University of Warsaw
Faculty of Mathematics, Computer Science and Mechanics

Rafał Nagrodzki
Student no. 219548

The Mobility Project

Master's thesis
in COMPUTER SCIENCE

Supervisor:

dr Janina Mincer-Daszkiewicz
Institute of Informatics

November 2009

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Abstract

Student mobility becomes a phenomenon of growing scale and importance. It is common that higher education institutions keep IT infrastructure which enables them to reduce the amount of work needed to manage all the issues involved in running the studies. However, the process of exchanging students is carried in a traditional way which engages a lot of paperwork and communication by phone, fax, e-mail. Thus the process is quite ineffective and further development of mobility is endangered. It became apparent that tertiary education community all across Europe needs a platform which would provide effective electronic exchange of data in the mobility context.

This thesis presents an effort on creating an infrastructure to support electronic data exchange facilitating student mobility. The term infrastructure is understood as data format, architecture and a prototype generic software solution which could be integrated with IT solutions used at higher education institutions.

Keywords

student mobility, Erasmus programme, SOA, P2P, web services, WSDL, UDDI, data exchange, Rich Internet Applications

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatics

Subject classification

J. Computer Applications
J.1 Administrative Data Processing - *Education*
H. Information Systems
H.3 Information Storage and Retrieval
H.3.4 Distributed systems

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Student mobility becomes a phenomenon of a continually increasing extent and importance. From Polish and European perspective student exchange is mainly led by Socrates-Erasmus programme, organized by European Union. Table 1.1 shows the number of incoming mobility students to several European countries.

Host country	Year		
	2005	2006	2007
France	236 518	247 510	246 612
Italy	44 921	49 090	57 271
Poland	10 185	11 365	13 021
UK	318 399	330 078	351 470

Table 1.1: Total inbound mobility students traffic in tertiary education by host country [UIS]

In Europe as a whole, this undertaking involves now over 4000 higher education institutions while the number of participating students approaches two million [ECERA]. The European Commission introduced the new Lifelong Learning Programme which is aimed to further stimulate mobility and achieve a number of 3 million Erasmus students by 2012. Therefore it becomes an important part of HEIs' activity. Important also in a sense of the amount of administration work needed.

Currently, the vast majority of higher educational institutions is equipped with IT infrastructure which enables them to keep all the necessary data needed to run the studies, i.e. personal data, learning achievements data, accounting, etc. The process of exchanging the necessary data in order to actually perform the student exchange is carried in a manual way, i.e. employees from local International Relations Office contact with their counterparts at the partner's IRO through traditional mail, e-mail, fax. Details of mobility conditions are discussed, lists of students along with their study history get exchanged. That data needs to be entered to local IT system in order to render an id card to the guest student which is often the only way to let the guest student be recognized as a full member of local student society. Otherwise, their grades would not be recorded, access to library would not be granted, etc. After mobility period transcripts of records need to be issued in order to let the home institution recognize the progress of its students at the partner HEI.

All these cases of data exchange involve making a physical copy (usually a printout) of the data from the home institution and entering it to the partner system. This procedure

is carried by humans which tend to err, especially when handling texts in foreign languages. If the source of data is a form filled by hand, the copy process is even less reliable and error-prone. The result is that the quality of data kept in the local database suffers. The most common issue is for instance that the same person is visible to the system as two different individuals.

This manual procedure has another side effect which one cannot disregard. Namely, mobility initiatives may look to the potential participants less attractive because of the amount of bureaucracy involved. It may become a significant obstacle in further and larger cooperation in the mobility area.

A pretty obvious idea that instantly emerges when one considers how to improve this situation is to make those IT study-oriented systems talk to each other and perform necessary data exchange. That certainly would greatly simplify the whole process and make it much more reliable and less error-prone.

In general, every single higher educational institution has its own specific system and stores custom data in a custom format. The main problem to solve is to identify the most common subset of information needed to successfully conduct the student exchange and design the data formats used in transmission. There are on-going initiatives aiming to develop standards covering various aspects of student mobility, e.g. CEN Workshop on Learning Technologies [CENWSLT] and RS3G [RS3G] joint work. Yet there is no official or unofficial standard regarding electronic data exchange in the context of mobility scenarios.

Carried by a will to fill this hole, two European Higher Education consortia – Polish MUCI [MUCI] and Italian CINECA [CINECA] – initiated the Mobility Project: an agreement upon cooperation in developing a prototype which could eventually evolve into a mature standard equipped with a reference implementation [AMDR09]. The described package would then be easily adopted by other HEIs in Europe. There is, however, no intention to limit the scope to the Erasmus programme only; on the contrary, the idea is to try to come up with a solution generic enough to be suitable for the entire academic world. The work presented in this document is just a one step on a way to achieve this goal.

1.1. The Mobility Project

Although the Mobility Project has initially been started by MUCI and CINECA, the idea gathered substantial number of other organizations interested in it. As of November 2009, the Mobility Project gathered following participants:

- University consortia (Student information management system providers):
  - OODI, Finland,
  - SIGMA, Spain,
  - CINECA, Italy,
  - Ladok, Sweden,
  - FS, Norway,
  - MUCI, Poland,
  - HIS, Germany,
  - SURF, Netherlands,
  - Almalaurea, Italy,
  - VHS, Sweden.

- Individual universities:
  - University of Stuttgart, Germany,
  - University of Malaga, Spain,
  - University of Thessaloniki, Greece,
  - University of Porto, Portugal,
  - University Fernando Pessoa, Portugal.
- Companies:
  - Digitary, Ireland,
  - AcademyOne, USA,
  - KION, Italy,
  - QS unisolution, Germany.

The first official meeting of the participants took place during workshop of RS3G which was held in Uppsala, 16-17 November 2009 [RS3GWS].

1.2. Basic terms

Here come the explanations of some basic terms related to the problem domain.

ECTS (European Credit Transfer and Accumulation System) – a standard for comparing the study attainment and performance of students of higher education across the European Union and other collaborating European countries.

ECTS Information Package/Course Catalogue – the primary guide for all students attending the institution, contains information about the qualifications offered, the teaching, learning and assessment procedures, the level of programmes, the single educational components and the learning resources available to students, institutional or departamental/subject level tutor names with contact information – a detailed checklist of the recommended contents of this document is provided in the ECTS Users' Guide [ECTSUG]; the document should be easily accessible and the recommended way of fulfilling this requirement is to publish it on the institution's website in English.

ECTS credits – a measure of the student effort required for completing a course; one credit corresponds to 25 to 30 hours of lectures, classes, etc.

ECTS Grading Scheme – a common grading scheme imposed by ECTS, used for the purpose of credit transfer occurring when the student completes the study programme and returns from a partner HEI to the home HEI; particularly used in ToR documents; the scheme is shown in Table 1.2.

National Agency – an agency which coordinates mobility programmes organized by the European Union (grant distribution, statistics collection).

International Relations Office – an organizational unit of a HEI which is responsible for cooperation with foreign partner organizations on student and staff mobility, internships and staff trips.

ECTS grade	Definition
A	Excellent: outstanding performance with only minor errors
B	Very good: above the average standard but with some errors
C	Good: generally sound work with a number of notable errors
D	Satisfactory: fair but with significant shortcomings
E	Sufficient: performance meets the minimum criteria
FX	Fail: considerable further work required before credit can be awarded
F	Fail: considerable further work is required

Table 1.2: ECTS Grading Scheme

Learning Agreement – a document which defines courses chosen by a student for their mobility period.

Student information management system – IT system which is used at higher educational institution (or wider – educational institution) with purpose of facilitating its usual tasks including organization of classes for students, storage of students achievements, financial matters (stipends, payments), etc.

Transcript of Records – a document issued by a HEI containing study achievements of a student.

USOS (Uniwersytecki System Obsługi Studiów) – SIMS used at the University of Warsaw.

USOSweb – a publicly accessible website enabling students and lecturers to access information gathered in USOS.

Organizations which are referred to in this paper:

CEN (the European Committee for Standardization) – a non-governmental organization dedicated to development of European Standards (ENs) and other technical specifications; it is officially recognized as a European standards body by the European Union – its standards are also national standards in each of its 30 members [CEN].

RS3G (Rome Student Systems and Standards Group) – a self-established group of software implementers and stakeholders in the European Higher Education domain which is focused on contributing to the definition and adoption of electronic standards for the exchange of student data between HEIs [RS3G].

Cedefop (The European Centre for the Development of Vocational Training) – the European Agency founded to promote the development of vocational education and training (VET) in the European Union by providing information on and analyses of vocational education and training systems, policies, research and practice.

Cedefop's tasks are to:

- compile selected documentation and analyses of data,
- contribute to developing and coordinating research,
- exploit and disseminate information,

- encourage joint approaches to vocational education and training problems,
- provide a forum for debate and exchanges of ideas [CEDEFOP].

Other terms:

Dublin Core – a general-purpose standard, developed by DCMI [DCMI], for cross-domain information resource description, e.g. video, sound, image, text but also composite objects, e.g. web pages; although implementations usually utilize XML and are Resource Description Format based, Dublin Core is syntax independent; its element set is standardized by ISO (ISO 15836) [WIKIPEDIA].

Application profile – "a set of metadata elements, policies, and guidelines defined for a particular application. The elements may be from one or more element sets, thus allowing a given application to meet its functional requirements by using metadata from several element sets including locally defined sets. For example, a given application might choose a subset of the Dublin Core that meets its needs, or may include elements from the Dublin Core, another element set, and several locally defined elements, all combined in a single schema. An Application profile is not complete without documentation that defines the policies and best practices appropriate to the application" [DCMIG].

Moreover, I use adjectives "home" and "partner" with reference to institutions participating in mobility experience in the following meaning:

home – denotes institution which sends its students; a sending institution,

partner – denotes institution at which mobility students arrive and stay for a study during mobility; a host institution.

1.3. Abbreviations

Abbreviations used throughout this thesis are listed below.

HEI – higher educational institution.

LA – Learning Agreement.

ToR – Transcript of Records.

SIMS – student information management system.

IRO – International Relations Office.

ECTS IP/CC – ECTS Information Package/Course Catalogue.

1.4. Overview

The remainder of this thesis is organized as follows. In the next chapter, several relevant existing or under-development standards are enumerated and thoroughly described from the perspective of their applicability and reuse in data exchange format definition. Chapter 3 contains detailed requirements description and analysis. Projects which solve a similarly scoped problem to the one stated in the thesis are presented in Chapter 4. Chapter 5 is devoted to the architecture and design of the developed software system, while Chapter 6 concentrates on

the XML Schema data format definition for the web service used by the system. In Chapter 7 various aspects of the implementation of the system are presented. The last chapter contains the summary and a deeper insight into possible directions which the Mobility Project may take in the near future.

The thesis is supplied with two appendices: Appendix A provides a comprehensive documentation of the software, whereas Appendix B lists the contents of a DVD media attached to the thesis.

Chapter 2

Related standards

When considering a data format for a specific application it is always vital to review existing and currently emerging standardization efforts in the problem area and consider their actual applicability and whether it is possible to reuse the ideas and approach. In this chapter several related standards are presented.

2.1. Europass

Cedefop came up with an initiative of Europass. Europass is a set of five documents (CV, Language Passport, Certificate Supplement, Diploma Supplement, Mobility) which are intended to ease the communication of personal skills and competences in the European job market.

From the point of view of the problem stated, the most interesting documents in the catalogue are CV and Mobility, as these two attempt to cover areas of personal data and mobility records. Although these standards are pretty mature and are endorsed by EU, they do not match the requirements. That's because these documents serve as a record of past events and the range of detail is not appropriate since the main appliance of Europass is mostly connected with an activity of applying for a study or a job and prepared documents are intended to serve as a generic basis for generation of office documents (such as Microsoft Office doc, Open Document or PDF file) which would look attractive when printed. There are online editors and generators of Europass CVs like for instance SmartCV (`http://www.smartcv.org/`).

At the time of writing, Cedefop has invented XML vocabulary for CV and Language Passport [EPXSD] only; the Mobility document XML Schema is currently just a draft [EPMOBXSD]. Although the Mobility document schema contains some useful definitions regarding grades, subject area classification, academic periods, etc., these constructs are actually tied to the ECTS environment and could not be directly used. Nevertheless, this kind of vocabulary is generally necessary in the Mobility Project.

Cedefop has delegated implementation of a distributed system facilitating exchange of Europass documents – refer to Section 4.2 for more information.

2.2. Metadata for Learning Opportunities and its derivatives

Metadata for Learning Opportunities Metadata for Learning Opportunities (MLO) [MLO], developed by CEN Workshop on Learning Technologies (CEN WS/LT), are an attempt to provide a standard for representation of course and learning opportunity information. It originated on a basis of the Bologna process, which main intention is making higher

education more comparable and compatible, therefore creating the European Higher Education Area. MLO focuses on needs of HEIs (study offerings, mobility), learners and employers but it also takes other potential stakeholders into account, for example:

- aggregators of learning opportunities such as:
  - PLOTEUS (Portal on Learning Opportunities throughout the European Space) – `http://ec.europa.eu/ploteus/home.jsp`,
  - HotCourses – `www.hotcourses.com`,
  - FastTomato – `www.fasttomato.com`,
- providers of value-added services (guidance, advice) like:
  - UCAS – `http://www.ucas.ac.uk/`,
  - Graduate Prospects – `http://www.prospects.ac.uk/`.

All above organizations would benefit from better consistency and availability of course information. The same also applies to other governmental and non-governmental agencies performing benchmarks and quality assurance of learning offerings.

Metadata for Learning Opportunities intends to contribute to the consolidation of European transparency documents, harmonize the application process, support analysis, quality evaluation, benchmarking and monitoring of learning opportunities and by making the information more consistent and available – enable new entrants to the market, e.g. Web 2.0 services.

ECTS IP/CC MLO Application Profile A MLO's derivative – ECTS IP/CC MLO Application Profile [MLO-AP] – facilitates representation of the ECTS Information Package/Course Catalogue (ECTS IP/CC), a core Bologna process document. The standard specifies refinements to the Metadata for Learning Opportunities (MLO) Information Model and technically it is defined as a set of properties and resources drawn from vocabularies of the Dublin Core element set [DCES].

ECTS IP/CC document is intended to serve as a primary source of information for mobile students and staff. The document lists the study programmes, course units and modules available at a certain HEI as well as other practical information regarding studying, procedures, living conditions, etc. The document itself does not have a defined format, it merely should cover all the items mentioned in the checklist provided in the ECTS Users' Guide [ECTSUG]. The Guide strongly encourages to make the document easily available on a HEI's website.

Although MLO does not define an XML representation, it is possible to translate Dublin Core conceptual world to XML world. There are actually two ways of doing that. The first, being an official recommendation of Dublin Core Metadata Initiative [DCMI] – the creator of Dublin Core metadata standard, is DC-XML-2003 [DCXML2003]. It has some limitations as its model is simpler than the Dublin Core Abstract Model. The other – DC-DS-XML [DCDSXML] – supports all the features of the description set described by the DCMI Abstract Model but its current status is no more than "proposed recommendation".

European Learner Mobility The main scope of this forthcoming standard, developed by CEN WS/LT, is to be "a data model for the expression and exchange of European Learner Mobility information, as defined by the European transparency instruments" [ELM].

In other words, the effort is put towards enabling IT systems in Europe to exchange Europass related information. At current, initial phase, the main attention is drawn to describe the Diploma Supplement document. The ELM project also aims to provide the vision and prepare the grounds "for the further development, augmentation and exploitation of transparency information that will lead to the implementation of valuable services to the community (e-portfolio, learning and employment opportunity exploration, etc.)" [SGWV].

The overall goals of the project include:

- improve European-wide adoption of electronic transparency documents and therefore improve data consistency,
- support the development of IT systems facilitating the Bologna process,
- support quality assurance and data quality management in the European higher education area,
- support wider availability of brokerage services.

European Learner Mobility is intended to build on MLO specification through an application profile approach. An application profile is understood here according to the definition found in [CWA15555] which states:

"An application profile is an assemblage of metadata elements selected from one or more metadata schemas and combined in a compound schema. The purpose of an application profile is to adapt or combine existing schemas into a package that is tailored to the functional requirements of a particular application, while retaining interoperability with the original base schemas".

It is quite similar to the definition used by DCMI which is cited in Section 1.2.

Although the project is still in its first steps, the current outcome has been submitted as an Enquiry Draft for a European Norm (EN) on an Europass DS interoperability. The specification recommendation is expected before the end of 2009 [SGWV] but at the time of writing this did not happen.

Summary Considering MLO and its Application Profile in the Mobility Project, it does not appear to be directly applicable. The most important argument is that MLO addresses quite divergent purposes – it aims at exchanging learning opportunities data only. Although the model serves the purpose of describing courses and credits awarded for them, the model is not sufficient because agreements, LAs and ToRs need to be described but these are not in the scope of the standard. Furthermore, in case of the Mobility Project some of the constructs are actually obsolete. Besides that, it is not completely clear how to use these specifications with some standard language (e.g. XML) due to the lack of a suitable specification.

It is, however, apparent that these standards have their place in the image of higher education and a high level of interoperability between the models behind the transparency documents and formats intended to convey the mobility data would be greatly desired and appreciated.

2.3. SCHAC

SCHAC (SCHema for ACademia) [SCHAC] is the result of work carried by TERENA (Trans-European Research and Education Networking Association), an organization associating

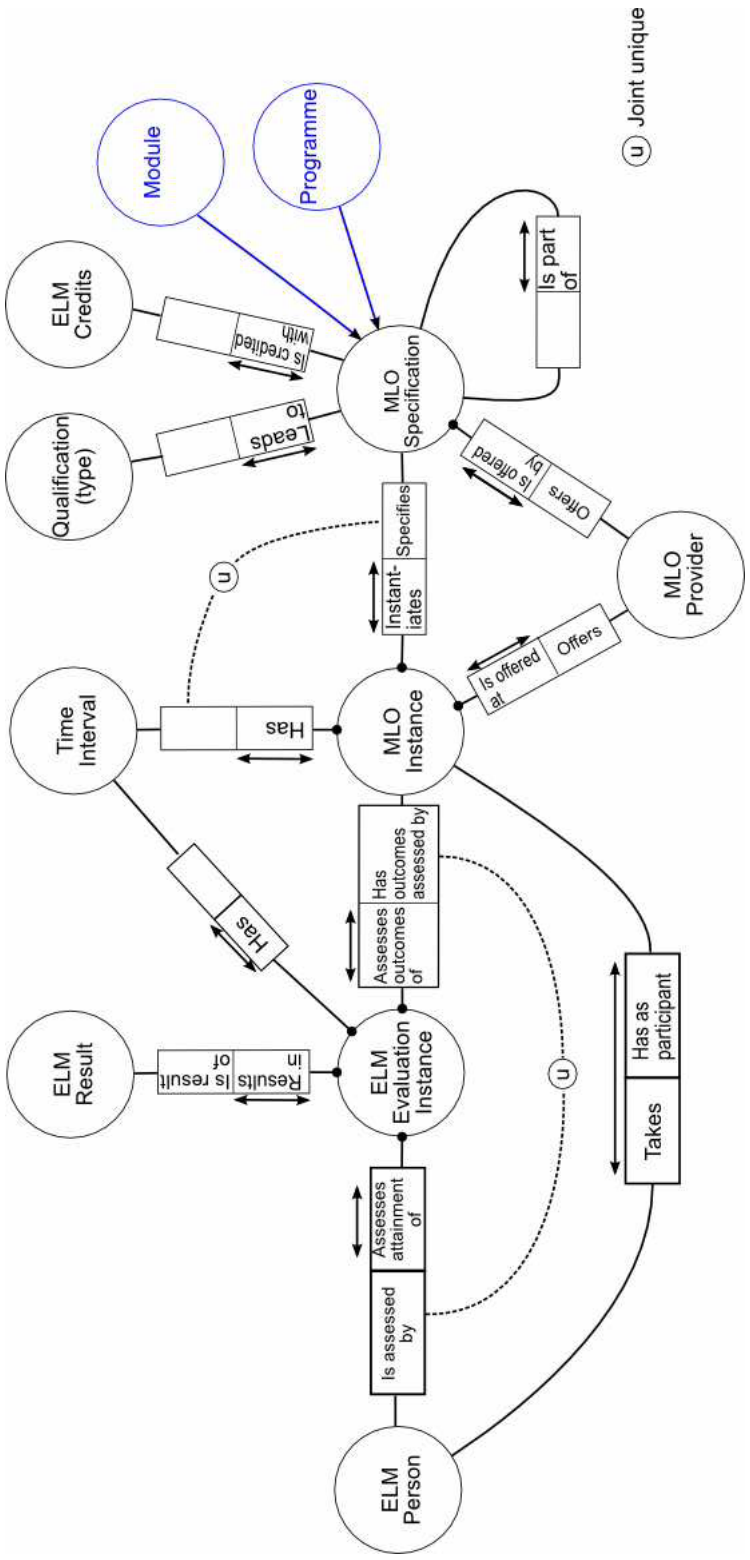

Figure 2.1: ELM Diploma Supplement Conceptual Model, taken from [SGWV]

members of research and education community willing to collaborate on development of Internet technology, infrastructure and services. The standard is intended to "promote common schemas in the field of higher education to facilitate interinstitutional data exchange" [SCHAC]. SCHAC is not tied to any particular technology, although at its current level it provides an appropriate LDAP profile. There is also an XML profile expected to be released in the near future.

