. At first all JCo classes are imported by the import com.sap.mw.jco.* command. Then an object of the class JCO.Client that will later represent the connection to SAP CM is declared. Next, the connection parameters are set up with several variables. After this the connection object (named mConnection) is instantiated using the method JCO.createClient(). Once the object instance is configured with all parameters, the real connection establishment to the SAP system is invoked by the connect() method.

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

(1) Source code to create a JCo function template using the class JCO.Repository

```
private IRepository mRepository;                           // Define interface object
[...]

private JCO.Function createFunction(String name) throws Exception {
        try {
                mRepository = new JCO.Repository ("myRepository", mConnection);   // Create Repository object
                IFunctionTemplate ft = mRepository.getFunctionTemplate(name.toUpperCase());
                if (ft == null) return null;
                return ft.getFunction();                   // Build function template
        } catch (Exception ex) {                           // Get function object and return this object
                throw new Exception ("Das JCO.Function-Objekt kann nicht erzeugt werden.");
        }
}
```

Information 47: Source code snippet of StudentDBAccess – JCO.Repository

Now that the connection is built up, Information 47 shows the method implemented to create an object representing a function module within the SAP system. As the input parameter the name of this function module is passed and an object representing the function module within Java is returned when exiting the method. Explained step by step this means at first an interface object of the type IRepository that represents the complete SAP function repository is built by using the JCO.Repository class. This is necessary because the metadata of all RFMs one wants to use must be available to the JCo and this is accomplished by creating a JCO.Repository object. For sure the IRepository variable named mRepository in this case has to be previously declared. Then the repository interface is used by the method getFunctionTemplate() to get an IFunctionTemplate interface containing all the metadata (parameters and exceptions) for a specific RFM. The JCo retrieves the metadata only once and caches it to optimize performance. Last but not least a JCO.Function object is instantiated by the getFunction() method. This function object not only has the metadata, but also the actual parameters for the execution of the RFM. The relationship between a function template and a function in the JCo is similar to that between a class and an object in Java. The code shown above encapsulates the creation of a function object. Creating a new function object for each individual execution is very positive, thus building a separate method. This way it is assured that the parameters do not contain any leftovers from the previous call, like table rows that should not really be sent (back) to SAP.

(1) Source code for the method isStudent to check whether a student exists or not

```
public boolean isStudent(long matriculationnumber) throws Exception {
    boolean valid = false;
    try    {
        this.openSAPConnection();
        String rfm = "Z_CM_IS_STUDENT";
        JCO.Function isstudent = null;
        try {
            isstudent = this.createFunction(rfm);
        } catch (Exception ex) {   ex.printStackTrace(); System.exit(1);   }
        if (isstudent == null) {  System.exit(1);  }
        isstudent.getImportParameterList().setValue(
                this.convertMatrNr(matriculationnumber),"MATRNR");
        try {
            mConnection.execute(isstudent);
            String obj_id = isstudent.getExportParameterList().getString( "IS_STUDENT");
            String ref_obj_id = "00000000";
            if (! obj_id.equals(ref_obj_id) ) {   valid = true;   }
        } catch (JCO.Exception e) { }
        this.closeSAPConnection();
    }
    catch (Exception e) { System.out.println(e.fillInStackTrace()); }
    return valid;
}
```

Create function object

Set function parameters

Execute function

Get return value from SAP CM

Interpret SAP CM return value

Information 48: Source code snippet of StudentDBAccess – getImportParameterList() / getExportParameterList()

Information 48 displays the source code of the method isStudent() which implements the check whether a student exists in the SAP system or not. The method is one of the two public methods of the StudentDBService. First of all a connection to SAP R/3 is established with the usage of the method named openSAPConnection() that was already depicted and described in Information 46. Then an object instance for the function module Z_CM_IS_STUDENT is created using the method createFunction already explained above in detail. To set the parameters of the function module, the methods getImportParameterList() and setValue() are used. getImportParameterList() accesses the import parameter list, setValue() is used to set the value of a scalar parameter, passing the value

as the first, and the name as the second argument. Many overloaded versions of setValue() exist in the JCo, in order to support several data types. The JCo will do its best to convert any value passed to the data type appropriate for the field, and an exception is thrown if the conversion fails. To prevent an exception in this case, a method (namely convertMatrNr) is called receiving the matriculation number (which is the only parameter for Z_CM_IS_STUDENT) of type Long as input value and returning the same one as a set of eight characters which is necessarily for SAP CM. Afterwards the execute() command passing the function object as the parameter performs the remote procedure call. Once this is done, the return values can be accessed by the getExportParameterList() method in the same way this is done by getImportParameterList() for the import parameters. The method getString() receives the name of the export parameter of the RFM as input parameter and returns the concrete value. This concrete return value of Z_CM_IS_STUDENT is the SAP-internal object ID of the specified student. It is interpreted by comparing it with the zero-valued object ID. Getting true as a result means no student could be found. Otherwise, the student does exist in the system. As a last step the connection to SAP CM is closed.

Including all methods, the StudentDBService has a total amount of about 330 LOC. In comparison the other Web services each has less code because they do not need to implement the complex access to SAP CM. The complete source code for the StudentDBService can be found in APPENDIX B.

3.1.3 Composition Layer – The BPEL Process

In the design part for the BPEL process, the business process modeling of obtaining a ToR is depicted with all necessary steps. The decision for BPEL is exposed and further needed tools and the design are pointed out. Now the implementation of the BPEL code will be examined.

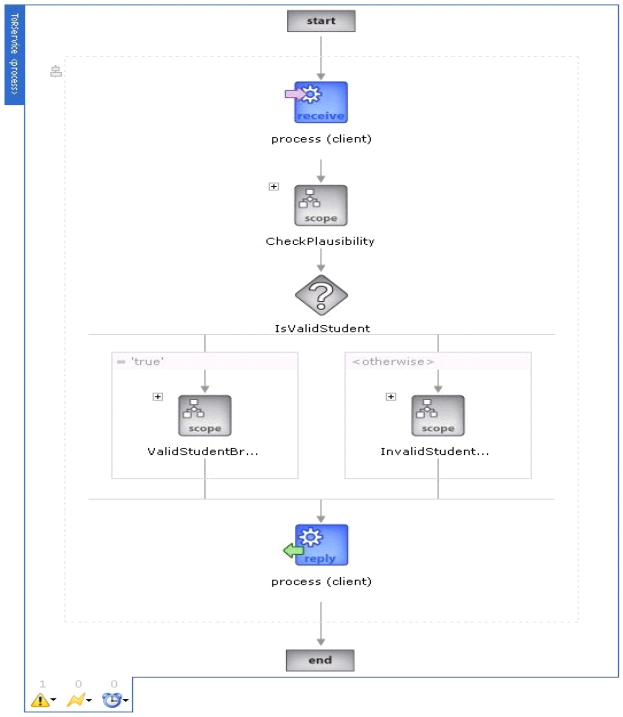
www.cm-tm.uka.de
C&M (Prof. Abeck)

```
<!-- 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>
```

Information 49: IMPLEMENTATION PHASE - BPEL Code for the ToRService

Information 49 depicts the 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 complete the BPEL code, which was generated out of BPMN (Information 29), and to make it executable, the Oracle BPEL Designer was used. The whole BPEL process is shown in Information 50 with its four scopes.



Information 50: The ToRService modelled with the Oracle BPEL Designer

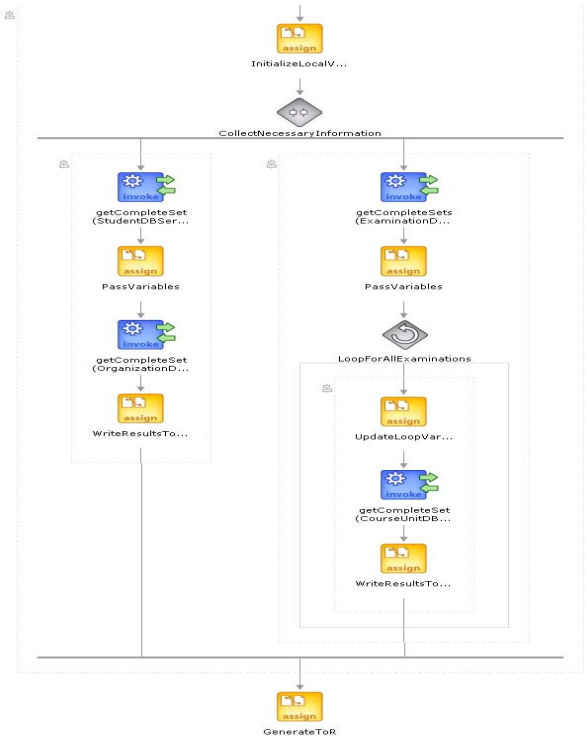
Scopes are helpful to handle errors individually in subject to the location they occur. An example for an easy error handling is shown in Information 51. The scope in which an error emerges is aborted. A catchAll causes the error value to be written to a specific variable.

Information 51: Example of a simple Error Handling

After receiving a ToR request from the portal, 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.

Information 52: The Scope in which the Plausibility Check is performed

As it can be seen in Information 52 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 executed and the necessary information is collected.



Information 53: 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 Information 29 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 respective faculty. After that both results are written to a global variable. Thus they can be used in other branches. The second sequence 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 [We05].

Concluding this scope a ToR has to be generated out of the gained information. At this point various approaches are possible. The first approach does 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. As a BPEL engine, which is needed to run this BPEL process, the Oracle Process Manager was installed. Its deployment and use is described later on.

3.1.4 Presentation Layer – Web based Portal as User Interface

The Design of the Portal already depicted the usability and reasons for using Jakarta Tomcat and Apache Axis. So far JSP supports the creation of dynamic Web pages that leverage existing

business systems. JSP technology separates the design of the interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content

Using some code fragments of the JSP pages, the way the JSP pages are used and how the implementation assists the claimed tasks will be explained. Not every detail of the implementation will be mentioned here to avoid redundancy and only an overview will be given.

Java sever pages (JSPs) are HTML pages with embedded Java code. When a client calls this HTML page this code is automatically transformed into servlets on the server-side. The byte code of this servlets then runs in the Java Virtual Machine (JVM). A servlet is similar to an applet that is executed on the server-sided. It extends the classes javax.servlet.Servlet or javax.servlet.http.HttpServlet and is executed in a servlet container that needs to be initialized and managed [RW05].

Important methods of a servlet are:

- doGet(): Method, e.g. processing HTTP requests from a client browser
- doPost(): Method, e.g. processing SOAP requests
- init(): Initializing the servlet, invoked by a servlet container
- destroy(): Method, to determine the servlet by the servlet container

Information 54 depicts the instantiation of a RequestDispatcher. RequestDispatcher defines an object that receives requests from the client and sends them to any resource. This can be a HTML file a servlet or a JSP file on the server. The object created by the servlet is used as a wrapper around a server resource located at a particular path or given by a particular name. The resource is defined by a leading forward slash that represents the root node.

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

```
(1) RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher("/student.jsp");
(2) dispatcher.forward(request, response);
```

Information 54: The Request Dispatcher

In our context the servlets can be used in various ways. First use is that of a forwarded value. Information 55 depicts the forwarding of the desired format the ToR will be supplied with.

```
<html>
<link rel="stylesheet" type="text/css" href="style.css">
<body>

<center><h1>Please select desired output format!
</h1></center>
<br><br>

<form action="displayXML.jsp" method="post">
<table border="0" cellpadding="8" cellspacing="8">
        <tr>
          <td><input type="radio" name="format" value="html"> HTML</td>
        </tr>
        <tr>
          <td><input type="radio" name="format" value="pdf"> Adobe Acrobat PDF</td>
        </tr>
        <tr>
          <td><input type="radio" name="format" value="ps"> Postscript PS</td>
        </tr>
        <tr>
          <td><input type="radio" name="format" value="plain"> Plain XML (Debug)</td>
        </tr>
        <tr>
          <td><input type="submit" value=" Submit "></td>
        </tr>
</table>
</form>

</body>
</html>
```

Information 55: Choosing the desired Format of the Transcript of Records

This value can be reused within the displayXML.jsp. The value is handed over as a string and responsible for the distinction of cases.


```
<%@ page language="java" contentType="text/html" %>
<%@ page import="javax.xml.transform.*"%>
<%@ page import="javax.xml.transform.stream.*"%>
<%! String FS = System.getProperty("file.separator"); %>
<%
String format = request.getParameter("format");

if (format.equals("pdf")) {
        […]

} else if (format.equals("html")) {
//set files to use
String xmlFile    = "ToR.xml";
String xslFile    = "ToRTemplate.xsl";

// get the real path for xml and xsl files;
        String ctx = getServletContext().getRealPath("") + FS;
        xslFile = ctx + xslFile;
        xmlFile = ctx + xmlFile;

// transform to HTML
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer(new StreamSource(xslFile));
transformer.transform(new StreamSource(xmlFile), new StreamResult(out));

} else {
   […]

}
%>
```

Information 56: Converting XML into HTML

Furthermore another use of JSPs is depicted in Information 56. With the use of the packages javax.xml.transform and javax.xml.transform.stram generic APIs for processing transformation

instructions and performing a transformation from source to result are defined. For transforming the XML file of the ToR a new TransformerFactory is instantiated.
Information 57 depicts an exemplary XML file used for conversion. Some passages are left out because only the structure is of importance and the complete listing would hamper clarity.


```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ToR.xml -->

<TranscriptOfRecords xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ToRSchema.xsd">
        <!-- Information about the Student and the University -->
        <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>
                        <InstitutionName>University of Strassbourg</InstitutionName>
                        <FacultyDepartmentName>Computer Science</FacultyDepartmentName>
                        <ECTSDepartmentalCoordinator>
                                <Surname>Binoche</Surname>
                                <GivenName>Juliette</GivenName>
                                <Telephone>+33 01337439520</Telephone>
                                <Fax>+33 01337439520</Fax>
                                <EMail>ects@strassbourg.fr</EMail>
                        </ECTSDepartmentalCoordinator>
           </ReceivingInstitution>
[…]
        </Header>
</TranscriptOfRecords>
```

Information 57: An abstract of an exemplary Transcript of Records in XML

Information 58 depicts the XSL file structuring the output of the factory. Due to this XML the HTML file is generated to yield the ToR. A showcase for the dynamic code generation is figured by the xsl:for-each statement. For each block in the XML file a row of a HTML table is generated containing the necessary columns. Every value of the XML document can be addressed by its corresponding XPath expression.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">

<html>
<head>
<title>ECTS - EUROPEAN CREDIT TRANSFER</title>
</head>
<body>
[…]

<tr> <td colspan="3"><b>NAME OF SENDING INSTITUTION: </b> <xsl:value-of
      select="//Header/SendingInstitution/InstitutionName" /></td></tr>
[…]
<xsl:for-each select="//TableOfGrades/CourseUnit">
<tr>
      <td height="22"><xsl:value-of select="CourseUnitCode"/></td>
      <td height="22"><xsl:value-of select="CourseTitle"/></td>
      <td height="22"><xsl:value-of select="CourseDuration"/></td>
      <td height="22"><xsl:value-of select="LocalGrade"/></td>
      <td height="22"><xsl:value-of select="ECTSGrade"/></td>
      <td height="22"><xsl:value-of select="ECTSCredits"/></td>
</tr>
</xsl:for-each>
[…]
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Information 58: Adressing parts of the XML Transcript of Records via XPath Expressions

So far the XML file can be transformed into HTML. Obtaining this XML file is solved via Web services. Information 59 depicts how to invoke a Web service inside the Jakarta Tomcat servlet container.


```
<html>
<body>

<%@ page import = "org.apache.axis.client.Call" %>
<%@ page import = "org.apache.axis.client.Service" %>
<%@ page import = "org.apache.axis.encoding.XMLType" %>
<%@ page import = "org.apache.axis.utils.Options" %>
<%@ page import = "org.apache.soap.rpc.Response" %>

<%@ page import = "javax.xml.rpc.ParameterMode" %>

<%   String endpoint = "http://localhost:9700/orabpel/default/ToRService/1.6.8";
      out.println(endpoint);
      Service service = new Service();
      Call call = (Call) service.createCall();
      call.setTargetEndpointAddress(new java.net.URL(endpoint));
      call.setOperationName("ToRService");
      call.addParameter("matriculationNumber",XMLType.XSD_LONG,ParameterMode.IN);

      call.setReturnType(XMLType.XSD_ANYTYPE);

      Object params[] = new Object[1];
      params[0]=(Object) new Long("2");

      Object[] ret = call.invoke(params);

      out.println("result: " + ret);

%>
</body>
</html>
```

Information 59: JSP code for invoking the ToR Service

The SOAP classes of Apache Axis are used to communicate with Web services. The import of these classes is depicted as well as the usage. First the endpoint that specifies the used Web service is defined. An instance of a service is created and the endpoint is assigned. Further steps are to name and specify the operation, parameters and return types.

After that, the method ToRService can be invoked.

3.2 Deployment

This chapter describes how to install the ToRService and all obligatory components and Web services on a system with Microsoft Windows XP Home / Professional Edition. A CD-ROM containing all source code and other files that were addressed in this document can be found enclosed with this document. The term $CD-ROM embodies the absolute path to the folder where the content of this CD-ROM is located. This CD is structured as follows:

- The folder Documents contains this document, the corresponding slides, two technology documents explaining ABAP and the JCo and other continuative material.
- The directories java respective bpel contain the core Web services and the ToRService.
- The gui_tor directory embodies all JSP files building the USOA portal as well as the XSLT template for the ToR.
- In the dbs folder the files to create the three databases are included.
- The config directory includes several configuration files e.g. for Tomcat that had to be edited to run the ToRService.

3.2.1 Software Requirements

To run the ToRService and the USOA Portal 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 4.1.10a)
- BPEL Process Manager from Oracle (Version 2.1.1)
- SAP Java Connector (Version 2.1.5)
- Microsoft .NET Framework (Version 1.1)

The software can be found in the folder $CD-ROM\External-Software.

3.2.2 Deployment of the ABAP Interfaces

Software	Used version
Java J2SE JDK	1.4.2
SAP Java Connector	2.1.5
Microsoft .NET Framework	1.1

Table 1: Used software

This subchapter deals with the installation procedure of the ABAP interfaces. Because the ABAP function modules are built in the ABAP Workbench which is directly located in the SAP system, there is no real deployment required. Within SAP CM the function modules are already deployed. But on the remote machines accessing those function modules some additional software needs to be installed.

Basic requirement for the usage of the JCo is the installation of an actual Java JDK, preferably version 1.4.2. The program can be found at $CD-ROM\External_Software\j2sdk_1.4.2-07_windows-i586-p.exe. This can be done very easily using the built-in installer program. C:\j2sdk1.4.2_07\ is used as the installation folder in this document. Additionally this directory has to be set as JAVA_HOME in the environment variables.