SCHAC is a work-in-progress standard which currently supports description of persons and organizations, so it is currently unsuitable to describe student exchange, their achievements, etc. However, it appears as a good starting point, because it conveys some conventions and ideas which seem especially attractive. First of all, SCHAC consequently implements a concept of a unique identifier. For example `schacPersonalUniqueID` attribute has a form of `urn:mace:terena.org:schac:personalUniqueID:<country-code>:<idType>:<idValue>`. It is quite clear that this form easily enables users of the standard to uniquely identify a person with their national identification number which can possibly equal for different persons in two countries but the national prefix comes to help.

Another important feature of SCHAC is that it utilizes ISO standards whenever possible. The `<country-code>` part from the last example is coded as an ISO 3166-1-alpha-2 [ISO3166] country code identifier or the string *int*.

One disadvantage of SCHAC's definitions is that they lack a hierarchy and therefore do not promote reuse of already defined constructs. Although the intention of SCHAC creators was not to tie it to any specific technology, it did not actually abandon the heritage of some LDAP-oriented standards which served as a point of reference for SCHAC on its way to satisfy the needs of European Higher Education environment. These standards are eduPerson/eduOrg [EDUPER] and inetOrgPerson [RFC2798]. EduPerson/eduOrg specifications were designed to provide a pattern for building general-purpose institutional directories at HEIs, whereas inetOrgPerson targeted a more general problem of describing a person in its organizational context.

It is worth to note that the format of attribute values is in several cases a bit bloated, for instance the already mentioned attribute `schacPersonalUniqueID` contains a considerable constant part (prefix). Long prefixes are also present in `schacHomeOrganizationType`, `schacPersonalPosition`, `schacPersonalUniqueCode` and `schacPersonalUniqueID` attributes.

SCHAC's concept of handling surnames does not seem as a particularly fortunate one. The standard distinguishes two-part surnames by two attributes `schacSn1` and `schacSn2`. An exemplary Polish double surname Górecka-Wolniewicz is coded as [`schacSn1`: Wolniewicz], [`schacSn2`: Górecka] but a Spanish example of Lopez de la Moraleda y de Las Altas Alcurnias is handled differently, i.e. parts are mapped in a different order: [`schacSn1`: Lopez de la Moraleda], [`schacSn2`: de Las Altas Alcurnias]. Apart from that the information about a hyphen in Polish and "y" in Spanish is lost.

SCHAC has been reused by some local standards, for example Australian auEduPerson [AUPWG] or Spanish irisEduPerson [IEP].

2.4. Finnish Virtual University specifications

Finnish Virtual University (FVU) is a partnership of all 21 Finnish universities. In order to strengthen national collaboration and improve on the student mobility, FVU defined a set of 4 specifications [FVUSPEC] approved by its members:

- M0: Study Rights Data (currently not available in English),

- M1: Degree, Study Module and Course Unit Data,
- M2: Course Unit Realisation Data,
- M3: Credits Data.

The project is pretty mature – the documents reached version at least 1.0 and they have not been modified since September 2006.

The specifications provide vocabulary to describe degrees, course units along with their dependencies, rules describing evaluation of student achievements, credits awarded for completing a course or study unit. The structures mirror the common model (degree programmes, specializations, study modules, free study modules) used by all Finnish HEIs which support a national programme called JOO studies. JOO is an acronym for Finnish phrase meaning "flexible study right" and this programme permits both undergraduate and postgraduate students to apply to take courses at other Finnish universities, provided that those courses are approved options within the student's degree program.

The Finnish standard makes some use of external norms (ISO 639, ISO 8601 [ISO8601]) but its particular weakness is that in numerous cases it allows values in free text which radically hardens automated processing of such data.

The format itself is actually abstract – the documents containing the description of the standard contain a notion of an "XML binding" but the actual XML representation is not supplied (at least in the documentation available in English).

Due to the above characteristics of FVU format and the fact that the model of a study introduced is tied to Finnish conditions, they are not of much straightforward use in the Mobility Project.

Chapter 3

Requirements analysis

3.1. Definitions

Here come the basic terms used to describe entities existing in mobility area. Some of them were identified by M. Krawczyński in his master's thesis [Kra06] documenting his effort on a USOS module which was designed specifically for local IRO at the University of Warsaw for mobility purposes. The list contains:

- agreement – an arrangement between two organizations which defines rights and obligations of the parties; in the context of mobility it describes:
  - numbers of students accepted by the parties,
  - requirements on these students such as study subject area and study level,
  - duration of mobility period,
- cooperation conditions – a part of an agreement regarding one specific field of study, study level and mobility duration,
- administrative coordinator – a person who is responsible for proper execution of decisions contained in an agreement,
- institutional coordinator – a person who is in charge of signing an agreement between two HEIs – usually a chancellor of university.

3.2. Business processes

This section contains business processes occurring at a HEI. The scenarios were discovered in a process of interviewing Ms. Klementyna Kielak, a representative of IRO held at the University of Warsaw. The information gained this way was supplemented with the insight of developers of USOS modules dedicated to handling mobility data [Kra06], [Lom08].

The core description is modeled in UML diagrams supplied. However, they would not be clear enough without a few words of additional commentary. Some details were omitted in the diagrams in order not to obscure the main outline.

3.2.1. Making an agreement

An initiative for making an agreement between two HEIs usually comes from interinstitutional research contacts and cooperation of research employees. The task of conducting the negotiations belongs to the duties of International Relations Office, so research employees contact the IRO which takes control of the issue. The situation described constitutes an entry point to the further procedure of making an agreement shown in Figure 3.1.

IRO representatives from both institutions discover other such intentions of cooperation on both sides and negotiate the agreement. The range of an agreement comprises (among other) subject area settings, levels of study with numbers of exchanged students, mobility duration. Finally, when the agreement is ready, a representative of one of the HEIs involved in the process prepares an official agreement document signed by the person entitled and sends it to the other HEI. The document gets signed and returns to its creator.

3.2.2. Making nominations

The process of nominating students (illustrated in Figure 3.2) starts with a recruitment phase driven by the faculties which begins right after the deadline for agreements in the current cycle. This deadline is scheduled by the National Agency. The recruitment phase is driven by the faculty recruitment commission which collects students' applications for mobility offers presented by the administrative coordinator.

When the commission gathers the applicants, the home coordinator sends the list of nominations to the partner IRO, partner coordinator and reports some aggregate data on the numbers of students nominated with mobility periods expected.

The partner coordinator informs the students nominated about the admission procedure. If some of the students fail to qualify, the vacancies may be taken by the applicants from the reserve list, unless it does not exist (it depends on the procedure at home HEI).

If there are still some available places for students, the home IRO announces an additional recruitment phase (unless it does not get permission from the authorities of the respective faculty). This phase significantly differs from the previous one because the students applying during it will not be granted with a mobility stipend.

The procedure has to finish by the turn of February and March – a deadline imposed by the National Agency. However, even after the deadline the list of nominated students may still slightly change.

Civil contract

The students qualified for a mobility programme need to sign a civil contract with their home HEI. The contract obliges a student to arrive at the partner HEI, study and pass the exams while the home HEI is obliged to recognize the achievements acquired at the partner HEI and pay the stipend on time.

3.2.3. Creating initial LA

A student who eventually succeeded to qualify for a mobility is required to choose courses from the partner catalogue (presumably using the ECTS IP/CC document issued by the partner HEI) which they decide to take, i.e. they define their LA.

At the University of Warsaw, students are asked to enter the courses of their choice into a form available through USOSweb. Next, the coordinator reviews the input from students

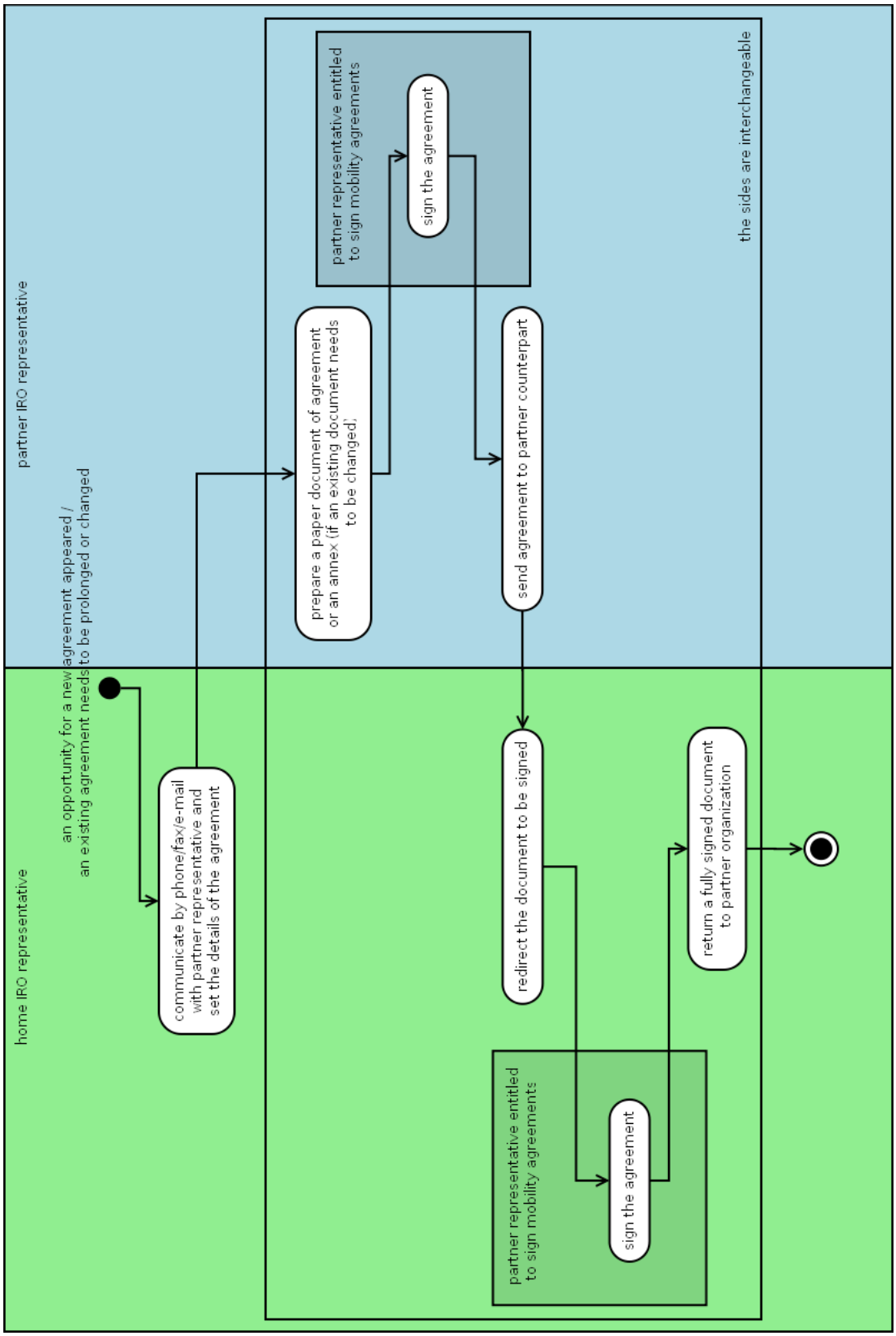

Figure 3.1: Making an agreement

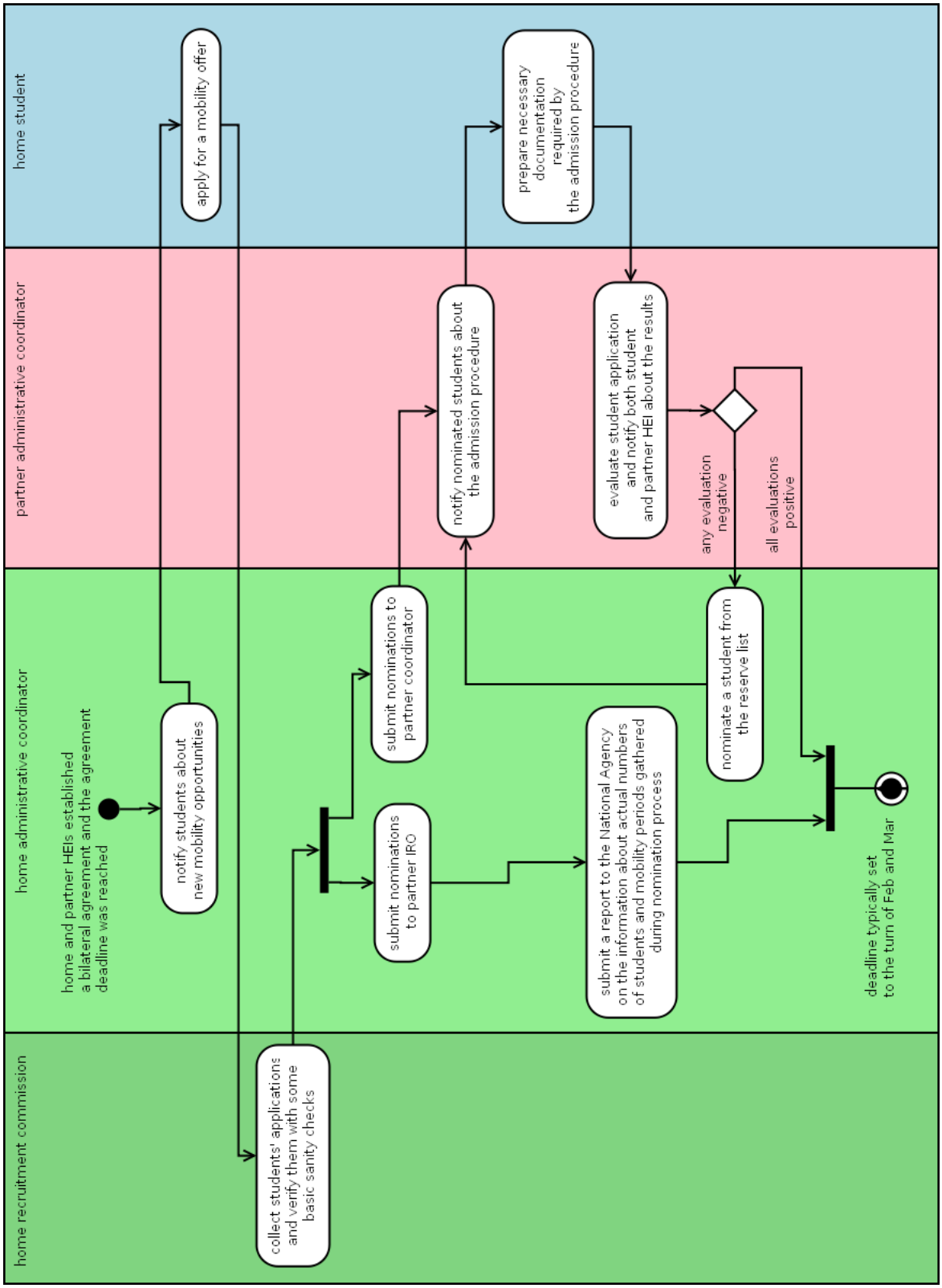

Figure 3.2: Making nominations

and either accepts it or asks them to correct their LAs. Accepted LAs get locked by the coordinator, students print their LAs and have them signed by the coordinator (unless the signature is not required by the partner HEI). Finally, students submit their paper LAs to the partner student's office.

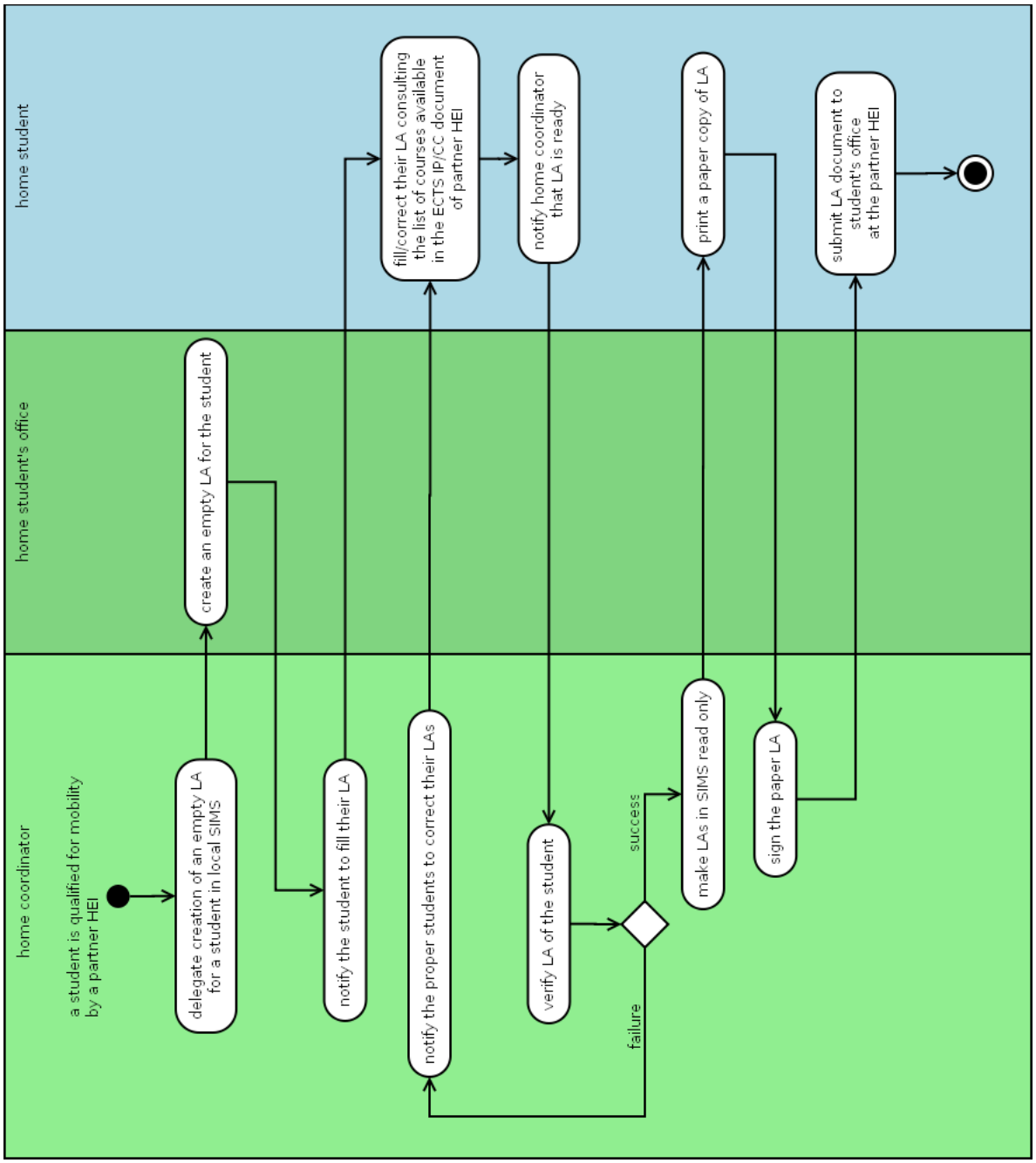

Figure 3.3: Creating initial LA

3.2.4. Updating LA

Even despite the greatest care from students and coordinators at both HEIs, there are often cases when students need to change their LA. It may happen when the course is removed or the actual language of the course is different than expected.

A student contacts with their home coordinator stating the issue; the coordinator unlocks the LA so the student can make the corrections and locks it again afterwards. The student prints the new LA and submits it to the partner HEI. Figure 3.4 presents this procedure.