The JCo itself can be downloaded at the SAP Marketplace under service.sap.com. It is best to always use the newest version of the JCo. The given installation guideline only refers the version 2.1.5. The corresponding installation file can be found at $CD-ROM\External_Software\sapjco-ntintel-2.1.5.zip. Since this version, under Windows the installation of the Microsoft .NET Framework is required. The easiest way to do this is the Microsoft Windows Update function. Therein the selection Optional Software Updates offers the necessary installation files. Unzip sapjco-nintel-2.1.5.zip to C:\sapjco. This folder contains two DLL files named librfc32.dll and sapjcorfc.dll. They must be copied into the directory C:\Windows\System32 (or analogous).

For testing purposes, set the environment variables to CLASSPATH=.;C:\sapjco\sapjco.jar and PATH=%PATH%;C:\j2sdk1.4.2_07\bin;C:\sapjco. Now a first connection to the SAP R/3 can be established. Therefore edit the C:\sapjco\demo\Example1.java. Compile the File with javac Example1.java and run javac Example1.

3.2.3 Deployment and Installation of the JCo Core Web services

Software	Used version
MySQL Database Server	4.1.10a
Apache Jakarta Tomcat	5.0.28
Apache Axis	1.1

Table 2: Used software

The deployment of the four core Web services as well as the installation of necessary software components are subject of this subchapter.

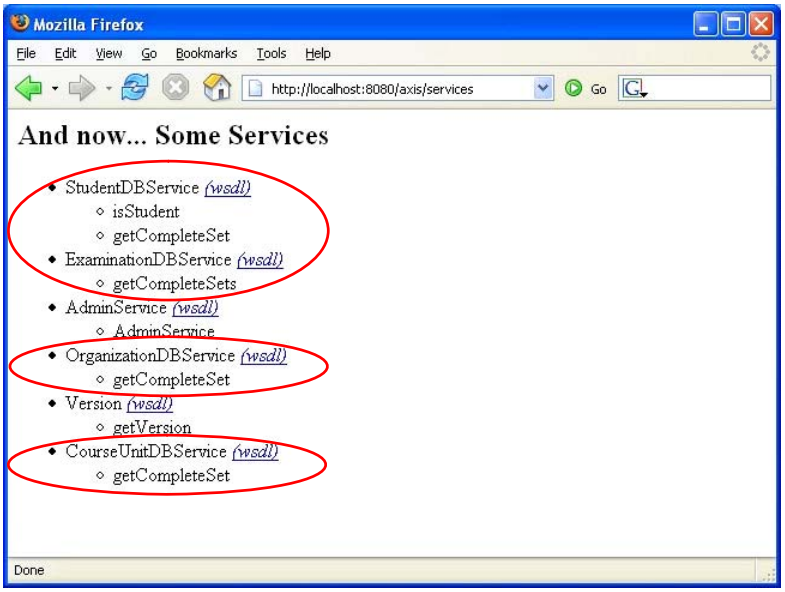
The start can be done with the necessary databases. After downloading the MySQL Database Server at dev.mysql.com/downloads or using the package $CD-ROM\External_Software\mysql-noinstall_4.1.10a-win32.zip it can be unpacked to the desired installation directory. In this document the installation path 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 three databases of Chapter 3.1.2 and fills in some test values: mysql < $CD-ROM\USOA\tor\dbs\create_dbs.sql [We05].

Now the Apache Jakarta Tomcat installer can be downloaded under jakarta.apache.org or copied from $CD-ROM\External_Software\jakarta-tomcat_5.0.28.exe. As 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 before. 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 $CD-ROM\USOA\tor\java\libs should be placed in C:\tomcat\common\lib.

After downloading Apache Axis under ws.apache.org/axis or using $CD-ROM\External_Software\axis_1.1.zip, 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 have 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. To be able to use the JCo within Axis, the package sapjco.jar also has to be copied to C:\tomcat\webapps\axis\WEB-INF\lib. Last but not least the value of the parameter sendMultiRefs must be set to false in the configuration file C:\tomcat\webapps\axis\WEB-INF\server-config.wsdd to enable good communication with the BPEL engine. This configuration file can be found in $CD-ROM\USOA\tor\config\tomcat\webapps\axis\WEB-INF and has to be copied to the appropriate location.

The next step will be the deployment of the four core Web services including the one JCo Web service. A detailed description how to deploy Web services to Axis in general can be found under [RW05]. The Web services themselves are located at $CD-ROM\USOA\tor\java\classes\de\cm\services. Using the corresponding deployment and undeployment descriptors deploy.wsdd and undeploy.wsdd, they can be easily deployed into Axis. This can be done by changing to the folder C:\tomcat\webapps\axis\WEB-INF\classes and entering the command java -cp "%AXISCLASSPATH%" org.apache.axis.client.AdminClient –l http://localhost:8080/axis/services/AdminService deploy.wsdd inside the shell. To deploy the four Web services in one step, the usage of the batch file $CD-ROM\USOA\tor\java\classes\deploy.bat is recommendable. Before doing so, the content of the folder $CD-ROM\USOA\tor\java\classes has to be copied to a special tomcat directory, namely C:\tomcat\webapps\axis\WEB-INF\classes.
After a successful deployment of the core Web service, their access points within Axis should be seen when typing http://localhost:8080/axis/services into the browser.



Information 60: The deployed four 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
- http://localhost:8080/axis/services/OrganisationDBServiceP

To start the RemoteClient and test the core Web services the batch file $CD-ROM\USOA\tor\RunRemoteClient.bat has to be executed into a shell and in the property file at $CD-ROM\USOA\tor\java\src\de\cm\clients, the service URLs of the core Web services has to be set correctly. Furthermore all the request parameters can be adjusted in this property file [We05].

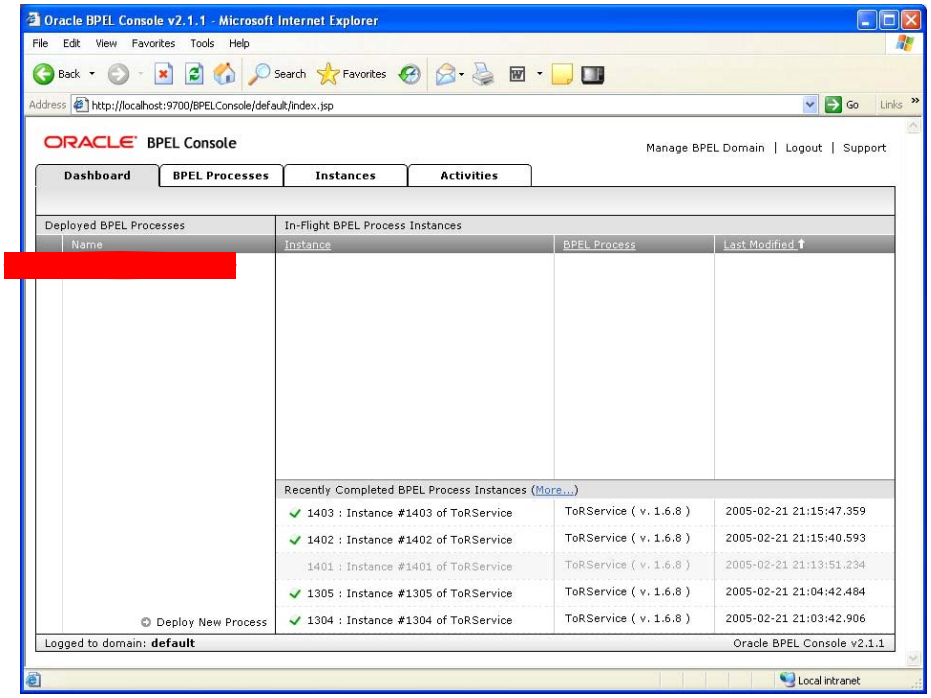
3.2.4 Deployment of the BPEL process

Software	Used version
Oracle BPEL Process Manager	2.1.1

Table 3: Used software

Now that the core Web services are running, the installation of the ToRService can begin. After downloading the Oracle Process Manager at http://www.oracle.com or using the file $CD-ROM\External_Software\orabpel_2.1.1_win32.exe 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 Server 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 $CD-ROM\USOA\tor\bpel\bpel_ToRService_1.6.8.jar deploys the ToRService.

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



Information 61: The deployed ToRService in the Oracle Process Manager

The service URL is http://localhost:9700/orabpel/default/ToRService.
Concluding it should be mentioned that the ToRService can of course be run on any other BPEL engine too. But this requires adaptations in the package structure and respectively in the deployment descriptor [We05].