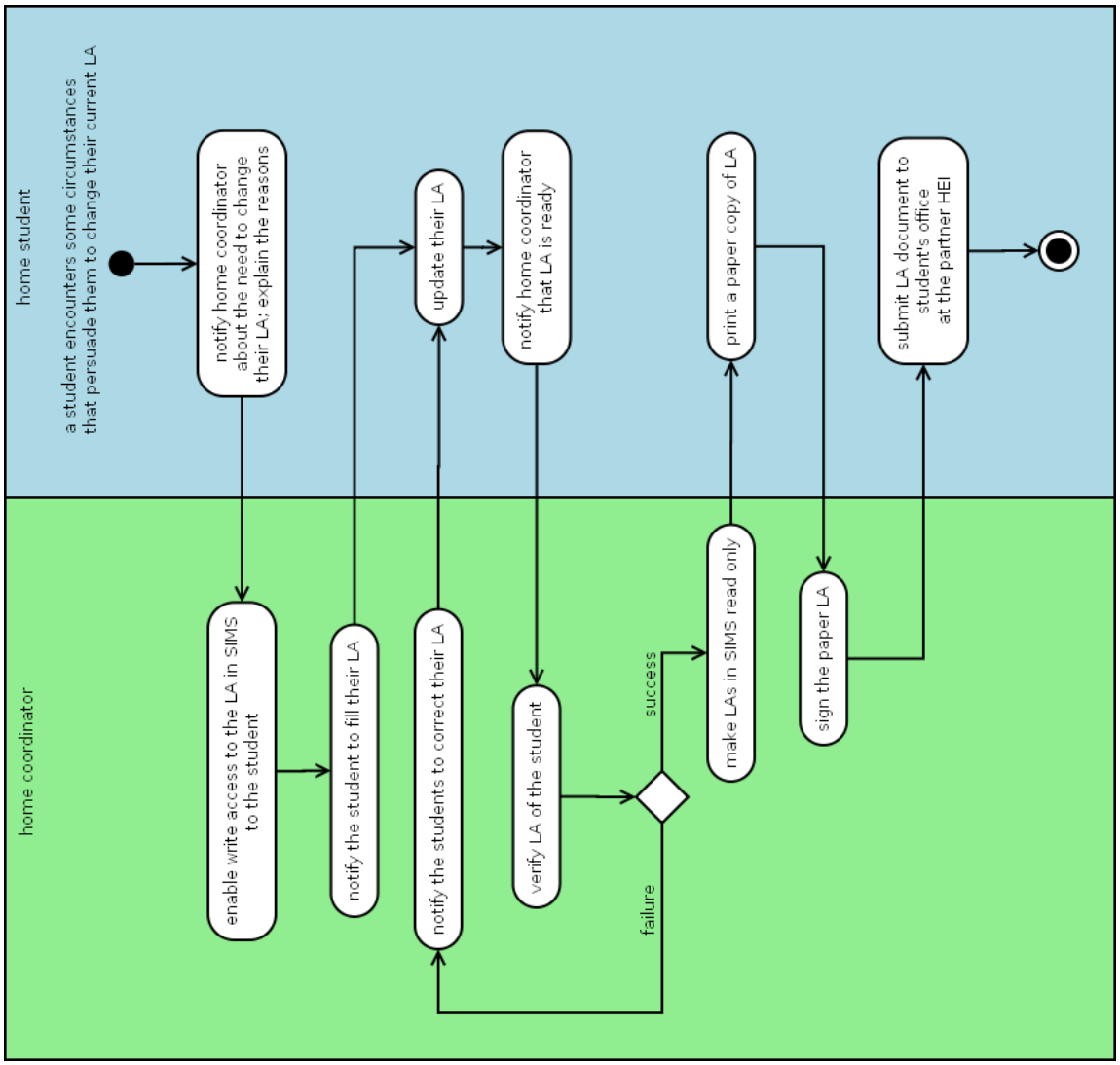

Figure 3.4: Updating LA

3.2.5. Creating ToR

A partner HEI issues the ToR document right after the mobility period of a student ends. Actually, two copies are prepared: one for the student themselves (precisely, many HEIs offer to send that copy to the address specified by the student and therefore they do not force the student to withdraw the document at the student's office) and the second for the IRO

of student's home HEI.

The copy for the student is intended to serve as a source of grades which are to be transferred to the student's records according to the "ECTS to local grade" mapping established by the student's home HEI. Any information about local grades is completely ignored.

The ToR instance sent to the IRO of student's home HEI is used to provide the statistical data demanded by the National Agency about achievements of students expressed in the ECTS Grading Scheme. The document is also stored in the student's files as a proof of participation in the mobility programme.

A diagram of the whole process is presented in Figure 3.5.

It may happen that the ToR needs to be reissued. Such situation occurs for example when a student improves their grade during the second examination which usually occurs after the mobility period or simply due to a mistake. The procedure of issuing the updated ToR is analogous.

3.2.6. Finalizing mobility

The stipends for mobility students are intended to be granted in an amount which exactly reflects their stay period. As the first approximation the dates defined in the civil contract (mentioned in the Subsection 3.2.2) between a student and their home HEI are taken. Students, however, do not arrive at and leave the partner HEI exactly on that dates. At the end of the mobility period both actual dates are known and it is possible to do the final accounting.

These dates are included in the document called "Letter of Confirmation" or "Confirmation of Stay" which is issued to a student by the partner HEI when the student is about to leave. The document needs to be presented at the home IRO and faculty. The home IRO resolves the financial issues. The final accounting has to be done ultimately usually by the end of August.

3.3. Requirements specification

3.3.1. Functional requirements

1. Provide necessary infrastructure for electronic data exchange over network between HEIs participating in student mobility programmes:
   (a) according to the use cases identified in Section 3.2,
   (b) the solution can be easily integrated with SIMS; it interoperates with SIMS in such a way that it does not require manual data entry more than once (the data is entered into SIMS only),
   (c) the data may be modified and exchanged multiple times.
2. Users must be authenticated.
3. The system should be testable with exemplary data; it should be possible to perform experimental transmissions without full integration with local SIMS.

3.3.2. Non-functional requirements

1. Data exchange format should be generic, i.e. facilitate the data exchange not only in terms of Erasmus programme but mobility programmes in general.

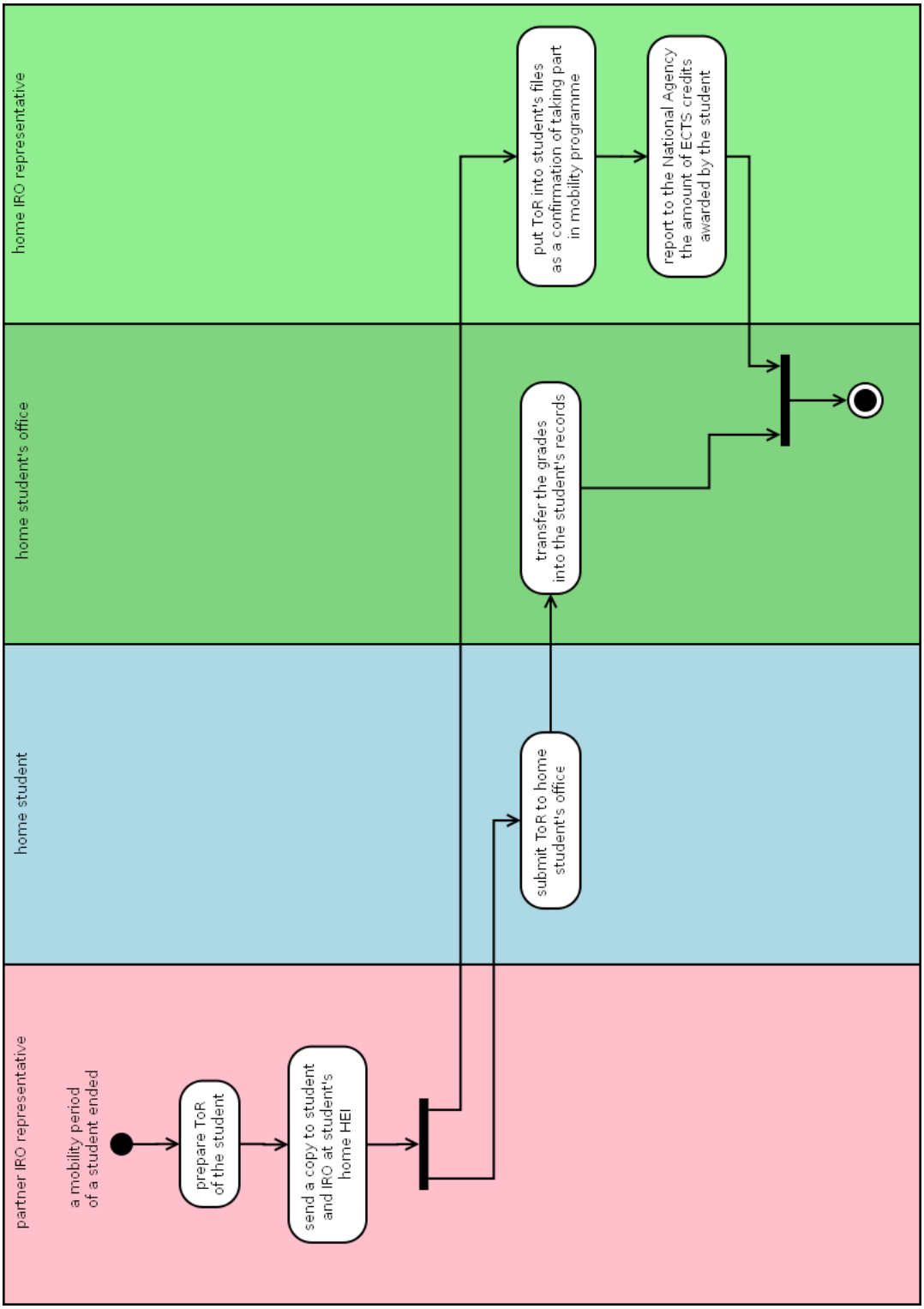

Figure 3.5: Creating ToR

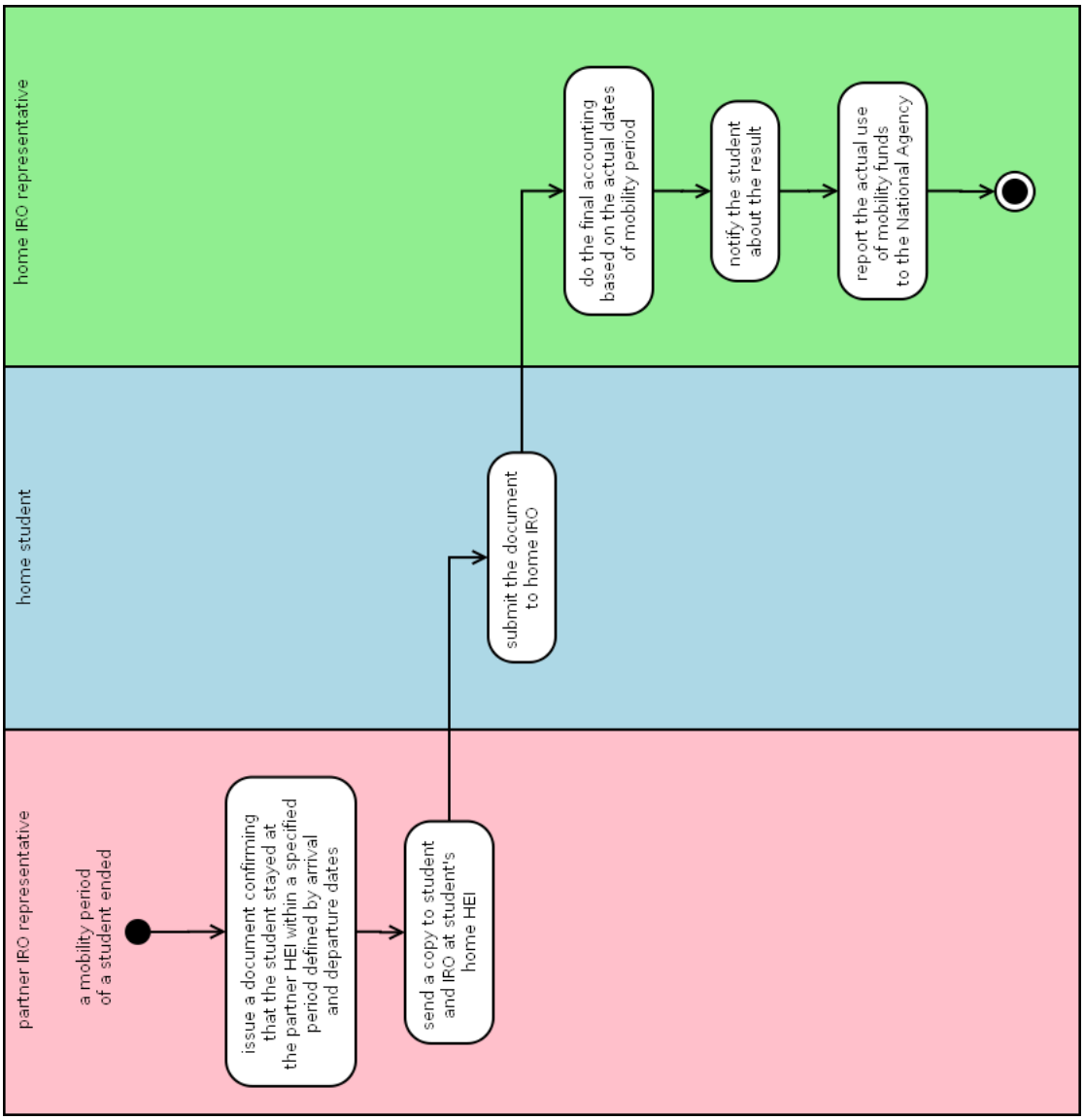

Figure 3.6: Finalizing mobility

2. Software should act as a complete, generic node being able to carry out transmission on its own.
3. Solution should be scalable.
4. Transport middleware should be highly vulnerable to transport data format changes, i.e. modification of the data format should have possibly minimal impact on the source code, ideally such operation would require no changes at all.
5. Data transmission must be secure, i.e. it cannot be intercepted and/or modified, data integrity must be preserved.
6. Consider personal data exchange legal concerns.
7. Software should be relatively easy to install and maintain.

8. Tools and technologies used to implement the software should be:
   (a) freely available; preferably licensed in a way which allows redistribution – ideally the licenses should be open source compatible,
   (b) maintained and widely used; with strong, active community,
   (c) independent of any specific OS or hardware architecture.

Non-functional requirements specific to the University of Warsaw

1. Software should be capable of communicating with Oracle Database $11g$ by supporting PL/SQL procedure calls and Oracle Advanced Queuing (Oracle AQ).

3.3.3. Assumptions

1. IT systems of HEIs have access to a common, public network – the Internet.
2. Number of nodes communicating may (and certainly will) change in time, some of them may be temporary unavailable.
3. A node may contact any other node in the system but typically it does not contact all the other nodes.

Chapter 4

Similar projects

This chapter describes two commercial solutions which are aimed at supporting the procedures which HEIs need to take between each other in order to successfully exchange mobility students.

4.1. Products of QS unisolution

4.1.1. moveon

Overview

QS unisolution is a company which emerged as a union of unisolution – a company grown on experience and work of two members of IRO at Technical University Darmstadt and QS – a company targeted at linking graduates, business schools, universities and employers in order to help with education and career-related decisions as well as recruiting the most promising graduates.

QS unisolution offers a commercial product called moveon, intended to fully aid in the field of international relations management. Moveon is an integrated system which is advertised as a complete solution for administration and management of international cooperations and exchange programmes. It supports following tasks:

- planning and organization of agreements and cooperations,
- marketing of the institution and its offer,
- daily activities connected to the mobility process,
- collecting statistical data in order to conduct analyses,
- generating reports for National Agency.

Moveon is currently being used by about 300 higher education institutions mainly in Europe – see Table 4.1 for moveon usage statistics per country. The software is available in 5 language versions.

Moveon acts as a stand-alone web application which is accessible for both HEI representatives and students. It is possible to customize moveon interface to resemble the layout and style of existing web applications and sites served by a HEI.

Functionality is divided into 3 so-called e-procedures: e-nominations, e-agreements, e-transcripts. For all the procedures moveon acts as a central place of the whole process –

Country	No. institutions using moveon
Belgium	9
France	94
Germany	140
Spain	9
Sweden	11
Switzerland	7
Turkey	3
United Kingdom	7

Table 4.1: Moveon usage in chosen countries, based on [UNICON]

neither of the cooperating HEIs can directly reuse the data from their SIMSes. The data need to entered by students and IRO representatives through publicly available web interface. All communication about events which occur in the system is carried via e-mail (generated from templates).

As regards integration, any data needed for mobility have to be either entered manually by HEI employees (typically IRO coordinator) or migrated from local, custom SIMS to moveon. Retyping the data from local SIMS may be cumbersome thus the functionality of migrations appears more practical. It is not, however, a persistent type of integration since migrations need to be carried by a unisolutions specialist every time it is needed to bring the two systems to sync – this is of course a commercial service. Other existing possibilities of data export formats are: MS Excel format and PDF for printable LAs and ToRs which are created according to ECTS regulations.

It is worth noticing that moveon has web service support, but for e-nominations procedure only. Unfortunately, the format of such exchange is not actually open and defined in a straightforward way because the data is in most cases coded into `xsd:base64Binary` type. Such practice violates the so called "spirit of XML"[1] and makes it much more difficult to create an interoperable implementation. Without any documentation, the only way to discover the format is to use reverse engineering techniques which may be ineffective.

E-procedures

e-agreements E-agreements are intended to automate the negotiation or renewal of bilateral agreements between partner institutions and enable them to generate and print the agreements with one click. Both partner institutions can work on the same agreement online, get an up-to-date list of all agreements and track the status of each agreement. During the whole process the communication between partner institutions is e-mail-driven. The procedure is as follows:

1. either of the cooperating institutions proposes an agreement,

2. both institutions make consecutive corrections to the proposed agreement,

[1] Any IT standard formal or informal (i.e. technology, programming language, data format) is usually supported by a community of authors, developers and users whose knowledge, experience and expertise expresses through some general guidelines and best practices on the usage of that particular standard. XML is also the case.

3. when the conditions are mutually accepted, either of the HEIs prints the agreement, signs it and sends to the other HEI,

4. the document of agreement gets co-signed and one copy is sent back.

A schematic overview of e-agreements process is shown in Figure 4.1.

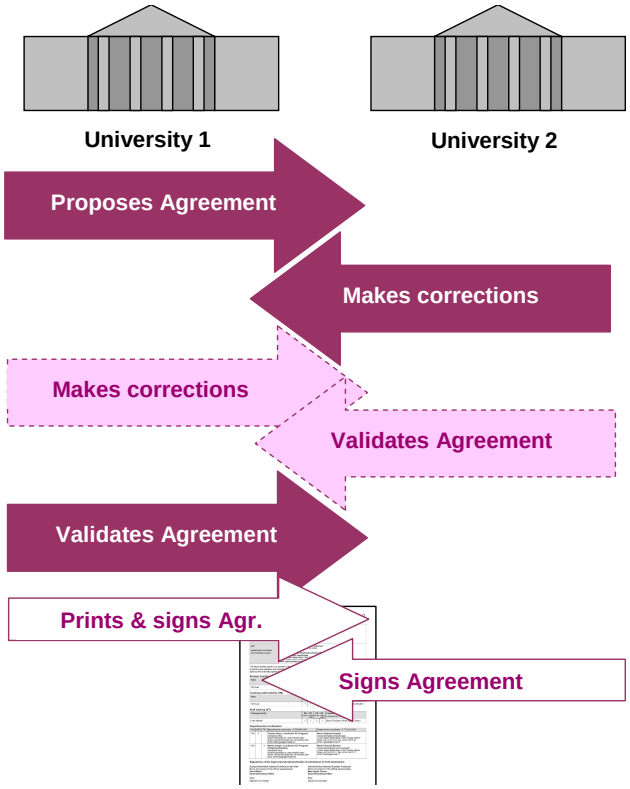

Figure 4.1: e-agreements workflow, taken from [UNICON]

e-nominations This is a procedure of nominating students for a mobility offer. The procedure is capable of dealing with two possible cases: outgoing and incoming students. A general workflow of this process is shown in Figure 4.2 – more details about how this workflow is handled by the software are listed here:

1. an employee of IRO defines mobility offers based on agreements with partner HEIs,

2. students create their accounts and apply for a mobility by filling appropriate forms with their personal data, information about their education, language skills, etc.,

3. coordinator reviews the applications and decides who is going to be nominated,

4. partner HEI is informed about nominations by automatically generated e-mail,

5. partner coordinator follows a link to a moveon webpage where they look up the data supplied by nominated students and inform them about decision and further procedure required,

6. students send application according to the demands of partner HEI.