3.2.5 Deployment of the Portal

Software	Used version
Java J2SE JDK	1.4.2
Apache Jakarta Tomcat	5.0.28

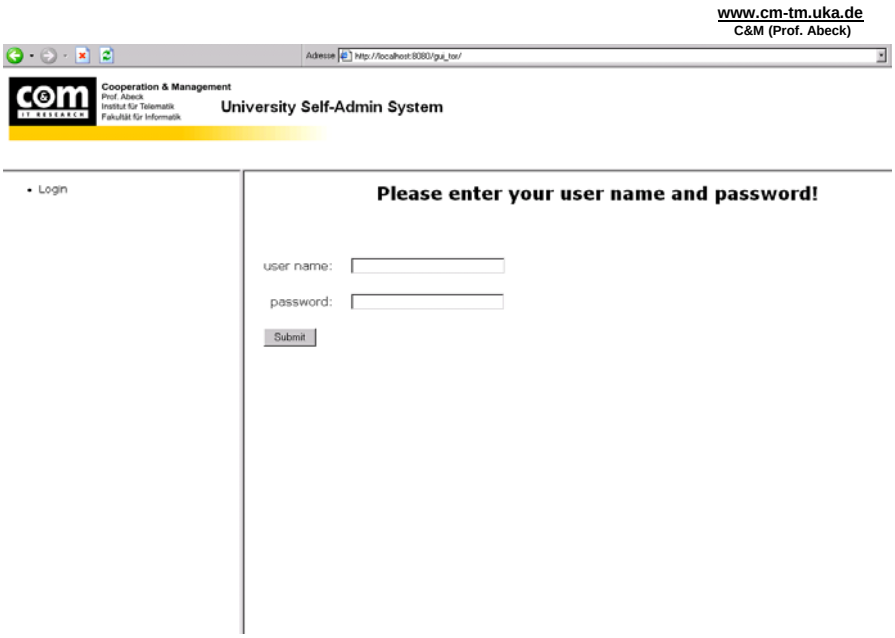
Table 4: Used software

Installing the JDK has been described in Chapter 3.2.3. In this case it is assumed that the Apache Jakarta Tomcat and the Web services are running on the same machine. Thus the JDK needed for the Apache Jakarta Tomcat is already available. Furthermore a servlet container is needed to run Apache Axis and support the JCo. Therefore the Apache Jakarta Tomcat is installed using the description in the mentioned chapter. Obviously the portal could be installed on a different computer. For simplification reasons and the facility of inspection it has been chosen to install this software on the same computer.

To use the portal the implementation of the GUI is necessary. The JSP pages of the GUI, found at $CD-ROM\USOA\tor\gui_tor, have to be moved to the Tomcat directory C:\tomcat\webapps\. Now the GUI is accessible by entering http://localhost:8080/gui_tor into a browser. The login screen should appear.

3.3 Usage

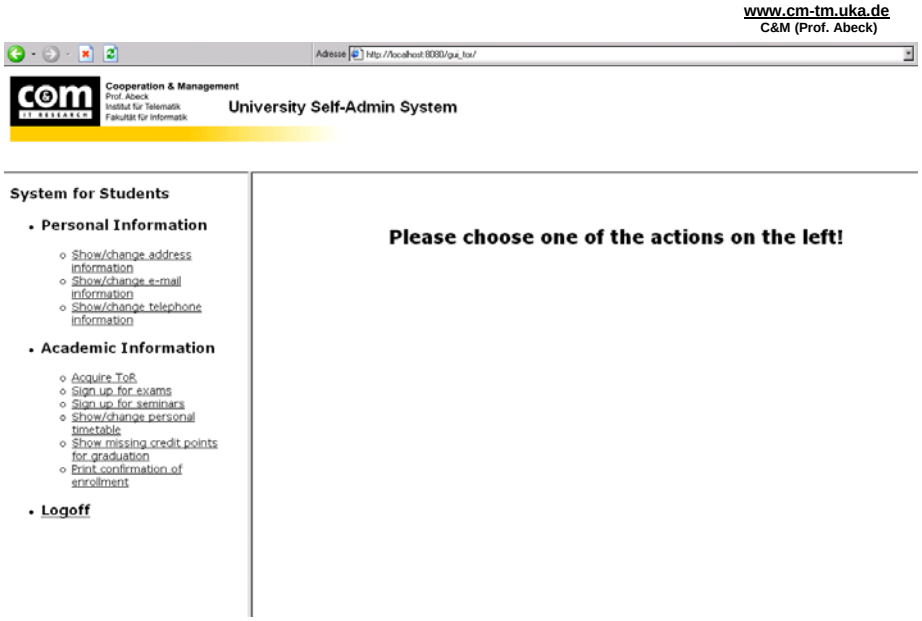
This chapter describes the usage of the GUI and how to obtain the ToR as a user.
Apache Jakarta Tomcat can be accessed via port 8080. So every user with permission to access this port on the computer can browse to http://IP:8080/gui_tor where IP is the IP number of the computer. Information 62 depicts the login screen of the deployed GUI. This is the first page the user is confronted with.



Information 62: Login Screen of the deployed GUI

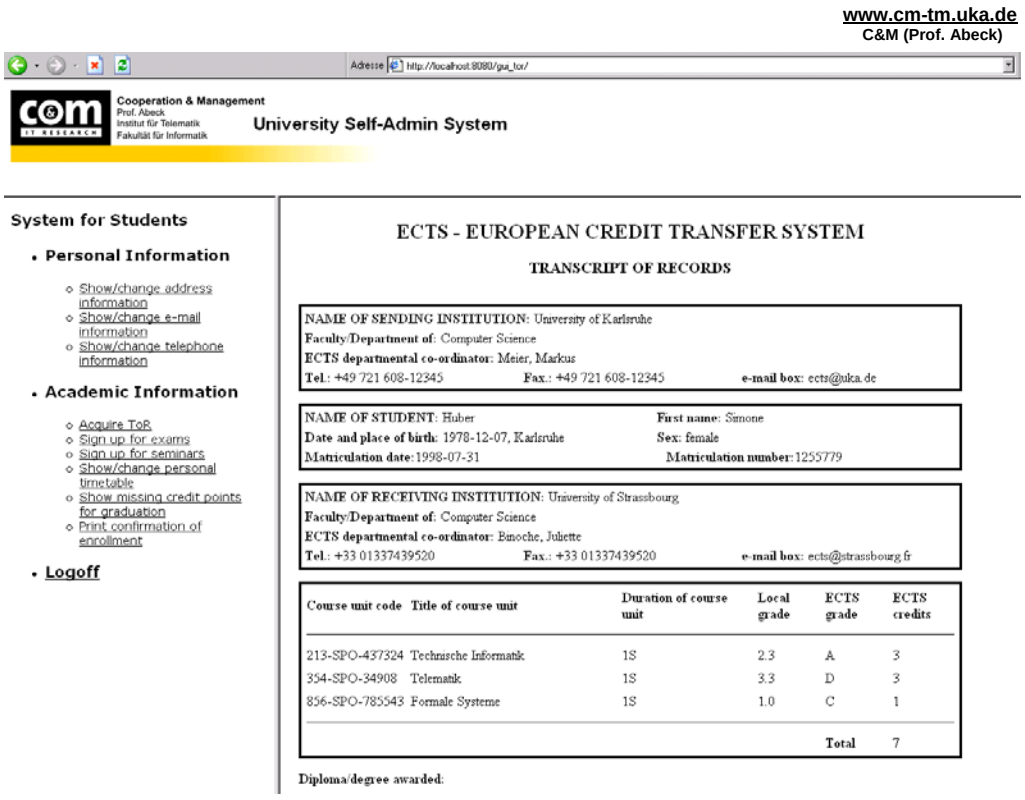
After accessing the GUI with a login name and password, the menu of the GUI appears. To view the possibilities for a student within the GUI, the login name has to be student_test. To view the staffs' possibilities the login name staff_test must be chosen. These accounts are for

testing purpose only. Information 63 depicts the students' menu on the left hand side and a frame for further content on the right. The top frame is for representation and identification.



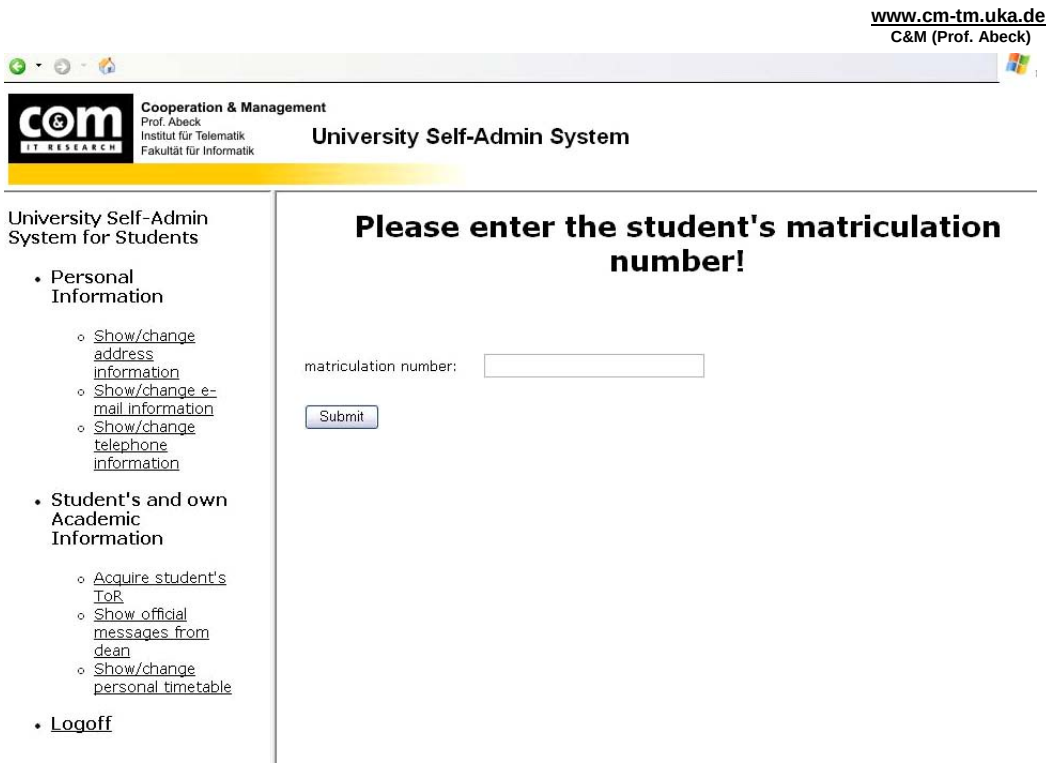
Information 63: Menu of the GUI

By clicking Acquire ToR the desired output format can be chosen. The following information slide depicts the HTML version of the ToR.



Information 64: The HTML Transcript of Records within the GUI

Some different functionalities appear logging in as staff. By entering a matriculation number it is possible to view a ToR for a specific student. Obviously this function is not available for a student.



Information 65: Further Options for Staff

4 OUTLOOK

The goal of this document was to depict the integration of a legacy university resource planning system (URP) into a Service-Oriented Architecture (SOA). This goal was attained by adding new Web service interfaces to SAP CM that were orchestrated in the SOA using BPEL. Furthermore additional functionality was developed to support the provision of a so-called Bologna-conforming ToR. The focus was placed on the methodology of the integration process. As a further step, a central university portal was developed to handle user interaction.

The SOA as described before facilitates the appropriation of various functionalities deriving from different software systems without confronting a user with specific characteristics of each system. Because different functions are used, the user does not need to login multiple times and a single-sign-on strategy is obtainable. Additionally the installation of further client programs can be omitted.

Thus the first step for a service-oriented integration process is completed and the portal can be arbitrarily enriched with further functionality. In the course of this modularization, the hierarchical structure and the business-oriented point of view, single entities can be easily exchanged, updated and maintained.

The following central step that is indispensable within live operation is the identity management. Numerous systems in a heterogeneous software environment come with their own user directory. The user directories are often different from system to system and rarely homogeneous. Most of them pursue different approaches in Rights Management, Access Management and all too often they also cover special function related tasks. Having a SOA in mind there is a lack of a methodology to centralize the access to the user directory. The question remains whether this identity management is to be rolled out and thereby be completely placed in the SOA, whether the authentication is always bounded to the communication between the SOA and the legacy system or as a third possibility a combination is utilized.

APPENDIX A WSDL schema of a core Web service

```
<?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 B Java Source Code

```
/**
 * StudentDBAccess.java
 * @author Heiko.Schandua@cm-tm.uka.de
 */
package de.cm.services.stud;


import com.sap.mw.jco.*;


public class StudentDBAccess {

	private JCO.Client mConnection;
	private IRepository mRepository;

	// Method, which establishes the SAP connection
	private void openSAPConnection() throws Exception {
		String clientSAP = "211";
		String login = "STUDENT13";
		String passwort = "TEAM13";
		String lang = "DE";
		String server ="rzdb10.rz.uni-karlsruhe.de";
		String sysnum = "00";
		try	{
			System.out.print("Verbinde mit SAP ... ");
			mConnection = JCO.createClient (clientSAP, login, passwort, lang,
                                                        server, sysnum);
			mConnection.connect();
			System.out.println("verbunden !");
			System.out.println("---------------------------------");

		} catch (Exception e) { System.out.println(e.fillInStackTrace()); }

	}

	// Method, which closes the SAP connection
	private void closeSAPConnection() throws Exception {
		try {
			mConnection.disconnect();
			System.out.println("---------------------------------");

			System.out.println("Verbindung mit SAP getrennt !");
		} catch (Exception ex) { ex.printStackTrace(); System.exit(1); }
	}

	// JCo Method which creates a function Template
	private JCO.Function createFunction(String name) throws Exception {
		try {
			mRepository = new JCO.Repository ("myRepository", mConnection);
			IFunctionTemplate ft =
                                mRepository.getFunctionTemplate(name.toUpperCase());

			if (ft == null) return null;
			return ft.getFunction();
		} catch (Exception ex) {
			throw new Exception ("Das JCO.Function-Objekt kann nicht erzeugt
                                        werden.");
		}
	}

	// Method for the type conversion of the matriculation number long to char[8]
	private String convertMatrNr (long matriculation_nr) {
		String matn = "" + matriculation_nr;
		while (matn.length() < 8) {
			matn = "0" + matn;
		}
		if (matn.length() > 8) {
			return "00000000";
		} else {
			return matn;
		}
	}

	// Method, which returns the function object containing the standing data
	private JCO.Function getStudent(long MatrNr) {
		String rfm = "Z_CM_STUDENT_GET";
		JCO.Function function = null;
		try {
			function = this.createFunction(rfm);
		} catch (Exception ex) {
			ex.printStackTrace();
```

```
                System.exit(1);
        }
        if (function == null) {
                System.err.println (rfm + " nicht gefunden in SAP");
                System.out.println("---------------------------------");

                System.exit(1);
        }

function.getImportParameterList().setValue(this.convertMatrNr(MatrNr),"MATRNR");
        try {
                mConnection.execute(function);
                System.out.println(rfm + " gefunden in SAP");
                System.out.println("---------------------------------");
        } catch (Exception e) {}
        return function;
}

// Method, which returns the function object containing the study data
private JCO.Function getStudydata(long MatrNr) {
        String rfm = "Z_CM_STUDY_DATA_GET";
        JCO.Function function = null;
        try {
                function = this.createFunction(rfm);
        } catch (Exception ex) {
                ex.printStackTrace();
                System.exit(1);
        }
        if (function == null) {
                System.err.println (rfm + " nicht gefunden in SAP");
                System.out.println("---------------------------------");

                System.exit(1);
        }

function.getImportParameterList().setValue(this.convertMatrNr(MatrNr),"MATRNR");
        try {
                mConnection.execute(function);
                System.out.println(rfm + " gefunden in SAP");
                System.out.println("---------------------------------");
        } catch (Exception e) {}
        return function;
}

// Method, which returns true if the student exists in the system
public boolean isStudent(long matriculationnumber) throws Exception      {

        boolean valid = false;
        try     {
                this.openSAPConnection();
                String rfm = "Z_CM_IS_STUDENT";
                JCO.Function isstudent = null;
                try {
                        isstudent = this.createFunction(rfm);
                } catch (Exception ex) {
                        ex.printStackTrace();
                        System.exit(1);
                }
                if (isstudent == null) {
                        System.err.println (rfm + " nicht gefunden in SAP");
                        System.out.println("---------------------------------");

                        System.exit(1);
                }
                isstudent.getImportParameterList().setValue(
                      this.convertMatrNr(matriculationnumber),"MATRNR");
                try {
                        mConnection.execute(isstudent);
                        System.out.println(rfm + " gefunden in SAP");
                        System.out.println("---------------------------------");
                        String obj_id =
                            isstudent.getExportParameterList().getString(
                            "IS_STUDENT");
                        String ref_obj_id = "00000000";
                        if (! obj_id.equals(ref_obj_id)) {
                                valid = true;
                        }
                } catch (JCO.Exception e) {
                }
                this.closeSAPConnection();
        }
        catch (Exception e) { System.out.println(e.fillInStackTrace()); }
        return valid;
}
```

```
        // Method for the filling of the complete resultset of this service class
        public SetOfStudentDB getCompleteSet(long matriculationnumber) throws Exception
{

                SetOfStudentDB completeset = null;
                try     {
                        this.openSAPConnection();
                        // Standing data ermitteln
                        JCO.Function studentGet = getStudent(matriculationnumber);
                        JCO.Structure stdata =
                           studentGet.getExportParameterList().getStructure(
                           "STUDENT_BASIC_DATA");
                        JCO.Structure adddata =
                           studentGet.getExportParameterList().getStructure(
                           "STUDENT_ADDRESS_DATA");

                        SetOfStandingTable standingset = new SetOfStandingTable();
                        standingset.setMatriculationNumber(matriculationnumber);
                        standingset.setGivenName(stdata.getString("FIRST_NAME"));
                        standingset.setSurName(stdata.getString("LAST_NAME"));
                        if (stdata.getString("GENDER_KEY").equals("1")) {
                                standingset.setSex(true);
                        } else if (stdata.getString("GENDER_KEY").equals("2")) {
                                standingset.setSex(false);
                        }
                        standingset.setDateOfBirth(stdata.getDate("DATE_BIRTH"));
                        standingset.setPlaceOfBirth(stdata.getString("BIRTHPLACE"));
                        standingset.setStreet(adddata.getString("STREET"));
                        try {
                                standingset.setHouseNumber(adddata.getInt("HOUSE_NO"));
                        } catch (Exception e) {}
                        standingset.setCity(adddata.getString("CITY"));
                        try {
                                standingset.setCodePostale(adddata.getInt("POSTL_COD1"));
                        } catch (Exception e) {}
                        standingset.setNationality(stdata.getString("NATIONALITY_TXT"));

                        // Study data ermitteln
                        SetOfStudyTable studyset = new SetOfStudyTable();
                        JCO.Function studydataGet = getStudydata(matriculationnumber);
                        JCO.Table studydata =
                                  studydataGet.getTableParameterList().getTable(
                                  "STUDIENGANGSDATEN");
                        if (studydata.getNumRows() > 0) {
                                studyset.setMatriculationNumber(matriculationnumber);
                                studyset.setDateOfMatriculation(
                                                studydata.getDate("BEG_KEY_DATE"));
                                studyset.setFieldOfStudy(
                                                studydata.getString("PROGRAM_TXT"));
                                studyset.setAward("");
                        }
                        this.closeSAPConnection();
                        completeset = new SetOfStudentDB();
                        completeset.setStandingData(standingset);
                        completeset.setStudyData(studyset);
                } catch (Exception e) { System.out.println(e.fillInStackTrace()); }
                return completeset;
        }

}
```

APPENDIX C ABAP Source Code

Function module Z_CM_IS_STUDENT:

```
FUNCTION Z_CM_IS_STUDENT.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(MATRNR) TYPE  PIQSTUDENT12
*"  EXPORTING
*"     VALUE(IS_STUDENT) TYPE  OBJEKTID
*"----------------------------------------------------------------------

DATA:    z_objid type OBJEKTID.