It is possible to store information about calculated grants in the system, so at the end of mobility an automatically generated report for National Agency is available.

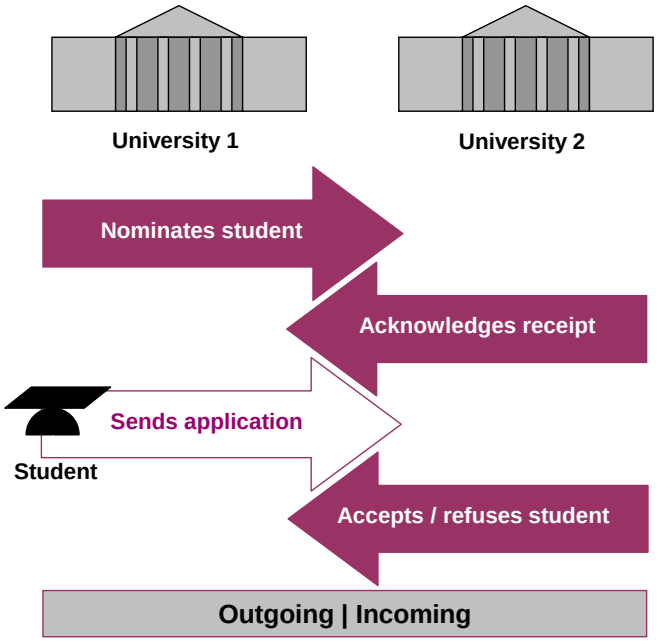

Figure 4.2: e-nominations workflow, taken from [UNICON]

e-transcripts This is a procedure of creating a ToR document. The procedure obviously involves a previously agreed LA which is complemented with grades entered by the partner administrative staff. Finally, a PDF document is generated and a printed version gets signed and sent by the partner institution to the home institution of the student. A schematic diagram of this process is presented in Figure 4.3.

4.1.2. moveonnet

Moveonnet is a worldwide portal (directory) of Higher Education which is mainly targeted at providing information about:

- guides on education systems, institutions and study programmes,
- HEIs including general information, contacts, list of partners, information for exchange students, ranking positions, location on a map, etc.
- countries including general information, regions/states, higher education system, institution types and list of institutions,
- documents on the internationalization of higher education.

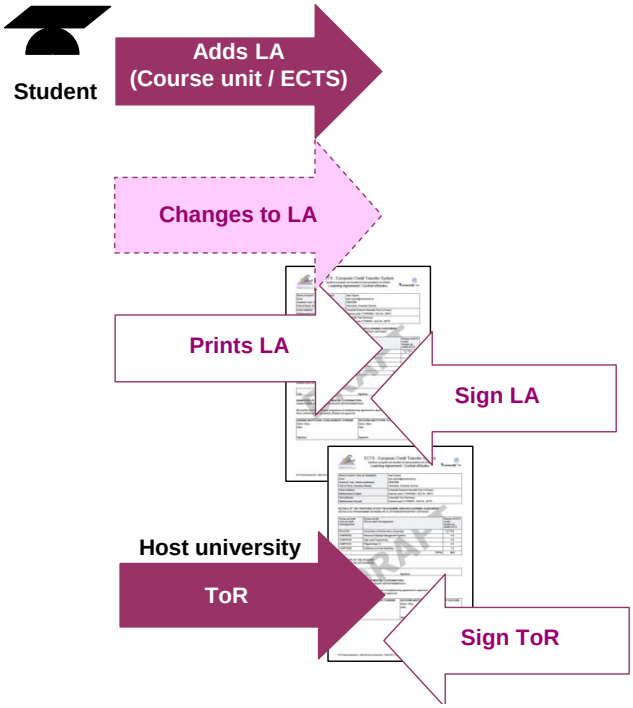

Figure 4.3: e-transcripts workflow, taken from [UNICON]

The idea behind moveonnet is to provide a searchable catalogue where a student could browse all the relevant information in one place and HEIs could communicate, make agreements, present their offer. It is worth noticing that with moveonnet it is possible to perform the same e-procedures for IROs as moveon provides, except for e-transcript which at the time of this being written was under development. It is also possible to integrate a moveon installation with moveonnet membership to have the best of two approaches – a dedicated, customized stand-alone system as well as promote one's offer amongst competitors.

The functionality is centered around the following areas:

- cooperations, agreements: exchange possibilities, publication of mobility offer,
- mobility: online application, selection, learning agreement, transcript of records, publication of reports,
- finances: fundings, grants, grant holders status, calculation of grants, automated payments, financial reports, export to national agencies,
- support functions: correspondence, reports, statistics and indicators.

In order to use the e-procedures the data needs to be entered to the system by filling its web forms by hand. It is a particularly superfluous activity when the data demanded is already stored in HEI's local SIMS. According to Ms. Klementyna Kielak, an employee of IRO at the University of Warsaw, that is not the only problem – the e-nominations procedure often requires data which may be not gathered by a HEI or be irrelevant at that stage.

4.2. Europass Mobility System

In order to facilitate the lifecycle (creation, completion and issuing) of Europass Mobility documents, Cedefop delegated development of a dedicated distributed system to Quality & Reliability company. As a result, Europass Mobility System (EMS) [SITI06] has been created.

The Europass Mobility System has been designed as a multi-node, hierarchical distributed system. Nodes, which contain all the database and application components required to run the Europass Mobility System in an autonomous fashion, are installed in each country of EU by local National Europass Centres (NECs). Larger countries may have more than one system, e.g. HEIs may host their own instances. The communication is carried through web services in secure channel. The nodes are supported by a central naming service, called the Central Authority node and hosted by Cedefop, which keeps the addresses of the nodes so they may communicate to each other. Each new node needs to be registered within the Central Authority node in order to let all the nodes in the network become aware of the newcomer. The system is accessible to humans via a web interface.

The system supports the following workflow of issuing a Europass Mobility document:

1. a home institution connects to the web interface of the local NEC's EMS node and initiates a new Europass mobility experience for its student,
2. the home institution prepares a provisional document with suitable data regarding the mobility experience filled in (if the student is already known to that NEC's node, data need not to be reentered),
3. the home institution locates the partner institution in EMS by using the search facility, it can do this by searching by keyword, by country, etc., then it sends the semi-completed document to the partner,
4. when the mobility experience is completed, the partner institution completes its part (e.g. Transcript of Records and/or Skills and Competences acquired during the experience), signs it digitally and sends the document back to the sending partner,
5. the home institution then issues the Europass Mobility document to the student, in the form of a digitally signed PDF file enriched with an XML attachment.
6. the Europass Mobility document is finally archived and can be retrieved at any time, e.g. for re-issuing.

The validity of documents issued by the system can be checked in two ways:

1. by verifying them directly against the appropriate root CA certificate installed locally,
2. or by verifying at the Europass website – the document needs to be uploaded.

The Europass Mobility System is claimed to bring the following benefits:

- availability of full documentation (including local and global statistics) for all European mobility experiences,
- reduction of workload and paperwork involved in completing a Europass Mobility document,
- promotion of citizen's mobility in Europe.

It is apparent that a substantial part of the Europass Mobility documents managed by EMS is the Erasmus programme mobilities between HEIs. Similar issues regarding problem domain along with some architectural and implementation similarities (network of nodes communicating via web services) build the potential to work on further integration and standardization of the two (to some extent) complementary solutions. There is an informal initiative to achieve a high level of interoperability between EMS and SIMSes used by European universities. Cooperation between Cedefop and SIMS vendors (participants of the Mobility Project) creates an opportunity for common efforts in widening Cedefop's standardization efforts with mobility scenarios stated in this document.

Chapter 5

Architecture and design

Requirements analysis leads to a conclusion that to obtain the goals of the Mobility Project we need a distributed architecture which has the following characteristics:

- any two nodes should contact directly – the only entities involved are the two HEIs which carry the student mobility initiative; there is no need for an agent which sits in the middle,
- existence of a single point of failure is discouraged – a consequence of the former point,
- any two nodes are totally equivalent including but not limiting the available kinds of data exchange, data transmission initiation, etc.

There are also other factors worth noticing:

- amounts of data are not very significant and the data is not exchanged continuously,
- flash crowd is not likely to happen.

This leads to a conclusion that we need a P2P-like architecture where every node acts as a server and client at the same time. We need to define the protocol used to perform data exchange. As a platform for creating one, a SOAP-based [SOAP11] web service defined in terms of a WSDL [WSDL] document seems a reasonable choice. Web services offer a great level of abstraction from the underlying network transport considerations and issues. Web services make use of well-known standards such as HTTP or XML. They have become very popular and today are recognized as an undisputed industry standard. They are also supported by virtually any commonly-used programming language. The last is true for the 1.1 revision of the standard. The newest version – 2.0 – has much less support among web services frameworks, therefore WSDL 1.1 is chosen.

Besides SOAP web services, there are other styles of web services as well, namely XML-RPC and REST. The former one has been discouraged due to the fact that it has serious limitations, one to mention is a lack of support for developer-defined data types or character set. REST, on the other hand, is rather a set of principles than a formal standard and it proved advantageous in scenarios which assumed rapid development of a web service interface and did not impose a requirement of being compliant with a very formal, clearly defined specification.

Since the number and addresses of endpoints may change, there is a need to maintain a database of them. It is always possible to develop a custom solution for that but there exists a core web service standard for storage of web services' metadata named UDDI (Universal

Description Discovery and Integration) [UDDITC] – for further reference see the web page of UDDI Technical Committee [UDDITC].

There is a dilemma whether to provide one global UDDI registry instance or delegate the requirement of having a private registry to each node. It is currently not quite clear how this should be solved. A single registry has an advantage of keeping all the information in one place but it is an apparent single point of failure. There is also a major concern of the body responsible for maintaining the registry and what procedures should be used when its data need to be updated. In the light of these issues the concept of private registry appears presently more appropriate. However, as further requirements are discovered it may turn out that this decision needs to be verified.

The software is divided into 4 independent modules: 2 essential transport modules – one for client and one for server side of communication and 2 web interfaces for each transport module. An overview is presented in Figure 5.1. The modules are all located in the middle part of the diagram, which is labelled "Transport middleware". The top box represents client transport and web modules while the one at the bottom – server modules respectively.

The software acts as a generic node. The nodes can communicate to each other. A schematic diagram of a graph of nodes communicating is shown in Figure 5.2. The circles in the diagram denote individual nodes.

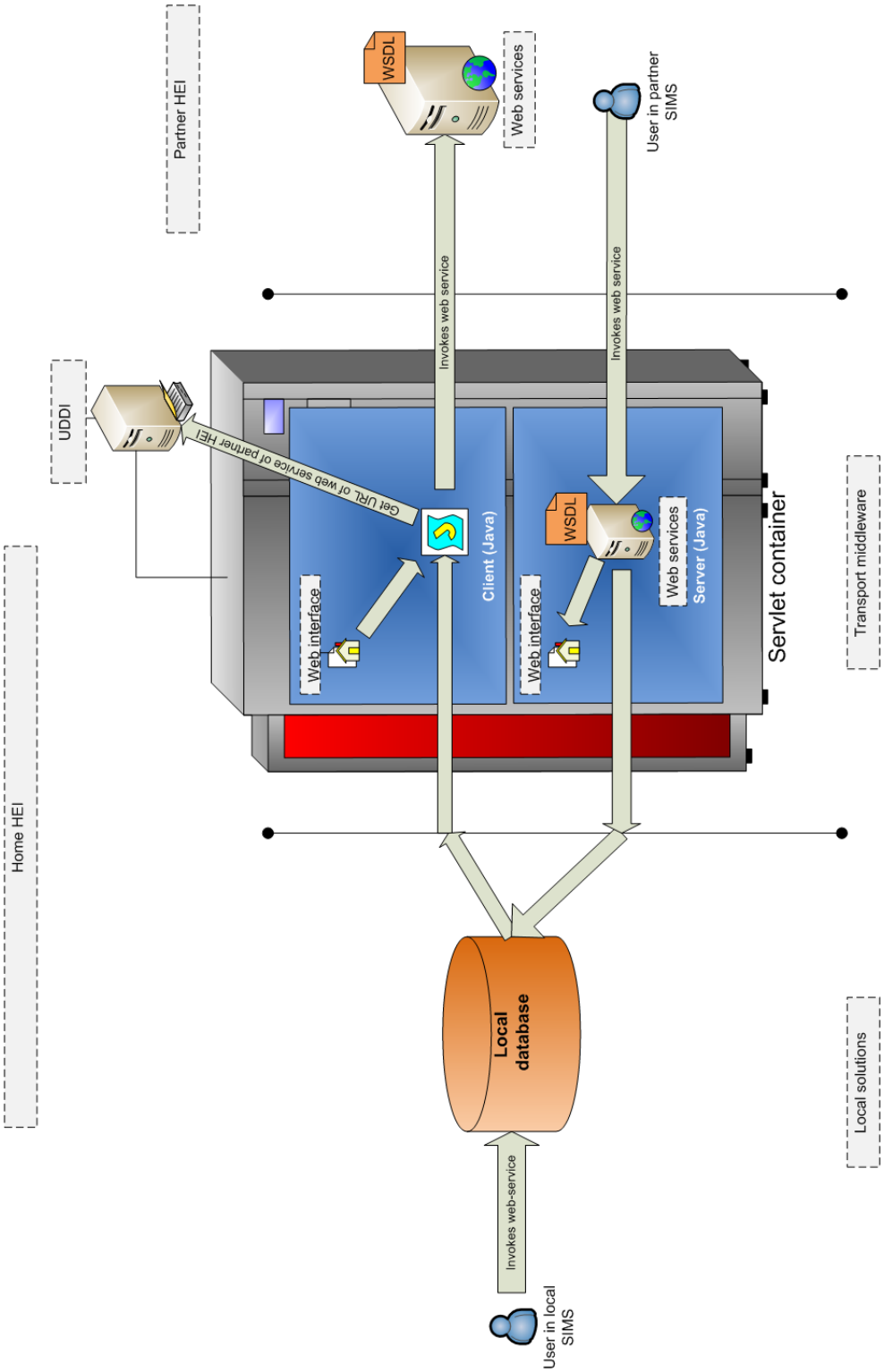

Figure 5.1: Architecture of the system

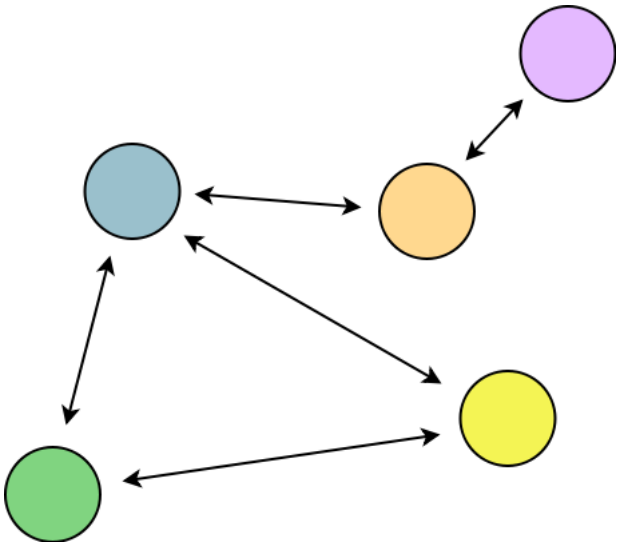

Figure 5.2: Nodes in the system

Chapter 6

WSDL Document

This chapter contains a detailed description of the designed WSDL document describing the vocabulary and behaviour of the exposed web service.

6.1. WSDL standard overview

This section is intended to describe the purpose of the WSDL standard along with other standards related to it.

6.1.1. WSDL

WSDL is an XML format of network services description. The network services are seen as a set of endpoints exchanging messages. The messages use vocabulary defined in the `types` section of WSDL with XML Schema language [XMLSCHEMA]. The operations and messages are abstract by themselves. The binding to a specific protocol is defined separately. However, WSDL defines a binding for SOAP 1.1 protocol. WSDL document consists of several consecutive parts shown in Figure 6.1.

6.1.2. XML Schema

XML Schema definition language offers facilities to describe the structure and constrain the contents of XML 1.0 documents, including those which exploit the XML Namespace facility. It extends in a considerable way the capabilities of XML 1.0 DTDs (Document Type Definitions). The main difference is an extensive and extendable type system which provides substantial amount of flexibility in refining one's own vocabulary. Thanks to this a great deal of work involved in the process of document validation is delegated to XML parser.

6.1.3. SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that defines the contents of a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols but the standard describes how to use SOAP in combination with HTTP only.

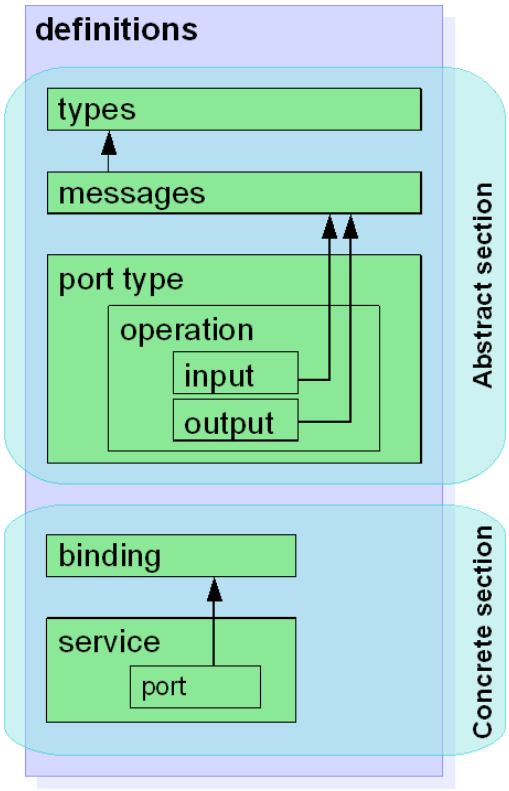

Figure 6.1: Overview of WSDL 1.1. Based on: `http://en.wikipedia.org/wiki/File:WSDL_11vs20.png`

6.2. Overview

There are some key aspects to consider when proposing a new standard. Firstly, there is a need to provide a vocabulary of well-defined terms which model the problem domain well. Secondly, there is a need to avoid unnecessary complexity and keep things possibly simple according to the rule of Ockham's razor. Thirdly, a good standard leaves a way to extend it easily. And the last but not least, employment of existing standards and practices to the maximum extent would be highly desired, so the effort of converting the data already formatted in compliance to standards is minimized. The level of adoption of a standard may be significantly influenced by considerations regarding potential integration easiness.

I decided to reuse ideas of SCHAC (see Section 2.3) because, despite its deficiencies described in Section 2.3, its purpose is the closest to the problem to be solved. Besides, SCHAC leverages ISO and RFC norms and provides precise definitions.

I have also made a few definitions connected with domain classification, grade, ECTS credits to some extent similar to those found in Europass (Mobility).

6.3. Conventions

Note: It is assumed that `xsd` prefix is bound to `http://www.w3.org/2001/XMLSchema` namespace, whereas definitions (types, elements, etc.) which belong to WSDL's target namespace `http://mobility.usos.edu.pl` do not contain any prefix.

I followed several conventions regarding WSDL document:

- all names are in lower camel case (example: anotherNameInCamelCase),
- names of types end with 'T',
- names of groups end with 'G',
- operations which require personal data are constructed in such a way that:
  - personal data comes first wrapped in a `uniquePersonalData` element, it is constrained with a `xsd:key` construct placed on `studentId` and `employeeId` for students and employees respectively,
  - the actual data sent by a particular method has references to the `uniquePersonalData` element through `xsd:keyref`s,
- all operations are request-response, i.e. the endpoint receives a message and sends a correlated message (see `http://www.w3.org/TR/wsdl#_porttypes`),
- all operations throw an `errorMessage` fault,
- messages used to compose an operation have exactly one part, i.e. `message` element has exactly one `part` child,
- input messages having an agreement context (input messages of so called "agreement-dependent methods") have an `agreementId` element as a child of the document root, so XPath[1] query `/<requestName>/agreementId/homeId/organizationId`, where <requestName> is the name of the current request, evaluates to a client `organizationId` value.