*---- Rufe FB auf, um aus Matr-Nr. die SAP interne ObjektID zu ermitteln
    CALL FUNCTION 'HRIQ_STUDENT_NUMBERS_GET'
         EXPORTING
              iv_student12   = MATRNR
         IMPORTING
              ev_objid     = IS_STUDENT
         EXCEPTIONS
              OTHERS = 1.

ENDFUNCTION.
```

Function module Z_CM_STUDENT_GET:

```
FUNCTION Z_CM_STUDENT_GET.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(MATRNR) TYPE  PIQSTUDENT12
*"     VALUE(LANGUAGE) TYPE  BAPI_LAISO-LANGU_ISO DEFAULT 'EN'
*"  EXPORTING
*"     VALUE(STUDENT_BASIC_DATA) TYPE  BAPISTUDENT_PERSONALT
*"     VALUE(STUDENT_ADDRESS_DATA) TYPE  BAPISTUDENTADDRESST
*"----------------------------------------------------------------------

DATA:    z_objid type OBJEKTID.

*---- Rufe FB auf, um aus Matr-Nr. die SAP interne ObjektID zu ermitteln
    CALL FUNCTION 'HRIQ_STUDENT_NUMBERS_GET'
         EXPORTING
              iv_student12   = MATRNR
         IMPORTING
              ev_objid     = z_objid.


*---- Aufrufen des BAPI zur Ermitteln der Studentenstammdaten
    CALL FUNCTION 'BAPI_STUDENT_GETDETAIL3'
         EXPORTING
              objectid     = z_objid
              read_texts   = 'X'
              language_iso = LANGUAGE
         IMPORTING
              studentpersonaldatat  = STUDENT_BASIC_DATA.

*---- Aufrufen des BAPI zur Ermitteln der Studentenadressdaten
    CALL FUNCTION 'BAPI_STUDENT_ADDRESS_GETDETAIL'
         EXPORTING
              objectid     = z_objid
              read_texts   = 'X'
              language_iso = LANGUAGE
         IMPORTING
              addressdatat  = STUDENT_ADDRESS_DATA.

ENDFUNCTION.
```

Function module Z_CM_STUDY_DATA_GET:

```
FUNCTION Z_CM_STUDY_DATA_GET.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(MATRNR) TYPE  PIQSTUDENT12
*"     VALUE(LANGUAGE) TYPE  BAPI_LAISO-LANGU DEFAULT 'EN'
```

```
*"  TABLES
*"      STUDIENGANGSDATEN STRUCTURE  PIQRFC_STUDYSEGMENTS_TXT OPTIONAL
*"----------------------------------------------------------------------

DATA:    z_objid type OBJEKTID,
         plan type PLVAR.

    plan = '01'.

*---- Rufe FB auf, um aus Matr-Nr. die SAP interne ObjektID zu ermitteln
    CALL FUNCTION 'HRIQ_STUDENT_NUMBERS_GET'
         EXPORTING
              iv_student12   = MATRNR
         IMPORTING
              ev_objid     = z_objid.


*---- Aufrufen des RFC FB zur Ermitteln der Studiendaten des Studenten
    CALL FUNCTION 'HRIQ_STUDENT_STUDIES_GET_RFC'
         EXPORTING
              objectid     = z_objid
              planversion  = plan
              read_texts   = 'X'
              read_studysegments = 'X'
              language = LANGUAGE
         TABLES
              studysegments_text = STUDIENGANGSDATEN.


ENDFUNCTION.
```

APPENDIX D SQL Statements for MySQL databases

```
CREATE DATABASE organizationdb;

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 VARCHAR(32),
       ectscoordinator_fax VARCHAR(32),
       ectscoordinator_email VARCHAR(64),
       PRIMARY KEY (universityname, facultyname)
);

INSERT INTO universitydata VALUES ('Universitaet Karlsruhe
(TH)','Karlsruhe','76128','Deutschland');
INSERT INTO universitydata VALUES ('Universita di Bologna','Bologna','40126','IT');

INSERT INTO facultydata VALUES ('Universitaet Karlsruhe
(TH)','Informatik','Billy','Gates','+49 721 608-3248','+49 721 608-
3248','ects@ira.uka.de');
INSERT INTO facultydata VALUES ('Universitaet Karlsruhe (TH)','Master of Information
Science','Christian','Emig','+49 721 608-3248','+49 721 608-3248','emig@ira.uka.de');
INSERT INTO facultydata VALUES ('Universitaet Karlsruhe
(TH)','Staatswissenschaft','Hans','Muentefering','+49 721 608-8432','+49 721 608-
8432','muenty@ira.uka.de');
INSERT INTO facultydata VALUES ('Universitaet Karlsruhe
(TH)','Sport','Joey','Deckarm','+49 721 608-4376','+49 721 608-4376','joey.deckarm@uni-
karlsruhe.de');



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 ('45-SPO-1234','Wettskandale','2S','4');
INSERT INTO courseunitdata VALUES ('424-SPO-965','Hallenhalma','2T','2');
INSERT INTO courseunitdata VALUES ('832-SPO-935','Rasenschach','1T','1');
INSERT INTO courseunitdata VALUES ('872-SPO-5335','Selbstvermarktung','1S','5');

INSERT INTO courseunitdata VALUES ('893-SWI-543','Revolutionslehre','2T','4');
INSERT INTO courseunitdata VALUES ('854-SWI-2783','Praktische Putschtheorie
2','1T','4');
INSERT INTO courseunitdata VALUES ('125-SWI-7247','Die Neue Mitte','2S','6');
INSERT INTO courseunitdata VALUES ('362-SWI-685','Diaetenmanagement','1S','2');
INSERT INTO courseunitdata VALUES ('27-SWI-1276','Durch eine Labyrinth von 2 hoch 30
Reformmoeglichkeiten','3T','8');

INSERT INTO courseunitdata VALUES ('34-INFO-29775','Internet-Systeme und Web-
Applikationen','1S','6');
INSERT INTO courseunitdata VALUES ('224-INFO-65775','Complierbau','2S','4');
INSERT INTO courseunitdata VALUES ('365-INFO-32475','Softwaretechnik','2S','2');
INSERT INTO courseunitdata VALUES ('87-INFO-13324','Verteilte Datenbanken','1S','6');
INSERT INTO courseunitdata VALUES ('378-INFO-25435','Hoehere Mathematik','1S','2');



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 ('1','45-SPO-1234','2.3','A');
INSERT INTO resultdata VALUES ('1','832-SPO-935','2.3','A');
INSERT INTO resultdata VALUES ('1','872-SPO-5335','2.3','A');

INSERT INTO resultdata VALUES ('2','362-SWI-685','1.7','C');
INSERT INTO resultdata VALUES ('2','893-SWI-543','1.7','C');
INSERT INTO resultdata VALUES ('2','27-SWI-1276','1.7','C');
INSERT INTO resultdata VALUES ('2','125-SWI-7247','3.7','C');
INSERT INTO resultdata VALUES ('2','854-SWI-2783','4.0','B');

INSERT INTO resultdata VALUES ('3','34-INFO-29775','3.0','E');
INSERT INTO resultdata VALUES ('3','224-INFO-65775','3.0','E');

INSERT INTO resultdata VALUES ('17','34-INFO-29775','1.3','A');
INSERT INTO resultdata VALUES ('17','224-INFO-65775','1.7','B');
INSERT INTO resultdata VALUES ('17','87-INFO-13324','2.0','C');
```

APPENDIX E XSLT code for Generating the HTML ToR

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- HTMLToRTemplate.xslt -->
<!-- @author Tom.Stiller@cm-tm.uka.de -->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<head>
<title>ECTS - EUROPEAN CREDIT TRANSFER SYSTEM</title>
</head>

<body>

<table border="0" width="800" cellspacing="10" cellpadding="2" id="table1">
   <tr>
      <td>
      <p align="center"><b><font face="Times New Roman" size="5">
      ECTS - EUROPEAN CREDIT TRANSFER SYSTEM</font></b></p>
      <p align="center"><b><font face="Times New Roman" size="4">
      TRANSCRIPT OF RECORDS</font></b></p><p></p></td>
   </tr>
   <tr>
      <td>
      <table border="2" width="100%" cellspacing="0" cellpadding="0"
bordercolor="#000000" id="table2">
         <tr>
            <td>
            <table border="0" width="100%" cellspacing="4" cellpadding="0" id="table3">
               <tr>
                  <td colspan="3"><b>NAME OF SENDING INSTITUTION: </b>
                  <xsl:value-of select="//Header/SendingInstitution/InstitutionName" />
                  </td>
               </tr>
               <tr>
                  <td colspan="3"><b>Faculty/Department of:  </b>
                  <xsl:value-of
select="//Header/SendingInstitution/FacultyDepartmentName" />
                  </td>
               </tr>
               <tr>
                  <td colspan="3"><b>ECTS departmental co-ordinator: </b>
                  <xsl:value-of
select="//Header/SendingInstitution/ECTSDepartmentalCoordinator/Surname" />,
                  <xsl:value-of
select="//Header/SendingInstitution/ECTSDepartmentalCoordinator/GivenName" /></td>
               </tr>
               <tr>
                  <td width="33%"><b>Tel.: </b>
                  <xsl:value-of
select="//Header/SendingInstitution/ECTSDepartmentalCoordinator/Telephone" />
                  </td>
                  <td width="33%"><b>Fax.: </b>
                  <xsl:value-of
select="//Header/SendingInstitution/ECTSDepartmentalCoordinator/Fax" />
                  </td>
                  <td width="33%"><b>e-mail box: </b>
                  <xsl:value-of
select="//Header/SendingInstitution/ECTSDepartmentalCoordinator/EMail" />
                  </td>
               </tr>
            </table>
            </td>
         </tr>
      </table>
      </td>
   </tr>
   <tr>
      <td>
      <table border="2" width="100%" cellspacing="0" cellpadding="0"
bordercolor="#000000" id="table6">
         <tr>
            <td>
            <table border="0" width="100%" cellspacing="4" cellpadding="0" id="table7">
               <tr>
                  <td><b>NAME OF STUDENT: </b>
                     <xsl:value-of select="//Header/Student/Surname" />
                  </td>
```

```
                  <td colspan="2" width="45%"><b>First name: </b>
                     <xsl:value-of select="//Header/Student/GivenName" />
                  </td>
               </tr>
               <tr>
                  <td><b>Date and place of birth: </b>
                     <xsl:value-of select="//Header/Student/DateOfBirth" />,
                     <xsl:value-of select="//Header/Student/PlaceOfBirth" />
                  </td>
                  <td colspan="2" width="45%"><b>Sex: </b>
                     <xsl:if test="//Header/Student/Sex='false'">female</xsl:if>
                     <xsl:if test="//Header/Student/Sex='true'">male</xsl:if>
                   </td>
               </tr>
               <tr>
                  <td colspan="2"><b>Matriculation date:</b>
                     <xsl:value-of select="//Header/Student/DateOfMatriculation" />
                  </td>
                  <td width="45%"><b>Matriculation number:</b>
                     <xsl:value-of select="//Header/Student/MatriculationNumber" />
                  </td>
               </tr>
            </table>
            </td>
         </tr>
      </table>
      </td>
   </tr>
   <tr>
      <td>
      <table border="2" width="100%" cellspacing="0" cellpadding="0"
bordercolor="#000000" id="table10">
         <tr>
            <td>
            <table border="0" width="100%" cellspacing="4" cellpadding="0" id="table11">
               <tr>
                  <td colspan="3"><b>NAME OF RECEIVING INSTITUTION: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/InstitutionName" />
                  </td>
               </tr>
               <tr>
                  <td colspan="3"><b>Faculty/Department of: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/FacultyDepartmentName" />
                  </td>
               </tr>
               <tr>
                  <td colspan="3"><b>ECTS departmental co-ordinator: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/ECTSDepartmentalCoordinator/Surname" />,
                     <xsl:value-of
select="//Header/ReceivingInstitution/ECTSDepartmentalCoordinator/GivenName" />
                  </td>
               </tr>
               <tr>
                  <td width="33%"><b>Tel.: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/ECTSDepartmentalCoordinator/Telephone" />
                  </td>
                  <td width="33%"><b>Fax.: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/ECTSDepartmentalCoordinator/Fax" />
                  </td>
                  <td width="33%"><b>e-mail box: </b>
                     <xsl:value-of
select="//Header/ReceivingInstitution/ECTSDepartmentalCoordinator/EMail" />
                  </td>
               </tr>
            </table>
            </td>
         </tr>
      </table>
      </td>
   </tr>
   <tr>
      <td>
      <table border="2" width="100%" cellspacing="0" cellpadding="0"
bordercolor="#000000" id="table14">
         <tr>
            <td>
            <table border="0" width="100%" cellspacing="5" id="table15">
               <tr>
                  <td><b>Course unit code</b></td>
                  <td width="33%"><b>Title of course unit</b></td>
```

```
                  <td><b>Duration of course unit</b></td>
                  <td width="10%"><b>Local grade</b></td>
                  <td width="10%"><b>ECTS grade</b></td>
                  <td width="10%"><b>ECTS credits</b></td>
               </tr>
               <tr>
                  <td colspan="6" width="100%" height="1" bordercolor="#000000">
                  <hr noshade="" color="#000000" size="1"></hr></td>
               </tr>
               <xsl:for-each select="//TableOfGrades/CourseUnit">
               <tr>
                  <td height="22"><xsl:value-of select="CourseUnitCode"/></td>
                  <td height="22"><xsl:value-of select="CourseTitle"/></td>
                  <td height="22"><xsl:value-of select="CourseDuration"/></td>
                  <td height="22"><xsl:value-of select="LocalGrade"/></td>
                  <td height="22"><xsl:value-of select="ECTSGrade"/></td>
                  <td height="22"><xsl:value-of select="ECTSCredits"/></td>
               </tr>
               </xsl:for-each>
               <tr>
                  <td colspan="6" width="100%" height="1" bordercolor="#000000">
                  <hr noshade="" color="#000000" size="1"></hr></td>
               </tr>

               <tr>
                  <td></td>
                  <td width="33%"></td>
                  <td></td>
                  <td width="10%"></td>
                  <td width="10%"><b>Total</b></td>
                  <td width="10%">
                     <xsl:value-of select="//TableOfGrades/TotalECTSCredits" /></td>
               </tr>

            </table>
            </td>
         </tr>
      </table>

      </td>
   </tr>
   <tr>
      <td><b>Diploma/degree awarded:</b> </td>
   </tr>
   <tr>
      <td></td>
   </tr>
</table>
</body>
</html>

</xsl:template>
</xsl:stylesheet>
```

TABLES

Abbreviations and Glossary

Abbreviation or Term	Full Name and/or Term Description
ABAP	Advanced Business Application Programming Programming language for the development of applications for SAP systems.
API	Application Programming Interface An Application Programming Interface is a set of definitions of the ways one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software.
BAPI	Business Application Programming Interface Interface on the business object layer, providing access for non-SAP components.
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 [Wikipedia, http://en.wikipedia.org/wiki/BPEL].
BPM	Business Process Management The term Business Process Management (or BPM) refers to a set of activities which organizations can perform to either optimize their business processes or adapt them to new organizational needs. As these activities are usually aided by software tools, the term BPM is synonymously used to refer to the software tools themselves. Although it can be said that organizations have been performing BPM for some time, a new impetus has been given to the theme with the advent of software tools (business process management systems or BPMS) which allow for the direct execution of the business processes without a costly and time intensive development of the required software. In addition, these tools can also monitor the execution of the business processes, providing the management of an organization the means to analyze their performance and make changes to the original processes with the aim of improving them. Using the BPMS the modified processes can then be quickly placed into operation [Wikipedia, http://de.wikipedia.org/wiki/Business_Process_Management].
BPMN	Business Process Modeling Notation Important specification in the context of Business Process Modeling (BPM)

developed by the Business Process Management Initiative (BPMI).

BOR — Business Object Repository
The Business Object Repository is the central access point to the SAP business object types and their BAPIs.

CGI — Common Gateway Interface
Common Gateway Interface is an important World Wide Web technology that enables a client web browser to request data from a program executed on the Web server. CGI specifies a standard for passing data between the client and the program.

COBOL — COBOL is a third-generation programming language. Its name is an acronym, for COmmon Business Oriented Language, defining its primary domain in business, finance, and administrative systems for companies and governments.
COBOL was initially created in 1959 by The Short Range Committee, one of three committees proposed at a meeting held at the Pentagon in May 1959, organized by Charles Phillips of the United States Department of Defense. The Short Range Committee was formed to recommend a short range approach to a common business language.
Nowadays many COBOL programs are still in use in major commercial enterprises, notably financial institutions. The expense of rewriting a very large code base that has already been debugged, in a new language has not been thought worth any benefits that might ensue. In the late 1990s, the Gartner Group, a data-processing industry research organization, estimated that of the 300 billion lines of computer code that existed, eighty percent - or 240 billion lines - were COBOL. They also reported that more than half of all new mission-critical applications were still being created using COBOL.
[http://en.wikipedia.org/wiki/COBOL].

ECTS — The European Credit Transfer and Accumulation System
A student-centered system based on the student workload required to achieve the objectives of a program, objectives preferably specified in terms of the learning outcomes and competences to be acquired [http://europa.eu.int/comm/education/programmes/socrates/ects_en.html].

EAI — Enterprise Application Integration
“Enterprise Application Integration (EAI) is the use of software and architectural principles to bring together (integrate) a set of enterprise computer applications. It is an area of computer systems architecture that gained wide recognition from about 2004 onwards. EAI is related to middleware technologies such as message-oriented middleware MOM, and data representation technologies such as XML. Newer EAI technologies involve using web services as part of Service-oriented Architecture as a means of integration.”
[Wikipedia, http://en.wikipedia.org/wiki/Enterprise_application_integration]

ERP — Enterprise Resource Planning
ERP systems are management information systems that integrate and automate many of the business practices associated with the operations or production aspects of a company.