XML Schema groups were introduced mainly for the purpose of clarity of XML instances. The same ideas could be modeled without using groups. Although the instance would then have a more verbose, nested structure, it would rather not be more understandable for humans. These pieces simply require a bit of commentary which cannot be reasonably expressed with a short element or type name.

Some of the types are equipped with key and keyref constraints. These constructs prevent unnecessary data repetition and some basic data inconsistencies.

6.4. Vocabulary

6.4.1. Helper types

- `nameT` – general name without whitespace;
- `internationalizedStringT` – list of strings in different languages; see Figure 6.2;

Figure 6.2: `internationalizedStringT` type

[1]For simplicity we assume that the query and the context node share the same, proper namespace.

- `academicPeriodT` – enumeration of possible parts into which academic year is divided; see Table 6.1;

Value	Description
Y	Full academic year
S1	Winter semester
S2	Summer semester
T1	First trimester
T2	Second trimester
T3	Third trimester

Table 6.1: Possible values of `academicPeriodT` type

- `academicPeriodSinceT` – `academicPeriodT` value in a specific academic year plus duration expressed in units imposed by `academicPeriodT`; see Figure 6.3;

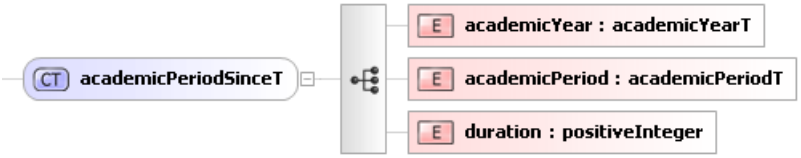

Figure 6.3: `academicPeriodSinceT` type

- `addressT` – sequence describing an address; see Figure 6.4;

Figure 6.4: `addressT` type

- `emailT` – string restricted to contain a valid e-mail address;
- `errorT` – error code and message; see Figure 6.5;

6.4.2. Identifiers

For these types of information which have an ISO standard, I decided to use them (as also did SCHAC and Europass). These types are shown in Table 6.2.

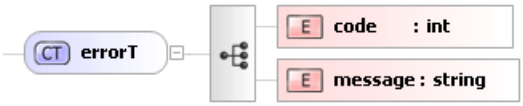

Figure 6.5: `errorT` type

WSDL type	Description	ISO standard	Example value(s)
`countryCodeT`	Country	ISO 3166-1 alpha-2	"PL", "IT", "GB"
`languageCodeT`	Language	ISO 639-1	"pl", "en", "de"
`genderT`	Gender	ISO/IEC 5218:2004	"1", "2"

Table 6.2: ISO-based types

Note: The `xsd:date` data type uses format conforming to ISO 8601:2004(E) (it is a subset). For reference see: `http://www.w3.org/TR/xmlschema-2/#isoformats`.

There is also a need to uniquely identify a HEI. Following SCHAC's `schacHomeOrganization`, I defined `organizationIdT` type which points to `domainT` type defined according to domain names as described in RFC 1035 [RFC1035]. There is an issue of what the values for certain organizations actually are. It may be safely assumed that the domain name used for Internet services (e.g. website) of an organization would serve as a valid and pretty straightforward identifier. Some examples are given in Table 6.3.

Organization name	Web page address	`organizationIdT` value
University of Warsaw	www.uw.edu.pl	uw.edu.pl
Warsaw University of Technology	www.pw.edu.pl	pw.edu.pl
University of Parma	www.unipr.it	unipr.it

Table 6.3: Example values of `organizationIdT` type

Although the project currently concentrates on only one type of organization – namely an HEI – it may be extended in the future for a wider range of organizations. In order to provide organization type information I introduce a SCHAC's `schacHomeOrganizationType`-based type `organizationTypeT`. Its values are a result of the following transformation of original SCHAC values:

1. strip `urn:mace:terena.org:schac:homeOrganizationType` prefix with a consecutive colon,
2. split the result through colons,
3. arrange the result into a two-element sequence.

SCHAC defines possible values at `http://www.terena.org/registry/terena.org/schac/homeOrganizationType/` and a list of equivalent values for `organizationTypeT` is given in Table 6.4[2].

[2]The list provided in the first column of the table does not obviously contain valid XML values. The format

Value ([nationalNamespace, value])	Description
[eu, higherEducationInstitution]	HEI
[eu, educationInstitution]	Education institution with multiple levels of education
[int, NREN]	National research and education network
[int, universityHospital]	University hospital
[int, NRENAffiliate]	Affiliate of a national research and education network
[int, other]	Other organization

Table 6.4: Possible values of `organizationTypeT` type

Identifiers of students and employees are coded as `organizationalPersonalIdT` type which consists of an `organizationIdT`-like prefix, a colon and an id given by the organization, e.g. uw.edu.pl:60225. Type `organizationalPersonalIdT` would have a more elegant definition, if XML Schema were able to use values of complex types with key definitions (`xsd:key`).

In case a national id of a person is needed, there is the type `nationalPersonalIdT` which is a string based on SCHAC's attribute `schacPersonalUniqueID` but does not use a prefix `urn:mace:terena.org:schac:personalUniqueID` and id type information. Value consists of a country prefix (exactly as in `countryCodeT`), a colon and an identification number. The national identifier is the most convenient kind of personal identifier possible but due to the legal issues connected with personal data protection regulations in some countries it may not be possible to share such data. Therefore its use is optional, see `personalCharacteristicsT` type in Figure 6.8.

Type of national id	Example value
Finnish FIC	fi:260667-123F
Spanish NIF	es:31241312L
Swedish NIN	se:12345678
Polish PESEL	pl:77121201230

Table 6.5: Example values of `nationalPersonalIdT` type

Table 6.5 shows a few examples.

Courses are identified by codes unique within an organization but use of pairs [organizationId, courseCode] is not a necessity due to the fact that course codes are always placed within a context of an organization.

Type `studyLevelT` identifies a required level of study of a nominated student. Possible values are presented in Table 6.6.

In order to identify a bilateral agreement between organizations, type `agreementIdT` is provided (shown in Figure 6.6, example in Listing 6.1). It consists of two sub-ids for both organizations. These sub-ids comprise of an `organizationIdT` value along with an internal agreement identifier – `localAgreementIdT` – shown in Figure 6.7, example in Listing 6.2.

supplied in the header in brackets reflects the names of XML element nodes while the actual values contained in the table are the possible text content of the respective element nodes. The intention was to simplify notation in order to keep the table small.

Value	Meaning
1	Student
2	Master student
3	PhD. student
4	Student of any level

Table 6.6: Possible values of `studyLevelT` type

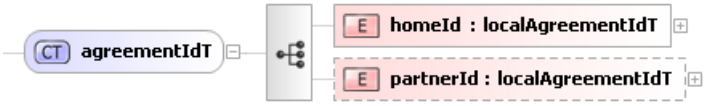

Figure 6.6: `agreementIdT` type

```
<tns:homeId>
    <tns:organizationId>unipr.it</tns:organizationId>
    <tns:value>1207/E/XI08</tns:value>
</tns:homeId>
<tns:partnerId>
    <tns:organizationId>uw.edu.pl</tns:organizationId>
    <tns:value>982/E/II08</tns:value>
</tns:partnerId>
```

Listing 6.1: `agreementIdT` example

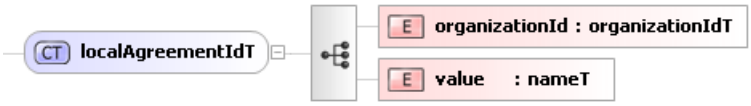

Figure 6.7: `localAgreementIdT` type

```
<tns:organizationId>uw.edu.pl</tns:organizationId>
<tns:value>982/E/II08</tns:value>
```

Listing 6.2: `localAgreementIdT` example

6.4.3. Personal data types

- `personalCharacteristicsT` – sequence of basic personal data; definition in Figure 6.8;
- `personalPositionT` – free format string containing a description of a person's position inside an institution;
- `employeePersonalCharacteristicsT` – extension of `personalCharacteristicsT` which adds data specific for an employee; definition in Figure 6.9, example in Listing 6.3;

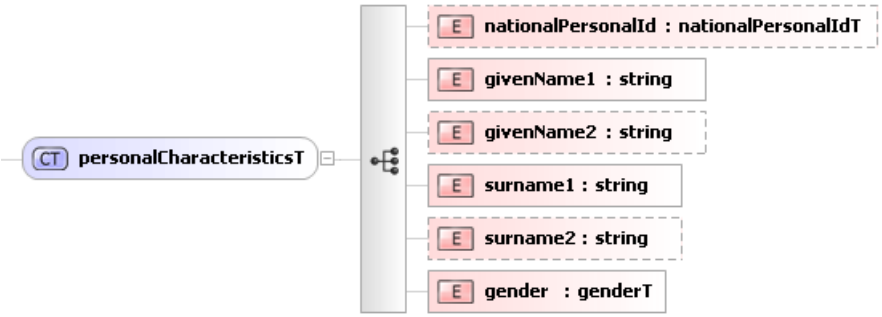

Figure 6.8: `personalCharacteristicsT` type

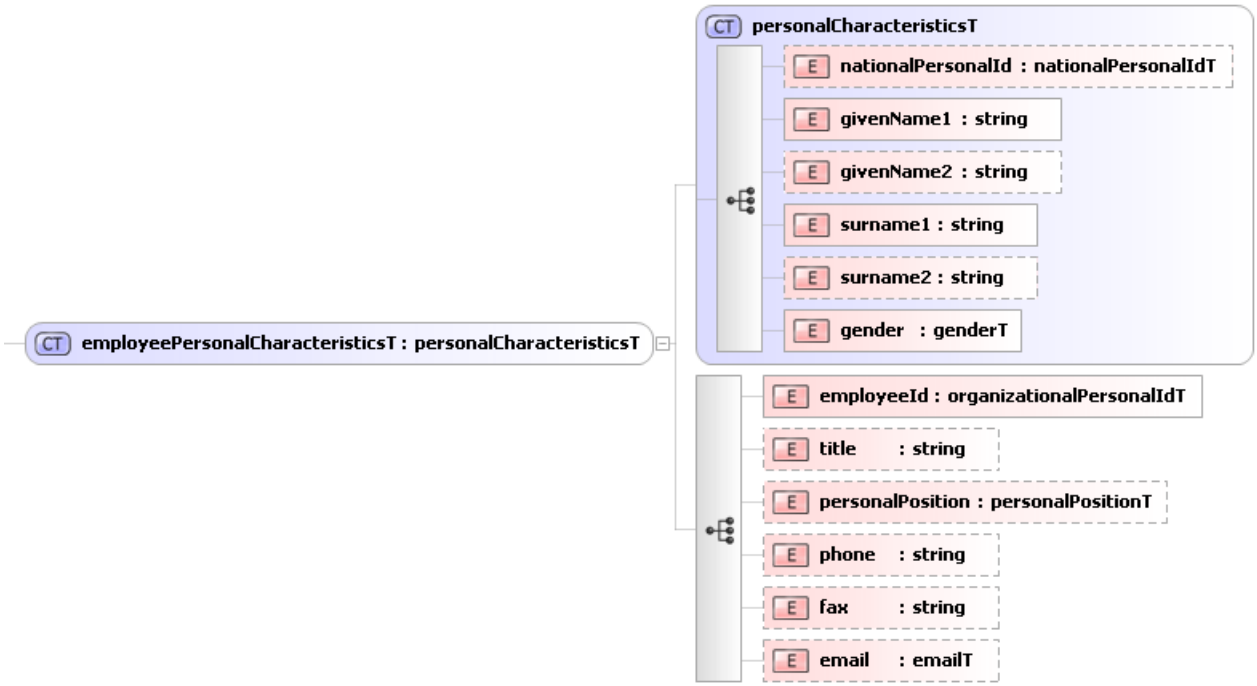

Figure 6.9: `employeePersonalCharacteristicsT` type

```
<tns:givenName1>Marcin</tns:givenName1>
<tns:surname1>Benke</tns:surname1>
<tns:gender>1</tns:gender>
<tns:employeeId>uw.edu.pl:1234</tns:employeeId>
```

Listing 6.3: `employeePersonalCharacteristicsT` example

- `studentPersonalCharacteristicsT` – extension of `personalCharacteristicsT` which adds data specific for a student; definition in Figure 6.10, example in Listing 6.4;

```
<tns:nationalPersonalId>IT:RCLFBA70D19D708A</tns:nationalPersonalId>
<tns:givenName1>Fabio</tns:givenName1>
<tns:surname1>Arcella</tns:surname1>
<tns:gender>1</tns:gender>
<tns:studentId>unipr.it:9001</tns:studentId>
<tns:dateOfBirth>1970-04-19</tns:dateOfBirth>
<tns:placeOfBirth>Formia, IT</tns:placeOfBirth>
<tns:email>f.arcella@kion.it</tns:email>
<tns:permanentAddress>
    <tns:street>Via Cristoni</tns:street>
    <tns:houseNumber>70</tns:houseNumber>
    <tns:flatNumber/>
    <tns:postalCode>40033</tns:postalCode>
    <tns:city>Bologna</tns:city>
    <tns:country>IT</tns:country>
</tns:permanentAddress>
<tns:stationaryPhone>00390516111453</tns:stationaryPhone>
<tns:mobilePhone>00393489006216</tns:mobilePhone>
```

Listing 6.4: `studentPersonalCharacteristicsT` example

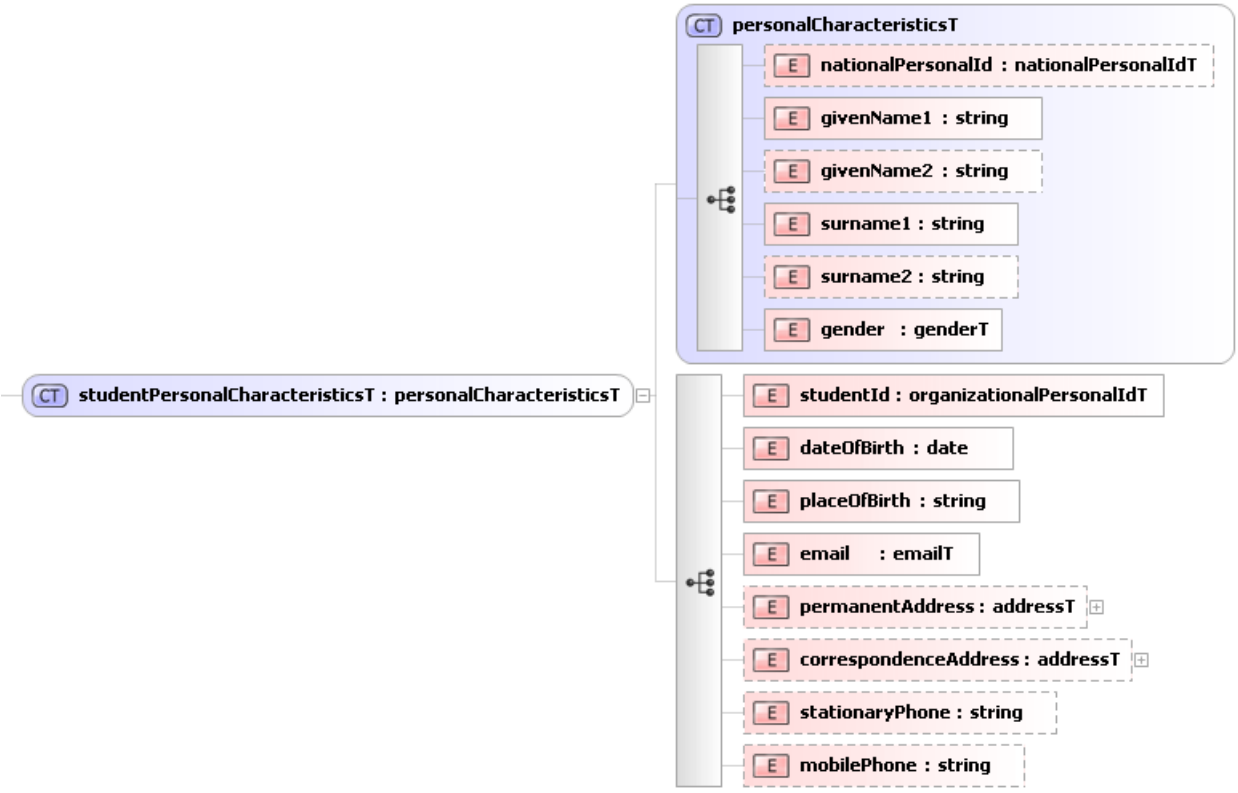

Figure 6.10: `studentPersonalCharacteristicsT` type

6.4.4. Organization-related types

- `organizationDataT` – structure describing an organization; type presented in Figure 6.11; example in Listing 6.5;

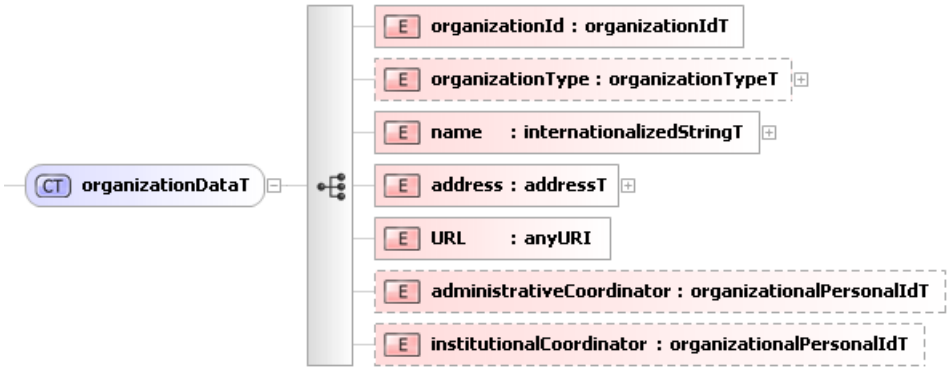

Figure 6.11: `organizationDataT` type

```
<tns:organizationId>uw.edu.pl</tns:organizationId>
<tns:name>
    <tns:value language="en">University of Warsaw</tns:value>
    <tns:value language="pl">Uniwersytet Warszawski</tns:value>
</tns:name>
<tns:address>
    <tns:street>Krakowskie Przedmiescie</tns:street>
    <tns:houseNumber>26/28</tns:houseNumber>
    <tns:postalCode>00-927</tns:postalCode>
    <tns:city>Warsaw</tns:city>
    <tns:country>PL</tns:country>
</tns:address>
<tns:URL>www.uw.edu.pl</tns:URL>
<tns:administrativeCoordinator>uw.edu.pl:45065</tns:administrativeCoordinator>
<tns:institutionalCoordinator>uw.edu.pl:112233</tns:institutionalCoordinator>
```

Listing 6.5: `organizationDataT` example

6.4.5. Course-related types

- `subjectAreaCodeT` – code of a study subject area within a specific classification; classifications supported are: EU, ISCED97 (case insensitive); EU denotes Socrates/Erasmus coding while ISCED97 – UNESCO ISCED 97 standard [ISCED]; examples:
  - EU – "11.3" corresponds to Computer Science, "15.1" – to Journalism,
  - ISCED97 – "48" denotes Computing, "32" refers to Journalism and information;

  see definition in Figure 6.12; any element of this type is constrained with a key on `classification` attribute;

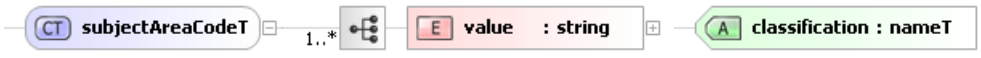

Figure 6.12: `subjectAreaCodeT` type

- `courseCodeT` – organization's internal course code, a wrapper of `nameT`;

- `studyCreditsT` – credits of a given type granted for completing the course; shown in Figure 6.13; implementations are required to recognize value ECTS of `unit` attribute; any element of type `studyCreditsT` is constrained with a key on `unit` attribute;

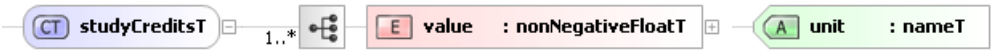

Figure 6.13: `studyCreditsT` type

- `contactHoursT` – a non-negative float value indicating total number of hours of a course (lectures, classes, labs), e.g. 30, 60;

- `courseDataT` – description of a course; definition in Figure 6.14; example in Listing 6.6;

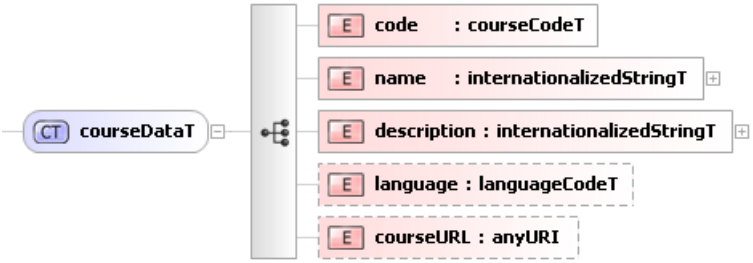

Figure 6.14: `courseDataT` type

```
<tns:code>1000-234aMRJ</tns:code>
<tns:name>
    <tns:value language="en">Compiler construction</tns:value>
    <tns:value language="pl">Metody realizacji jezykow programowania</tns:value>
</tns:name>
<tns:description>
    <tns:value language="en">Structure of a compiler. Phases of compilation: analysis (
        lexical analysis, syntax analysis, context vel semantic analysis), synthesis (code
        generation, optimization, loading). Data structures of a compiler (symbol table et al
        .).</tns:value>
    <tns:value language="pl">Struktura kompilatora. Fazy kompilacji: analiza leksykalna,
        skladniowa, kontekstowa, generowanie kodu, optymalizacja, skladanie kodu.
        Struktury danych kompilatora (tablica nazw, tablica symboli).</tns:value>
</tns:description>
<tns:language>pl</tns:language>
<tns:courseURL>https://usosweb.mimuw.edu.pl/kontroler.php?_action=actionx%3Akatalog2
    %2Fprzedmioty%2FpokazPrzedmiot(prz_kod%3A1000-234aMRJ)&amp;lang=2</
    tns:courseURL>
```

Listing 6.6: `courseDataT` example

- `courseInstanceT` – an instance of a course is distinguished by a particular period, semester or academic year, it also has a lecturer and amount of study credits defined; definition in Figure 6.15;

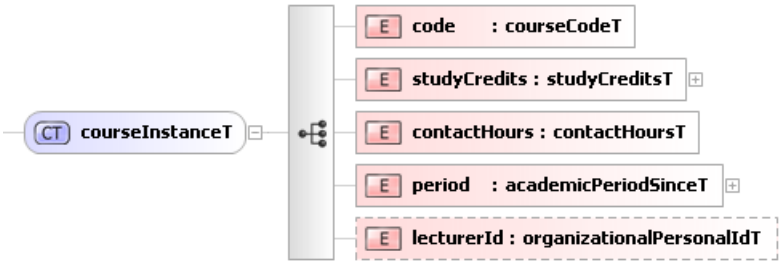

Figure 6.15: `courseInstanceT` type

```
<tns:code>1000−2M03DM</tns:code>
<tns:studyCredits>
    <tns:value unit=”ects”>5</tns:value>
</tns:studyCredits>
<tns:contactHours>30</tns:contactHours>
<tns:period>
    <tns:academicYear>2008</tns:academicYear>
    <tns:academicPeriod>S1</tns:academicPeriod>
    <tns:duration>1</tns:duration>
</tns:period>
<tns:lecturerId>uw.edu.pl:1111</tns:lecturerId>
```

Listing 6.7: `courseInstanceT` example

- `gradeT` – grade from a given grading scheme; presented in Figure 6.16; constrained with a key on `gradingScheme` attribute; supported values: ECTS (denoting ECTS Grading Scheme – see p. 13) and local (indicating grading scheme of the issuer HEI);

Figure 6.16: `gradeT` type

```
<tns:value gradingScheme=”local”>5</tns:value>
<tns:value gradingScheme=”ects”>A</tns:value>
```

Listing 6.8: `gradeT` example

- `academicYearT` – a wrapper of `xsd:gYear`, e.g. 2009.

6.4.6. Agreement-related types

- `cooperationConditionsIdT` – an artificial identifier for a set of conditions within an agreement between organizations; technically a positive integer constrained with a key to ensure its uniqueness.

6.5. Methods

Most methods have two versions, so any of the parties can initiate data exchange. A specific method call is supposed to complete successfully only when all prerequisite method calls have

already succeeded. Figure 6.17 shows method dependencies.

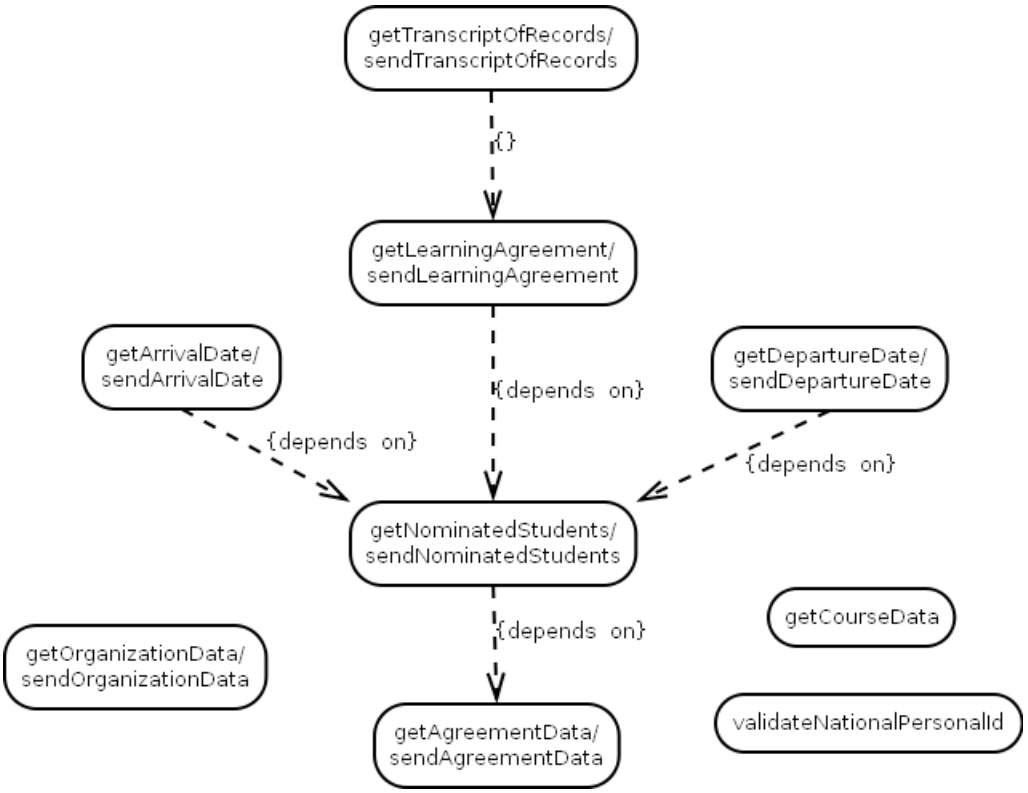

Figure 6.17: Method dependencies

All of the methods, excluding getCourseData and validateNationalPersonalId, need to include personal data. There is a possiblity that the same person may be referenced more than once, i.e. a lecturer leads multiple courses or one person is a coordinator and a lecturer at the same time. Each method including personal data is made of two parts:

1. a list of personal data items divided into students and employees,

2. the actual data carried by a particular method.

The personal data list is in each case constrained with two keys: one pointing to `employeeId` and the other to `studentId` values. Any definitions in the second part which include `employeeId` or `studentId` values have keyref constraints enforcing these values to reference elements in the first part. The construct described ensures that the data about each employee and student appears only once.

6.5.1. Agreement-independent methods

Methods listed in this section do not depend on a bilateral agreement.

sendOrganizationData/getOrganizationData

These methods exchange `organizationDataT` structures (Figure 6.11).

sendAgreementData/getAgreementData

These calls are responsible for exchanging contents of a bilateral agreement signed by two HEIs. An agreement modeled by `agreementData` (shown in Figure 6.18) includes:

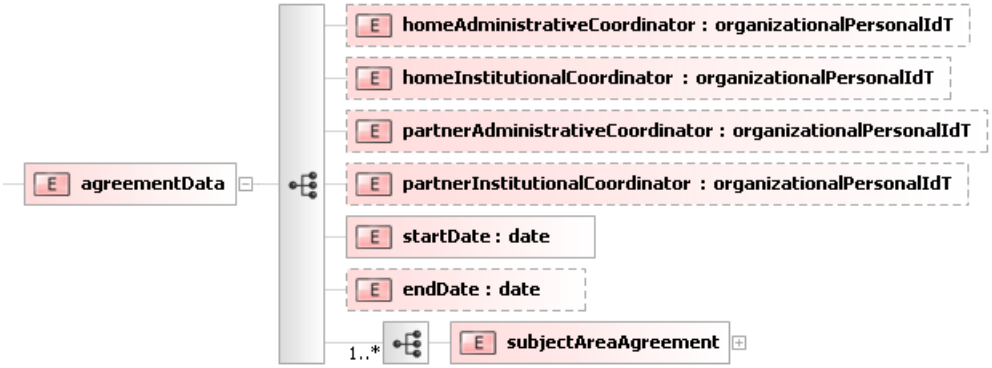

Figure 6.18: `agreementData` type

```
<tns:homeAdministrativeCoordinator>uw.edu.pl:60225</tns:homeAdministrativeCoordinator>
<tns:homeInstitutionalCoordinator>uw.edu.pl:45065</tns:homeInstitutionalCoordinator>
<tns:partnerAdministrativeCoordinator>unipr.it:1212</tns:partnerAdministrativeCoordinator>
<tns:partnerInstitutionalCoordinator>unipr.it:1313</tns:partnerInstitutionalCoordinator>
<tns:startDate>2008-10-01</tns:startDate>
<tns:endDate>2009-06-30</tns:endDate>
<tns:subjectAreaAgreement>
 ...
</tns:subjectAreaAgreement>
```

Listing 6.9: `agreementData` example

- id of an agreement (`agreementIdT` value – Figure 6.6),
- administrative and institutional coordinators at both sides,
- start and end dates of the agreement,
- lists of `subjectAreaAgreement` elements presented in Figure 6.19.

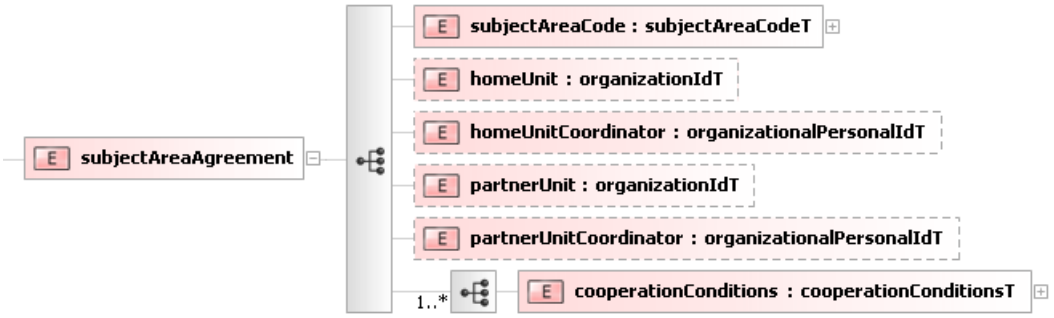

Figure 6.19: `subjectAreaAgreement` element

```
<tns:subjectAreaCode>
    <tns:value classification="eu">11.3</tns:value>
</tns:subjectAreaCode>
<tns:homeUnit>mimuw.edu.pl</tns:homeUnit>
<tns:homeUnitCoordinator>uw.edu.pl:270</tns:homeUnitCoordinator>
<tns:partnerUnit>scienze.unipr.it</tns:partnerUnit>
<tns:partnerUnitCoordinator>unipr.it:1212</tns:partnerUnitCoordinator>
<tns:cooperationConditions>
 ...
</tns:cooperationConditions>
```

Listing 6.10: `subjectAreaAgreement` example

One `subjectAreaAgreement` element describes agreement details per `subjectAreaCodeT` value (in Figure 6.12) and is further divided into `cooperationConditionsT` values (Figure 6.20). Each `cooperationConditionsT` value gathers a more detailed set of conditions agreed, i.e. number of places for students and number of months agreed for a specific study level. These conditions function within a defined time period (`startDate` and `endDate`) and specific subject area which is determined by a `subjectAreaCodeT` value found in the parent element.

A `subjectAreaAgreement` element also contains:

- an optional identifier of a unit, i.e. an organization contained within the main organization which is set with an `agreementIdT` value,
- an identifier of coordinator responsible for a given subject area.

The values may be obviously given for both parties. It has to be noted that the definition is supplied with a key constraint for `studyLevel` elements placed as children of all of the `cooperationConditions` elements.

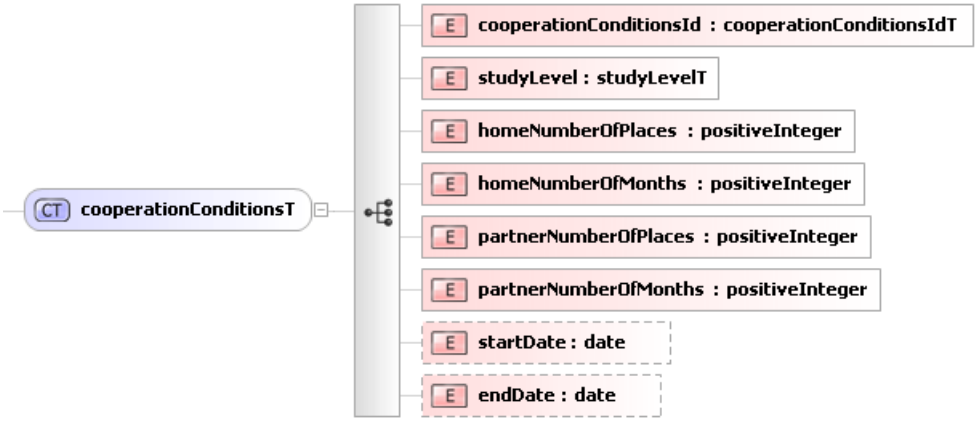

Figure 6.20: `cooperationConditionsT` type

```
<tns:cooperationConditions>
    <tns:cooperationConditionsId>1</tns:cooperationConditionsId>
    <tns:studyLevel>1</tns:studyLevel>
    <tns:homeNumberOfPlaces>3</tns:homeNumberOfPlaces>
    <tns:homeNumberOfMonths>5</tns:homeNumberOfMonths>
    <tns:partnerNumberOfPlaces>4</tns:partnerNumberOfPlaces>
    <tns:partnerNumberOfMonths>5</tns:partnerNumberOfMonths>
</tns:cooperationConditions>
```

Listing 6.11: `cooperationConditionsT` example

getCourseData

For a given list of `courseCodeT` values method returns a list of corresponding `courseInstanceT` values (Figure 6.15).

validateNationalPersonalId

An utility method to check whether a specific national identifier is valid. It obviously needs a `nationalPersonalIdT` value but apart from that some additional data necessary to carry the validation process may be added.

The format of that additional data is generic: a parameter name and a value should be given according to convention used by the implementer of this method – see Figure 6.21. For example, Polish PESEL is generated using information about a person's date of birth and their gender (see Listing 6.12 for a complete request example).

Figure 6.21: `additionalValidationData` type

```
<tns:validateNationalPersonalIdRequest>
    <tns:nationalPersonalId>PL:85102602439</tns:nationalPersonalId>
    <tns:additionalValidationData>
        <tns:parameter name="dateOfBirth">1985-10-26</tns:parameter>
        <tns:parameter name="gender">1</tns:parameter>
    </tns:additionalValidationData>
<tns:validateNationalPersonalIdRequest/>
```

Listing 6.12: `validateNationalPersonalIdRequest` example

6.5.2. Agreement-dependent methods

These methods exchange data in a context of an agreement – there is an assumption that both organizations have already exchanged the agreement through either sendAgreementData or getAgreementData method call and are aware of that agreement.

Methods: sendArrivalDate, getArrivalDate, sendDepartureDate, getDepartureDate, sendLearningAgreement, getLearningAgreement, sendTranscriptOfRecords, getTranscriptOfRecords are provided with a context of particular conditions within an agreement by means of `agreementAndCooperationConditionsContextG`. As it can be seen in Figure 6.22, the sequence comprises of an `agreementIdT` value and a `cooperationConditionsContextG` group.

The purpose of `cooperationConditionsContextG` group is to determine cooperation conditions and a particular duration in academic units. Because cooperation conditions are identified with a somehow artificial identifier, it is supplied in a form of `cooperationConditionsRedundantContextG` – presented in Figure 6.23 – which includes a `subjectAreaCodeT` and a `studyLevelT` value.

Although these values are obviously redundant with regard to the `cooperationConditionsId` which must have been accompanied with them when sendAgreementData or getAgreementData invocation occurred, they may be of some help when trying to determine the cooperation conditions and that term may be potentially perceived in a slightly different way by particular HEIs.

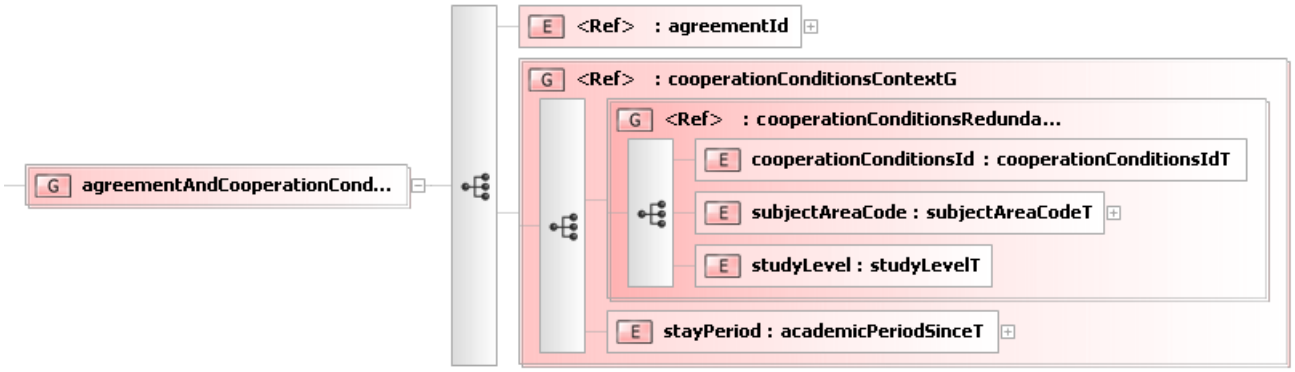

Figure 6.22: `agreementAndCooperationConditionsContextG` group

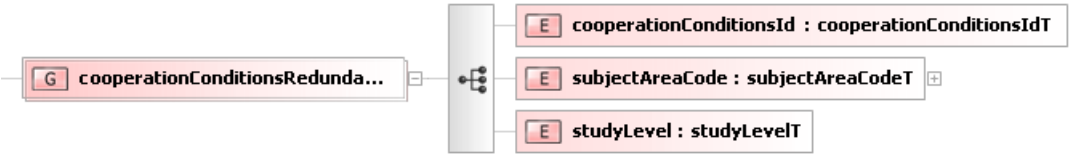

Figure 6.23: `cooperationConditionsRedundantContextG` group

sendNominatedStudents/getNominatedStudents

Basic semantics:

- sendNominatedStudents – send to partner organization a list of nominated students from home organization,
- getNominatedStudents – get from partner organization a list of nominated students from partner organization.

Lists of nominated students to be exchanged are contained in `nominations` element (see definition in Figure 6.24 and example in Listing 6.13 and grouped by `cooperationConditionsId`. Both methods need an `agreementIdT` (see Figure 6.6) and `academicYearT` value. The `cooperationConditionsRedundantContextG` group (see Figure 6.23) is optional and two cases are possible:

- if it is present, the scope of the call is narrowed to the `cooperationConditionsId` value (supplied with `cooperationConditionsRedundantContextG` group),
- if it is not present, it means that data for all `cooperationConditionsId` values mentioned in the applicable agreement should be provided.