HTML — HyperText Markup Language
In computing, HyperText Markup Language is a markup language designed

	for the creation of web pages and other information viewable in a browser.
HTTP	HyperText Transfer Protocol HTTP is the primary method used to convey information on the World Wide Web. The original purpose was to provide a way to publish and receive HTML pages.
J2SE	Java-2-Platform, Standard Edition Variant of the Java framework for the development of client applications.
JCo	SAP Java Connector A Java-based middleware, acting as a bridge between ABAP and Java.
JDK	Java Development Toolkit Set of design tools (interpreter, compiler) and class libraries
JNI	Java Native Interface The Java Native Interface (JNI) is a programming framework that allows Java code running in the Java virtual machine (VM) to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages, such as C, C++ and assembly. The JNI is used to write native methods to handle situations when an application cannot be written entirely in the Java programming language such as when the standard Java class library does not support the platform-dependent features or program library. It is also used to modify an existing application, written in another programming language, to be accessible to Java applications [Wikipedia, http://en.wikipedia.org/wiki/JNI].
JSP	Java Server Pages JSP or Java Server Pages, known to some as the Java Scripting Preprocessor, is a Java technology that allows developers to dynamically generate HTML, XML or some other type of web page. The technology allows Java code and certain pre-defined actions to be embedded into static content [Wikipedia, http://en.wikipedia.org/wiki/JavaServer_Pages].
Legacy System	A legacy system is an antiquated computer system or application program which continues to be used because the user (typically an organization) does not want to replace or redesign it [Wikipedia, http://en.wikipedia.org/wiki/Legacy_system].
RFC	Remote Function Call In the SAP context, RFC stands for Remote Function Call and is used to invoke function modules. Technically, it the same as a Remote Procedure Call (RPC).
RFM	RFC-enabled Function Module SAP program module, programmed in ABAP, which can be called by other systems.
RPC	Remote Procedure Call „A remote procedure call (RPC) is a protocol that allows a computer program running on one host to cause code to be executed on another host without the programmer needing to explicitly code for this. When the code in question is written using object-oriented principles, RPC is sometimes referred to as

	remote invocation or remote method invocation." [Wikipedia, http://en.wikipedia.org/wiki/RPC]
SAP	SAP (Systeme, Anwendungen, Produkte in der Datenverarbeitung) SAP is an acronym for "Systeme, Anwendungen, Produkte in der Datenverarbeitung", which means "Systems, Applications and Products in data processing". The company was founded in 1972 by five former IBM employees. As of 2005, SAP employs over 28,900 people in more than 50 countries.
SAP CM	SAP Campus Management Software module Campus Management of the SAP R/3 system especially for University Resource Planning (URP).
SAP GUI	Client program to access SAP R/3 Systems Graphical user interface (or GUI, pronounced "gooey") used to access SAP R/3 systems.
SAP R/3	SAP R/3 SAP R/3 is the name of the main ERP software developed by the company SAP. Its new version is named mySAP.
Servlet	The Java Servlet API allows a software developer to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to dynamic web content technologies such as CGI. It has the ability to maintain state after many server transactions. This is done using HTTP Cookies, session variables or URL rewriting.
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 hrough the interfaces. XML is the most obvious format that meets this constraint.”

[W3C Working Group Note 11.02.2004, http://www.w3.org/TR/ws-arch/]

SQL	Structured Query Language SQL is the most popular computer language used to create, modify and retrieve data from relational database management systems.
ToR	Transcript of Records At the end of a time of study at a university, the students receive a report (Transcript of Records) stating their completed courses and exams. In this report, the students’ achievements are stated in a plain and comprehensive way, such that a transfer to another higher education institution according to the Bologna requirements can be achieved smoothly.
UML	Unified Modeling Language Language to describe any objects (e.g., software systems or business units) in a semi-formal way using graphical elements.
URP	University Resource Planning University Resource Planning is a university-oriented ERP solution. It consists of supplementary services and integrated administration information services.
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.” [W3C Working Group Note 11.02.2004, http://www.w3.org/TR/ws-arch/]
WSDL	Web Service Description Language XML format used for describing Web service interfaces.
WSIF	Web Service Invocation Framework WSIF stands for the Web services Invocation Framework. It supports a simple Java API for invoking Web services, no matter how or where the services are provided. The framework allows maximum flexibility for the invocation of any WSDL-described service [Apache Web Services Project, http://ws.apache.org/wsif/overview.html].
WUSKAR	Werkstatt UnternehmensSoftware KARlsruhe WUSKAR has been initiated upon the recommendation of a task force of the

state government of Baden-Wuerttemberg.
Students of computer science in all higher education institutions in Karlsruhe have access to virtual servers in order to solve problems coming from practical cases in the industry.
Thus students gain knowledge about software used in industrial applications and additionally acquire skills regarding methods of problem solving techniques in the industry.

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.
XPath	XML Path Language XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer vocabulary [W3C consortium, http://www.w3.org].
XPointer	XML Pointer Language (XPointer) is used as the basis for a fragment identifier for any URI reference that locates a resource whose Internet media type is one of text/xml, application/xml, text/xml-external-parsed-entity, or application/xml-external-parsed-entity [W3C consortium, http://www.w3.org].
XSL	XSL is a family of recommendations for defining XML document transformation and presentation. It consists of three parts: XSL Transformations (XSLT), the XML Path Language (XPath) and XSL Formatting Objects (XSL-FO) an XML vocabulary for specifying formatting semantics vocabulary [W3C consortium, http://www.w3.org].
XSLT	XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary [W3C consortium, http://www.w3.org].

Index

Information and Exercise Slides

References

[AE+04] Sebastian Abeck, Christian Emig, Jochen 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

[APACHE-WSIF] Official Homepage of the Apache Web Services Project, Web Services Invocation Framework http://ws.apache.org/wsif/

[BE05] Ingo Beutler, Sabine Enge: Entwicklung von Core-Webservices zur Generierung eines Transcript of Records (WUSKAR-Projektteam WiSe 2004/05), Praktikumsarbeit, Universität Karlsruhe (TH), 2005.

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

[CB03] Communiqué of the Conference of Ministers responsible for Higher

Education
in Berlin on 19 September 2003, REALISING THE EUROPEAN HIGHER EDUCATION AREA, http://www.bologna-berlin2003.de/pdf/Communique1.pdf, 2003.

[C&M-ABAP] Cooperation & Management, Tomas Stiller: Einführung in ABAP, Dokument im Rahmen der C&M Technologien und Werkzeuge, Universität Karlsruhe (TH), C&M (Prof. Abeck), Mai 2005.

[C&M-JCo] Cooperation & Management, Heiko Schandua: Einführung zum SAP Java Connector, Dokument im Rahmen der C&M Technologien und Werkzeuge, , Universität Karlsruhe (TH), C&M (Prof. Abeck), Mai 2005.

[EA04] Christian Emig, Sebastian Abeck: Werkstatt Unternehmenssoftware Karlsruhe - Ein Beitrag zur praxisorientierten Informatik-Ausbildung an Hochschulen, erscheint in: UNIKATH, Karlsruhe 2004.

[EM+05] Christian Emig, Christof Momm, Jochen Weisser, Sebastian Abeck: Programming in the Large based on the Business Process Modelling Notation, Jahrestagung der Gesellschaft für Informatik (GI), Bonn, 2005.

[Er04] Thomas Erl: Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services, Prentice Hall, 2004.

[EU01] Communiqué of the meeting of European Ministers in charge of Higher Education in Prague on May 19th 2001, TOWARDS THE EUROPEAN HIGHER EDUCATION AREA, http://www.bologna-berlin2003.de/pdf/Prague_communiquTheta.pdf, 2001.

[EU99] Joint declaration of the European Ministers of Education, The Bologna Declaration of June 19th 1999, http://www.bologna-berlin2003.de/pdf/bologna_declaration.pdf, 1999.

[JAVA-SUN] Sun Developer Network
http://java.sun.com/

[Li03] David S. Linthicum: Next Generation Application Integration, From simple Information to Web Services, Addison Wesley, December 2003.

[RS+05] Elmar Reuther, Tomas Stiller, Sophie Tardif d'Hamonville, Jochen Weisser, Christian Emig, Sebastian Abeck: Geschäftsprozess- und Systemmodellierung von SAP Campus Management, Bericht zum Projekt „Werkstatt Unternehmenssoftware Karlsruhe" (WUSKAR), Karlsruhe 2005.

[RW05] Erik Rull, Jochen Weisser: Einführung in Java Webservices, Dokument im Rahmen der C&M Technologien und Werkzeuge, Universität Karlsruhe (TH), 2005.

[SAP-ABAP] SAP AG: ABAP-Einführung, Michael Höding, HCC-Kurs an der FH Brandenburg, März 2005.

[SAP-BasisTechnologien] SAP AG: SAP Basis Technologien, HCC-Kurs an der TU München, März 2005.

[SAP-CM] SAP AG: Campus Management (IS-HER-CM) Release 471, Online Help, 26.02.2003.

[SAP-Help] SAP Knowledge Warehouse, Communication between ABAP- and Non-ABAP Technologies, SAP Java Connector, http://help.sap.com/saphelp_erp2004/.

[SAP-JCo] Valentin Nicolescu, Yuriy Taranovych: Einführungsschulung: Entwicklung von SAP Web-Applikationen mit Java/JCo, (HCC-Kurs an der TU München), März 2005.

[Sl05] Christian Slamka: Portal Integration of SAP Campus Management, Practical Work, University of Karlsruhe (TH), C&M (Prof. Abeck), 2005.

[We05] Jochen Weisser: University SOA – Building a Transcript of Records Service, Studienarbeit, Universität Karlsruhe, 2005.