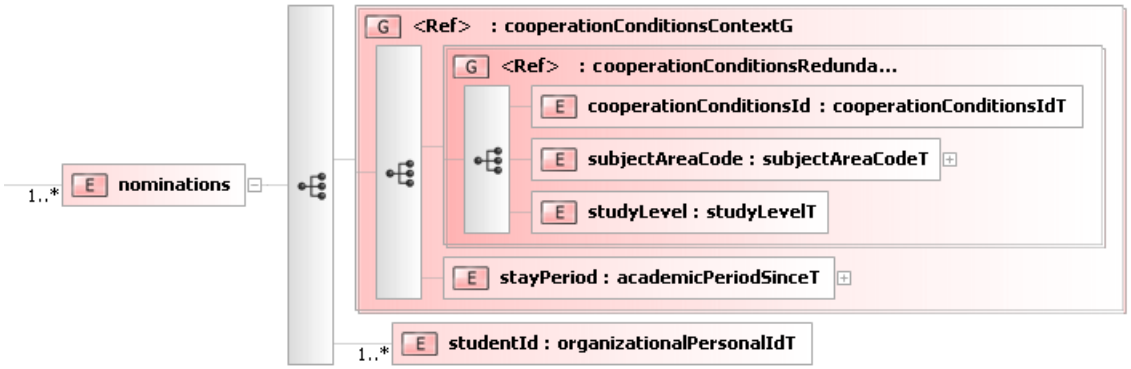

Figure 6.24: `nominations` element

```
<tns:nominations>
    <tns:cooperationConditionsId>1</tns:cooperationConditionsId>
    <tns:subjectAreaCode>
        <tns:value classification=”eu”>11.3</tns:value>
    </tns:subjectAreaCode>
    <tns:studyLevel>1</tns:studyLevel>
    <tns:stayPeriod>
        <tns:academicYear>2008</tns:academicYear>
        <tns:academicPeriod>S1</tns:academicPeriod>
        <tns:duration>1</tns:duration>
    </tns:stayPeriod>
    <tns:studentId>unipr.it:9001</tns:studentId>
    <tns:studentId>unipr.it:9002</tns:studentId>
    <tns:studentId>unipr.it:9003</tns:studentId>
</tns:nominations>
```

Listing 6.13: `nominations` example

sendArrivalDate/getArrivalDate, sendDepartureDate/getDepartureDate

Basic semantics:

- sendArrivalDate – send to partner organization arrival dates of partner organization's students at home organization,
- getArrivalDate – get from partner organization arrival dates of home organization's students at partner organization,
- sendDepartureDate – send to partner organization departure dates of partner organization's students from home organization,
- getDepartureDate – get from partner organization departure dates of home organization's students from partner organization.

SendArrivalDate and getArrivalDate methods operate on `studentArrivalDate` elements shown in Figure 6.25; example is provided in Listing 6.14. Similarly sendDepartureDate and getDepartureDate exchange twin `studentDepartureDate` elements – see Figure 6.26.

Figure 6.25: `studentArrivalDate` element

```
<tns:partnerStudentId>PL:85102602439</tns:partnerStudentId>
<tns:arrivalDate>2008-10-05</tns:arrivalDate>
```

Listing 6.14: `studentArrivalDate` example

Figure 6.26: `studentDepartureDate` element

sendLearningAgreement/getLearningAgreement

Basic semantics:

- sendLearningAgreement – send to partner organization a list of course instances (from the catalog of partner organization) chosen by a student from home organization,
- getLearningAgreement – get from partner organization a list of course instances (from the catalog of home organization) chosen by a student from partner organization.

These methods exchange a list of `courseInstanceT` values (Figure 6.15).

sendTranscriptOfRecords/getTranscriptOfRecords

Basic semantics:

- sendTranscriptOfRecords – send to partner organization a ToR of a student from partner organization,
- getTranscriptOfRecords – get from partner organization a ToR of a student from home organization.

These methods exchange a list of `gradeFromCourse` elements – see Figure 6.27; example in Listing 6.15.

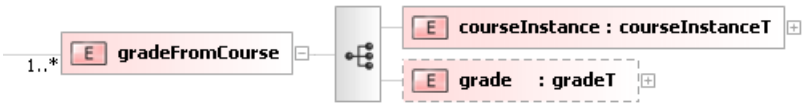

Figure 6.27: `gradeFromCourse` element

```
<tns:courseInstance>
    <tns:code>2200−1CWPM22</tns:code>
    <tns:studyCredits>
        <tns:value unit=”ects”>5</tns:value>
    </tns:studyCredits>
    <tns:contactHours>60</tns:contactHours>
    <tns:period>
        <tns:academicYear>2008</tns:academicYear>
        <tns:academicPeriod>S1</tns:academicPeriod>
        <tns:duration>1</tns:duration>
    </tns:period>
    <tns:lecturerId>uw.edu.pl:13321</tns:lecturerId>
</tns:courseInstance>
<tns:grade>
    <tns:value gradingScheme=”ects”>A</tns:value>
    <tns:value gradingScheme=”local”>5</tns:value>
</tns:grade>
```

Listing 6.15: `gradeFromCourse` example

Chapter 7

Implementation

7.1. Technologies

This chapter describes chosen technologies, important implementation decisions and other relevant implementation details.

The software platform chosen for the project is Java, mainly because it provides a modern, freely available and architecture-agnostic execution environment accompanied with plentiful of libraries which allow to quickly develop solutions using industry standards such as web services, UDDI, standards for MOM (Message-Oriented Middleware) to name a few particularly interesting from the project's perspective.

One possible choice is Java EE, a widely recognized and popular standard for complex, distributed, multi-tier applications, created and maintained by Java vendor – Sun Microsystems, Inc. Java EE specification provides lots of functionality through standardized APIs. However, the problem with Java EE is that it is not possible to use only these pieces of functionality that are really needed and the project does not require all the enterprise features incorporated by Java EE compliant application server. The process of starting and stopping of an application server is also significant and it has its impact on the general development time and comfort.

It has to be mentioned that before this implementation another prototype was created and it was a Glassfish-based solution. Experience gained from that attempt is highly convergent with what is written above. Moreover, the team responsible for that implementation have encountered a problem with connecting Oracle AQ to Glassfish JMS, because Glassfish cannot support Oracle queues out of the box. An Oracle-JMS proxy has been developed but sadly this solution did not work reliably. After further investigation it came out that the only promising solution based on Glassfish needed to include a commercial product.

It became apparent that an alternative solution is desired, better in terms of modularity, simplicity, easiness of development, maintainability and finally fully working and available at no cost. Fortunately, Apache Software Foundation projects provide the necessary communication functionality (UDDI, web services, messaging) without sacrificing (almost) any of these features.

7.1.1. Apache Tomcat

Apache Tomcat is a servlet container capable of running Java applications conforming to the Java Servlet and JavaServer Pages (JSP) specifications. An HTTP server is required in order to run a web service and a servlet container also fits in that place. It is lightweight, robust and very popular [INFOQ].

7.1.2. Spring Framework

Spring Framework is a general-purpose application framework providing IoC (Inversion of Control) container. It manages a lifecycle and configuration of application objects. Thanks to IoC objects are given their dependencies by the container which are composed declaratively, typically in an XML file. This allows to remove the burden of object creation and management through traditional object factories. The result is cleaner and more manageable code. As the primary source of reference, I used *Spring in Action* by C. Walls and R. Breidenbach [WaBr05].

7.1.3. Apache Camel

Apache Camel is a routing and mediation engine which provides an Enterprise Integration Patterns (EIP) implementation. It can work directly with numerous types of transport or messaging models, including but not limited to HTTP, JMS, AMQP, CXF (the project actually utilizes Camel's CXF support for web services, Oracle AQ access through JMS API and AMQP support). Interaction is done through a uniform API for all types of transport, yet it does not prevent access to specific characteristics of the underlying transport layer.

There are a few other similar and royalty-free solutions, mainly based on Enterprise Service Bus (ESB) architecture, for example: OpenESB, Apache ServiceMix, OW2 PEtALS. However, the scope of these projects is a bit different because ESB focuses on providing an enterprise messaging engine, usually through JBI specification. Therefore the messages are delivered in a normalized format through a mechanism built on web services model.

Camel looks simpler than ESB which has a more enterprise background but even if it would finally be felt that for some reason ESB is a better choice – there is still a possibility of integrating Camel with ServiceMix.

7.1.4. Apache CXF

Apache CXF is a web services framework allowing to build SOAP and WS-* (WSI Basic Profile, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging, WS-Security, WS-SecurityPolicy and WS-SecureConversation) standards-based services through JAX-WS API (Java API for XML Web Services). It also integrates with Spring Framework and is easily embeddable in custom solutions. All these features make CXF suitable for this project.

7.1.5. Apache jUDDI

Apache jUDDI is a UDDI Registry implementation. It conforms to UDDI version 2 specification which is sufficient enough to serve as a simple, private registry.

7.1.6. MySQL

MySQL acts as a DBMS backend for jUDDI. Although other DB engines are supported, MySQL was chosen because it is very popular and proven. It is also widely used at Polish universities. Nonetheless, there is an option to use any of the jUDDI-compatible DBMSs: DB2, HSQLdb (HypersonicSQL), Sybase, PostgreSQL, Oracle, TotalXML, JDataStore (Borland).

7.1.7. Apache Scout

Apache Scout is an implementation of the JAXR API [JAXR] which allows to interoperate with a UDDI version 2 compliant Registry instance. It is a simple implementation intended

to do only this one task and as it is an Apache project, one may expect easy integration with other ones. Another free UDDI library designed for Java – UDDI4J – uses a non-industry-standard API and its use is therefore discouraged.

Unfortunately, during development a few bugs in Scout were discovered (and reported to the developers of the project): two minor [SCOUT98], [SCOUT101] but also one rather critical [SCOUT99] as it effectively prevented retrieval of internationalized names from registry (at most one name could only be fetched). Due to the lack of feedback from Scout's maintainers, I fixed the issues by myself. Thanks to the license used for the project – Apache License 2.0 [APACHELIC] – the activity of source code modification as well as further redistribution of the modified code were legally permitted.

7.1.8. Apache Qpid

Apache Qpid is an AMQP (Advanced Message Queuing Protocol) implementation. I chose AMQP in favour of JMS mainly because the Apache JMS implementation – ActiveMQ has a disqualifying, critical bug: the broker must be available when the client starts, otherwise the clients hang until the broker is available [AMQ2114]. Main conceptual difference between AMQP and JMS is that AMQP defines a transport layer protocol. As a result, any two AMQP-compliant implementations should be truly interoperable which is not the usual case in the JMS world.

7.1.9. ICEfaces

ICEfaces [ICEFACES] is a server-based RIA (Rich Internet Application) framework. It is an extension to the standard JSF specification. It supports AJAX and thus it can handle rendering phase a bit differently. Namely, it is possible to avoid full page refreshes because only changed parts of the page are sent to the browser and get rendered. There is also support for an AJAX push feature which allows a server-initiated page update. This functionality allows to easily develop a page which would show current invocations of the web service methods. Other reasons to be in favour of ICEfaces are attractive default look and a rich component library.

I have to notice that there exists an Apache JSF implementation as well but it does not contain extensions provided by ICEfaces.

7.1.10. JavaScript addons

I used two JavaScript utility addons:

- BoxOver (`http://boxover.swazz.org/`) – configurable tooltips, a feature not available in JSF itself, nor in ICEfaces,
- CodeMirror (`http://marijn.haverbeke.nl/codemirror/`) – code editor with syntax highlighting.

7.2. Tools

7.2.1. NetBeans

I used NetBeans IDE as the main Java, JSP, XML editor. It significantly eases editing source files through many useful features such as syntax highlighting, syntax completion, a-jump-to-declaration feature, Javadoc support and many more.

7.2.2. Eclipse

Eclipse served for me as a WSDL editor – I found its code completion mechanism exceptionally convenient.

7.2.3. Apache Maven

Apache Maven is a build automation tool. It has an extensible, plugin-based architecture which removes the burden of creating an imperative specification what to do to accomplish a specific task. The behaviour is configured through (usually minimal) XML configuration. Maven assumes reasonable defaults which in many cases are quite sufficient.

Maven's main strength is the network-ready dependency handling mechanism. Maven maintains a local repository of libraries which are downloaded from remote repositories if the project being built is dependent upon them and they are not already present in the local cache. Transitive dependencies are also taken into account, so it greatly relieves the developer of manually completing their environment.

7.2.4. Subversion (SVN)

I used SVN version control system for storing the source code of the project.

7.2.5. Liquid XML Studio Community Edition

I used Liquid XML Studio to generate a HTML documentation for the WSDL document enriched with graphical representation of its parts.

7.2.6. UML

I utilized UML (Universal Modeling Language) 2.0 formalism to describe business processes in Section 3.2. UML includes a set of graphical notation techniques to describe static and dynamic aspects of a software system: design, structure, behaviour, use cases and other. UML is a de facto industry standard.

7.2.7. Drawing tools

I prepared UML diagrams using Dia. Other drawings were created in OpenOffice.org Draw and Microsoft Visio.

7.3. Components

The project has the following hierarchy:

```
mobility-project
|-- mobility-basics
|-- mobility-client
|   |-- mobility-client-transport
|   |-- mobility-client-web
|-- mobility-server
|   |-- mobility-server-transport
|   |-- mobility-server-web
|-- mobility-uddi
\-- mobility-wsdl
```

The software consists of 4 Java web applications (distributed as war archives):

- mobility-server-transport,
- mobility-server-web,
- mobility-client-transport,
- mobility-client-web.

Mobility-client-transport and mobility-server-transport applications act as a transport layer between HEI's RDBMS and the web service. Both of them use Camel's Java DSL to route the messages from database and vice versa. The transport layer interfaces to the SIMS with XML payload of the web service messages.

Because of the requirement of interoperability with USOS which has its logic coded in Oracle procedures (see p. 32), the software communicates with it through PL/SQL procedures and Oracle Advanced Queuing messaging system abstracted with JMS interface. Other solutions may expose a different interface but there is a great chance that it would be relatively easy to integrate with Camel due to its extensive communication technologies support.

Mobility-client-web (vel Mobility Client web interface) enables you to invoke operations of the web service manually and edit UDDI Registry. Mobility-server-web (vel Mobility Server web interface) enables you to preview request coming to server. Mobility-client-web and mobility-server-web applications are mainly testing tools.

Chapter 8

Summary

8.1. Goals achieved

Beginning with functional requirement number 1 (functional requirements are presented in Section 3.3.1), one can state that the infrastructure developed enables effective data exchange between educational organizations participating in student mobility programmes. The vocabulary for the modeled domain as well as behaviours of data exchange have been established and the network transport software has been created. The solution is capable of being integrated with a SIMS, which was verified by the example of USOS – some further information on that can be found in the beginning of the next section.

The requirement of testability (number 3) is also met. One can initiate method invocations through mobility-client-web interface. The interface is supplied with XML message templates which are intended to aid in performing this task. The results of subsequent invocations may be observed with mobility-server-web.

The status of requirement 2 is described lower, along with a connected non-functional requirement.

As regards the non-functional requirements, there are also quite a few accomplishments to be noted (see Section 3.3.2 for non-functional requirements).

Firstly, the architecture chosen assumes that any two nodes are equal, i.e. they do the same job, none of the nodes is privileged in any way, etc., so the non-functional requirement number 2 is achieved.

So is the case of the requirement 3. A single node in the system is not going to communicate with all the rest. Lack of central entity which gathers traffic from extensive number of nodes is another argument for claiming that the solution will scale well.

Requirement of high source code generality in terms of independence from transport data format definition (number 4) is fulfilled by features of implementation described in Section 7 to the extent given in subsection A.4.5.

As far as the legal issues are considered, there is a decision to make national personal identifier (`nationalPersonalId` element in `personalCharacteristicsT`) optional as it may not be desirable to expose it due to some legal concerns of personal data protection present in several European countries.

Referring to requirement number 7 – installation and management ease, it is hard to tell authoritatively whether this property holds or not. It has to be admitted that it is not a one-click install but rather requires performing several tasks which demand some basic knowledge about administration and Java conventions specifically. My personal opinion, however, is that installing it is not a hard task for a person who had any previous experience with Java web

applications. There is no need to perform any specific maintenance tasks for the applications.

Choice of technologies was carefully carried out with conditions 8a, 8b and 8c in mind and all of them are satisfied. Apache projects as well as Spring are licensed on Apache License 2.0, ICEfaces uses Mozilla Public License 1.1. These licenses as well as licenses used for other software components used are compatible with GNU GPL v3 [GPLCOMPAT].

Security issues (number 2 in Section 3.3.1, number 5 in Section 3.3.2) have not been thoroughly addressed in this work. Nonetheless, some guidance on achieving the goals specified by them is provided in Section 8.2.3.

It is quite hard to evaluate the generality of the data format designed (requirement number 1) but it is apparently true that it does not force to use concepts specific for Erasmus programme. The definitions of course-related types in Section 6.4.5: `subjectAreaCodeT` (see p. 54) and `gradeT` (see p. 56) are prepared to use different codings. Nonetheless, the real, practical applicability to different programmes needs to be discussed by the interested parties and verified empirically.

8.2. Future work

8.2.1. Integration with USOS

USOS development team has begun to integrate the provided software with USOS. At the time of writing, a great majority of methods defined in the WSDL document were already supported.

Apart from the logic which translates the data between USOS and the Mobility Project software, a prototype GUI is being created. This GUI part is essential in order to fully realize the requirement number 1. Although the mobility-client-web enables users to invoke web service methods, it is more of a generic testing tool than a client to be used in a production environment. In normal every-day use it would be far from convenient to extract data from USOS and construct XML messages manually. The GUI frontend will be implemented in Oracle Forms technology as the rest of the USOS presentation facilities already are.

8.2.2. WSDL

Further work needs to be done in order to discover all the complexities of mobility activity from organizational, procedural and educational point of view. This goal is going to be accomplished through extensive cooperation with other European HEIs vividly interested in the project.

It is expected that engaged HEIs will introduce installations of the provided software and try to integrate it within their IT environment. Despite the efforts some customizations and/or modifications may prove inevitable. Then hopefully exhaustive tests will be performed. The anticipation is, that throughout the steps described, a lot of experience is going to be collected and the outcome will be a new, even better-suited version of the WSDL definition.

It needs to be noticed that the above is perceived as an iterative process due to the fact that the requirements may evolve to some extent as the time goes by.

One possible direction of WSDL evolution is changing it to a more explicit CRUD-like style of operation. Instead of the current style which can be summarized as “add at first, then only updates possible”, one could imagine a bit different design with more understandable names and semantics. Examples of verbs following that style would be: create, update, delete.

Other suggestion would be to introduce a more negotiate style, i.e. currently the operations either succeed or fail. Until any of the parties invokes the operation again, it is assumed

that both of them agree on the shared state. This could be enhanced with providing a way of proposing the new state, which would not become effective until a confirmation is sent. Operations could be named as: suggest, propose, accept, reject.

8.2.3. Implementation

UDDI

As the security considerations will be taken into account, either the TLS-way or WS-Security-derivative way, the requirement for digital certificates/keys exchange will appear as well. Apart from that, the web service registry solution will need to be appropriately adjusted.

This may be done by transition to UDDI version 3 which is, amongst others, enhanced with:

- replication API (for load balancing and redundancy),
- PK signing.

There is also a chance for an easy upgrade from UDDI version 2 because Apache jUDDI project offers a version 3 compliant implementation.

Another possible solution addressing these issues could be found in OASIS ebXML Registry-Repository standard [EBXML]. Important features of ebXML Registry-Repository standard include:

- federation support – it allows for multiple registries to constitute a logically single registry while still having local autonomy and security policies active,
- security features: authentication, authorization, single sign-on (SAML support), fine-grained role-based access control,
- notification services,
- content validation and cataloging.

In addition, ebXML is JAXR-compliant, so implementation of the enhancements suggested here and in the following subsection is possible in a rather evolutionary than revolutionary fashion. The implementation of this standard which could be of particular interest is an open source project called freebXML Registry [FEBXML].

Web service

WSDL changes are expected to occur because the work on refining the standard is going to continue. When one imagines a situation of a new version of WSDL document being released but the target environment is composed of multiple independent nodes, it is apparent that some of them will adopt the new specification sooner and some later. This situation shows that we have to deal with several issues. Firstly, we need to maintain some sort of WSDL versioning and a mechanism which would fall back to the most recent version supported by two communicating nodes.

Other important matter to be considered is security. One way of achieving a secure transmission with authentication is to make web services use HTTPS rather than HTTP. It is relatively easy to turn on TLS support in Tomcat [TOMCATSSL], [PENTAHOWIKI] and have both sides authenticated with Server and Client Authentication enabled. However, due

to the potentially growing number of nodes the approach which assumes the keystore file being static may become impractical.

Other possibility is to introduce one of the security-related web services standards: WS-Security [WSS], WS-SecurityPolicy [WSSP] or WS-Trust [WST].

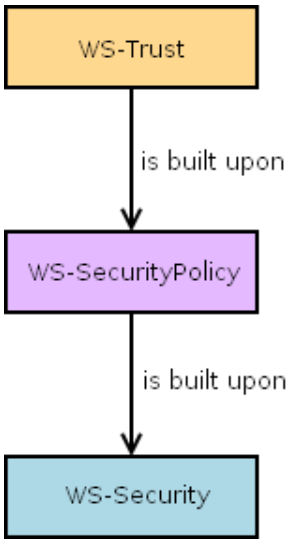

Figure 8.1: WS-Security and its derivatives

The relationship between these standards is shown in Figure 8.1. WS-Security is a flexible mechanism, yet also very low-level. It allows to attach signatures, encryption headers and security tokens to SOAP messages and encrypt the messages themselves. WS-SecurityPolicy is a refinement of WS-Security and WS-Policy and provides an easier and more standards-based way of configuring security requirements of a web service. But WS-Trust specification seems the most attractive:

"When using ≪straight≫ WS-Security, the client and server need to have keys exchanged in advance. If the client and server are both in the same security domain, that isn't usually a problem, but for larger, complex applications spanning multiple domains, that can be a burden. Also, if multiple services require the same security credentials, updating all the services when those credentials change can by a major operation.

WS-Trust solves this by using security tokens that are obtained from a trusted Security Token Service. A client authenticates itself with the STS based on policies and requirements defined by the STS. The STS then provides a security token (example: a SAML token) that the client then uses to talk to the target service. The service can validate that token to make sure it really came from the trusted STS." [CXFWST]

Performance is not a crucial issue due to relatively small amounts of data being transmitted and at relatively low frequency. However, if it would turn out that there is a need to improve on this, a possible solution would be to apply a compression algorithm to the XML content sent through web service. After all, XML documents by its nature contain a lot of repetitive markup. There is a report [Ste07] which claims that potential gains resulting from applying gzip algorithm range from 2 to 7 times. Another commercial solution [LSD] is advertised as capable of producing compact representations of XML files which are 20 times smaller.

There are also more sophisticated solutions to the problem which not only help to reduce the size of documents but also offer to decrease the time needed to parse them. There is a whole family of "binary XML" specifications aimed at providing a compact representation of XML. Two notable examples are:

- EXI (Efficient XML Interchange) – format developed by AgileDelta, Inc.; submitted to W3C, status: second draft of evaluation,
- Fast Infoset, being an ITU-T [ITUT] and ISO standard – ITU-T Rec. X.891 [ITUFI] and ISO/IEC 24824-1 (Fast Infoset).

Other

One useful enhancement in the matter of configuration management of the project would definitely be a set of unit tests. Anyone working on the project would benefit a lot if they would be able to have a means of detecting some regressions automatically.

8.3. Acknowledgements

I would like to thank Dr Janina Mincer-Daszkiewicz for her time, attention and invaluable advice regarding the work presented hereby.

Appendix A

Documentation

A.1. Overview

This chapter provides documentation of the software developed within the frames of the Mobility Project. It is divided into 5 sections.

Present section contains some general information on the organization of the documentation. Section A.2 describes actions which need to be taken in order to install and run the software, whereas the section following immediately (Section A.3) provides some insight into typical configuration useful for testing and experimenting with the software. The purpose of Section A.4 is to provide information useful for modification and maintenance. Section A.5 is the last one and it contains the user's guide.

A.1.1. General notes

Sections A.2 and A.4 refer to various files used by the applications.

Any configuration file change can be made in two ways:

- directly to the source project (and then recompiled),
- by altering an already built war or jar archive.

Symbol	Mode		
	src	war	jar
`RES_ROOT`	`src/main/resources`	`WEB-INF/classes`	ε
`WEB_ROOT`	`src/main/webapp`	ε	–

Table A.1: Auxiliary variable definitions

Table A.1 contains some auxiliary variable definitions used when referencing configuration files in the mentioned sections.

A.2. Getting started

A.2.1. Installation

Requirements

In order to successfully install and use the software provided the following requirements need to be fulfilled:

- Java SE Development Kit (JDK) 1.6 (the latest build recommended),
- Apache Tomcat 6 installation,
- Apache jUDDI 2 installation with database registry fully configured and working – installation procedure is included further in the present subsection (precisely in paragraph jUDDI 2),
- Apache Qpid installation, if mobility-server-web is going to be installed,
- a web browser – needed to access web interfaces, examples: Firefox, (Windows) Internet Explorer, Opera (the latest versions recommended for proper CSS interpretation).

Steps

jUDDI 2 Because the official installation guide of jUDDI is not good enough to seamlessly prepare a working instance (it is incomplete and difficult to understand), I decided to provide a more comprehensible guide. The following applies to jUDDI 2 release candidate versions (2.0rcx) and final releases (2.0.0, 2.0.1) as well. The only difference is that final releases lack a few jars.

1. Using MySQL administrative (root) account perform the following steps:
   (a) create database for UDDI registry:

   ```
   CREATE DATABASE <dbName>;
   ```

   where <dbName> denotes the name of the database,
   (b) create user with an arbitrary name, set him a password and grant him privileges to the database:

   ```
   CREATE user '<dbUser>'@'<host>' IDENTIFIED BY '<dbUser>';
   SET PASSWORD FOR <dbUser>@<host> = PASSWORD('<dbPass>');
   GRANT ALL privileges ON <dbName>.* TO <dbUser>@<host>;
   ```

   where:
   - <dbUser> – name of the newly created database user,
   - <dbPass> – password of this user,
   - <host> – the host which will initiate connections to jUDDI, i.e. the host where the Tomcat installation containing the Mobility Project software resides (typically localhost).
2. Copy the MySQL Connector/J 5.1 (JDBC driver for MySQL) jar (mysql-connector-java-5.1.xx-bin.jar) to `$CATALINA_HOME/lib`.

3. Unpack the jUDDI war archive to `$CATALINA_HOME/webapps/<juddiDir>`, where <juddiDir> stands for the directory name chosen; probably "juddi" would be a good choice.

4. If a final release of jUDDI is used, you need to copy several additional jars which can be found on the DVD attached (they reside in `juddi-2.0-final-jars` directory). The names of these jars are:

   - axis-1.4.jar,
   - axis-jaxrpc-1.2.1.jar,
   - axis-saaj-1.2.1.jar,
   - commons-discovery-0.2.jar,
   - standard-1.1.2.jar.

   They need to be copied to `$CATALINA_HOME/webapps/<juddiDir>/WEB-INF/lib`.

5. Add the contents of Listing A.1 to `$CATALINA_HOME/conf/server.xml` as a child of `Host` tag, taking care to substitute placeholders in angle brackets with actual values used in previous steps; <mysqlHost> and <mysqlPass> denote the hostname and port number of the machine containing a MySQL installation.

```
<Context path=''/<juddiDir>'' docBase=''<juddiDir>'' reloadable=''true''
    crossContext=''true''>
    <Resource name=''jdbc/juddiDB'' auth=''Container''
        type=''javax.sql.DataSource'' maxActive=''100'' maxIdle=''30''
        maxWait=''10000'' username=''<dbUser>'' password=''<dbPass>''
        driverClassName=''com.mysql.jdbc.Driver''
        url=''jdbc:mysql://<mysqlHost>:<mysqlPort>/<dbName>?autoReconnect=true''
        validationQuery=''select count(*) from publisher''/>
</Context>
```

Listing A.1: jUDDI entry in Tomcat's server.xml

6. Check that everything was configured properly by pointing your browser to the address of your installation and choose the *Validate* option. If there are no red messages, the installation is successful. During validation the database structure (a set of 31 tables) needed by jUDDI should have been created. At this point you can successfully create a jUDDI user account with the following SQL:

```
INSERT INTO publisher (publisher_id, publisher_name, is_enabled, is_admin)
VALUES ('<juddiUser>', '<juddiPass>', 'true', 'false');
```

   substituting <juddiUser> and <juddiPass> with custom values. These credentials need to be supplied to the UDDI configuration file of the Mobility software.

7. Release candidate releases of jUDDI 2 do not have logging configured. This can be done by creating a file `WEB-INF/classes/log4j.properties` in the root directory of your jUDDI installation (`$CATALINA_HOME/webapps/<juddiDir>`). Example configuration is given in Listing A.2.

```
log4j.rootLogger=INFO, out, log

log4j.appender.out=org.apache.log4j.ConsoleAppender
log4j.appender.out.layout=org.apache.log4j.PatternLayout
log4j.appender.out.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.log=org.apache.log4j.RollingFileAppender
log4j.appender.log.File=${catalina.home}/logs/juddi.log
log4j.appender.log.MaxFileSize=1MB
log4j.appender.log.MaxBackupIndex=3

log4j.appender.log.layout=org.apache.log4j.PatternLayout
log4j.appender.log.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.debug=false
```

Listing A.2: log4j configuration for jUDDI

Tomcat Java 6 has an issue with web service stack implementation (JAX-WS 2.1) which is used by Apache Scout [JAXWS-JAVA6]. As a workaround, you need to copy a few JAX-WS 2.1 jars (the attached DVD contains jars from JAX-WS 2.1.7 release) to the `endorsed` directory of your Tomcat installation, i.e. `$CATALINA_HOME/endorsed`. The filenames are:

- jaxb-api.jar,
- jaxb-impl.jar,
- jaxws-api.jar,
- jaxws-rt.jar,
- stax-ex.jar,
- streambuffer.jar.

1. Set desired HTTP port used by Tomcat – check `Connector` section attributed with `protocol="HTTP/1.1"` in `$CATALINA_HOME/conf/server.xml`.
2. Configure user accounts (for mobility-client-web, mobility-server-web) by editing `$CATALINA_HOME/conf/tomcat-users.xml`:
   (a) add the following elements as children of `<tomcat-users>`:
   ```
   <role rolename="mobility-client-web"/>
   <role rolename="mobility-server-web"/>
   ```
   (b) create user accounts which need access to the applications by adding an entry with custom username and password attribute values, example given below uses "admin" for both username and password:
   ```
   <user username="admin" password="admin"
    roles="mobility-client-web, mobility-server-web"/>
   ```

The Mobility Project applications

1. Deploy push-server.war contained in ICEfaces distribution (in order make AJAX push technology work properly):
   (a) download ICEfaces v1.8.1 distribution (file is named ICEfaces-1.8.1-bin.zip) from producer's website [ICEFACES],
   (b) unpack war file which is pointed by the path:
   `ICEfaces-1.8.1-bin/icefaces/push-server/push-server.war`
   and copy it to directory `$CATALINA_HOME/webapps/push-server`.
2. Unpack each of the application war archives:
   - mobility-client-transport.war,
   - mobility-client-web.war,
   - mobility-server-transport.war,
   - mobility-server-web.war

   to the directory `$CATALINA_HOME/webapps/<appName>`, where <appName> is the name of the jar unpacked without .war extension.
3. Configuration per application (file locations given here are relative to the root directory of the application directory tree):
   - mobility-client-transport
     (a) configure database connection by editing `RES_ROOT/database.properties`,
     (b) configure UDDI – edit `RES_ROOT/uddi.properties` located in the root of `WEB-INF/lib/mobility-uddi-1.0.jar`;
       i. make sure that the keys `uddi.username` and `uddi.password` have the same values as the username and password inserted into the `publisher` table of the database dedicated to jUDDI; the values needed here are the ones denoted with <juddiUser> and <juddiPass>,
       ii. make sure that the following keys have proper values:
       – `uddi.lifeCycleManagerURL` – `<url>/publish`,
       – `uddi.queryManagerURL` – `<url>/inquiry`,
       where <url> is the URL address of the application, i.e. http://localhost:8080/¡juddiDir¿.
   - mobility-client-web
     (a) configure UDDI – edit `RES_ROOT/uddi.properties` located in the root of `WEB-INF/lib/mobility-uddi-1.0.jar` in the same way as above,
   - mobility-server-transport
     (a) configure database connection by editing `RES_ROOT/database.properties`,
     (b) `RES_ROOT/server.properties` – enter home organization id (used to lookup appropriate XML file when the application uses XML files as the data source – for testing purposes).
   - mobility-server-web
     (a) `RES_ROOT/server.properties` – enter home organization id (for informational purposes).

4. Configure CATALINA_OPTS environment variable by setting it to `-Xms256m -Xmx512m -XX:PermSize=256m -XX:MaxPermSize=512m`; the values given worked for a 32-bit Windows platform – generally the values may differ (e.g. higher for 64-bit architectures). This is similar to the case previously noted in subsection A.4.1.

A.2.2. Running

In order to run the applications:

1. run Qpid broker (needed by mobility-server-web):
   - `%QPID_HOME%\bin\qpid-server.bat` (Windows)
   - `$QPID_HOME/bin/qpid-server` (Unix)
2. start Tomcat by executing the following commands:
   - `%CATALINA_HOME%\bin\startup.bat` (Windows)
   - `$CATALINA_HOME/bin/startup.sh` (Unix)

A.3. Testbed

A common approach at integrating the software with local SIMS is to test it locally in both client and server scenarios. Therefore two instances of nodes need to be established. One possibility is to make two installations on different ports within one Tomcat instance by configuring `$CATALINA_HOME/conf/server.xml` with two `Service` sections with `Connector` sections using different ports.

But in order to simulate a real-world environment better it would be desired to make two installations of Tomcat with different ports specified in `Connector` section (as described in subsection A.2.1). It may be also helpful to supply the UDDI registry with an entry mapping the local identifier with the local web service to test a case of a partner invoking methods (using mobility-client-web).

Apart from the client part which may be simulated with mobility-client-web using its sample data files containing requests (see subsection A.5.1), it is possible to make the server part – mobility-server-transport – respond with the contents of a file instead of the result of a database procedure call. Instructions on how to configure this behaviour are provided in the next two sections (A.3.1 and A.3.2).

An exemplary testbed is shown in Figure A.1.

A.3.1. Modifying requests routing – mobility-server-transport

File `RES_ROOT/mobility-server-transport-context.xml` contains Spring bean definitions. Routing decision, i.e. whether the response is created in the database or read from the XML file, depends on the `proxyDAO` properties configuration:

- `defaultMapping` – value is a `ResponseDAO` instance; denotes the default bean handling a request,
- `forwardTo` – value is a `Map<String, ResponseDAO>` instance which provides overrides for the setting imposed by `defaultMapping`; key should be an operation name, value is a target bean – the beans which apply here (implement `ResponseDAO` interface) have id attributes of `xmlFileDAO` and `databaseDAO`.

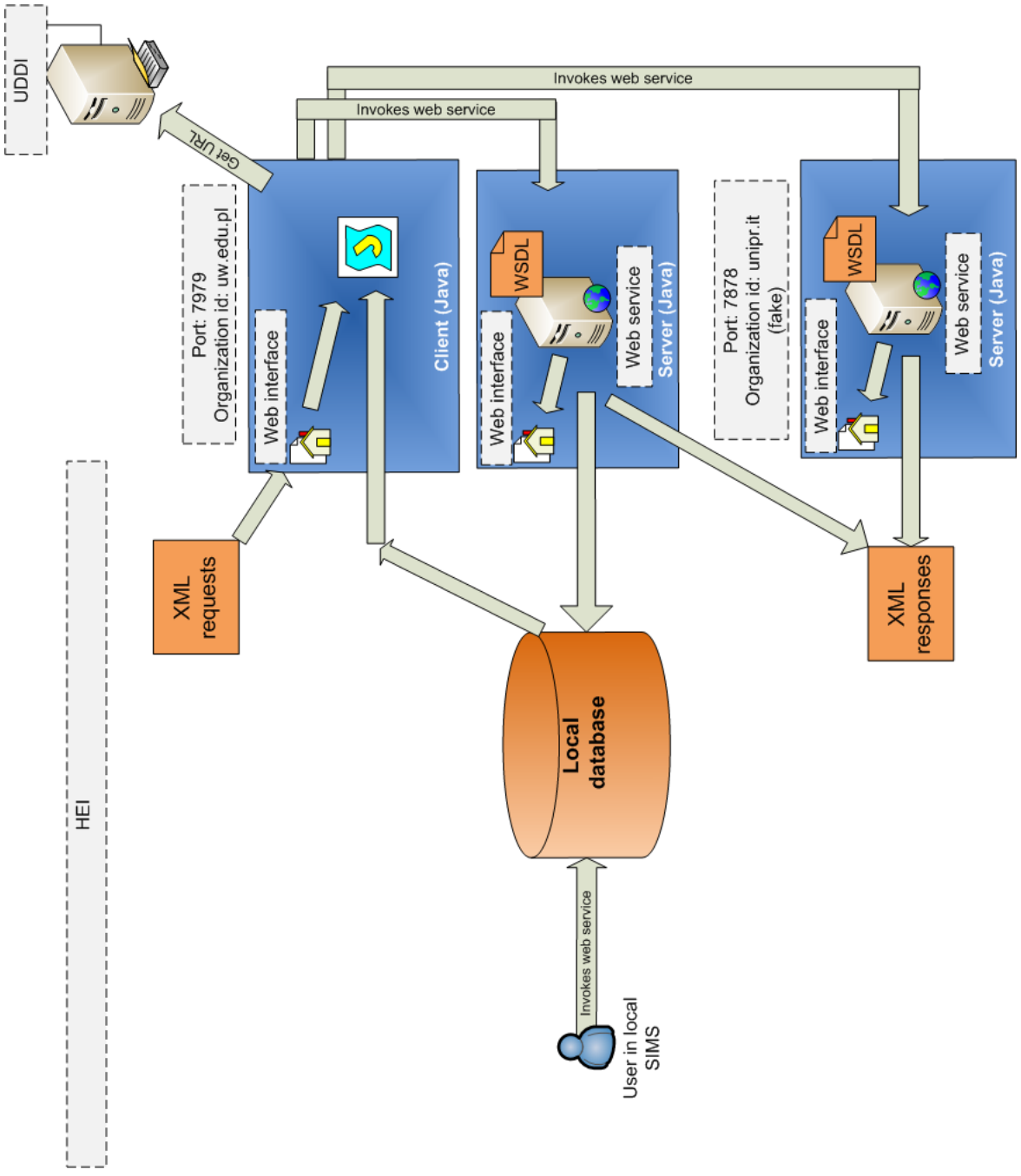

Figure A.1: Exemplary testbed

A.3.2. Adding response files to mobility-server-transport

In order to leverage existing behaviour when providing a response for a request which is configured to use a file through `xmlFileDAO` as described in Section A.3.1, it is advised to put the response files of:

- agreement-independent methods – to `RES_ROOT/xml/server=<serverOrganizationId>/client=@ny`,
- agreement-dependent methods – to `RES_ROOT/xml/server=<serverOrganizationId>/client=<clientOrganizationId>`.

The filename format is <responseName>.xml (for example the file containing a response for getOrganizationData method would be named getOrganizationDataResponse.xml) with an exception for files containing responses for getCourseDataResponse method. In this case the filenames are formatted as getCourseDataResponse_<code>.xml, where <code> is substituted with a code of a course – a `courseCodeT` value. You can provide different behaviour by implementing `FileNameMapper` interface and registering an override through `fileNameMappers` property analogously to `forwardTo` property described in Section A.3.1.

A.4. Other tasks

A.4.1. Compilation

Requirements

- Java SE Development Kit (JDK) 1.6 (the latest build recommended).
- SVN installation (the newer the better).
- Apache Maven 2 installation.
- Internet access if there exists a project dependency which is not present in the local Maven repository[1].
- Oracle jars:
  - Advanced Queuing – aqapi13.jar,
  - JDBC for JDK 1.5 (version compatible with Oracle DB installation, project requires at least 10.2.0.4.0) – ojdbc5.jar, orai18n.jar, orai18n-mapping.jar.

Steps

1. Make sure the Oracle jars are installed in local repository in the following manner:

```
mvn install:install-file -Dfile=<path-to-file> -DgroupId=<group-id> \
    -DartifactId=<artifact-id> -Dversion=<version> \
    -Dpackaging=jar -DgeneratePom=true
```

   where:

[1] Project dependency list can be obtained by issuing `mvn dependency:list` in the project root (assuming that the project libraries are installed in local repository – running `mvn install` does that).

- <group-id>=com.oracle,
- <artifact-id> is the jar name without .jar extension,
- <version> is an Oracle version release (e.g. 10.2.0.4.0, 11.1.0.6.0), for aqapi just enter 1.0.

2. Perform a patched compilation of Apache Scout 1.2:
   (a) retrieve the source tree of Scout:
   ```
   svn checkout \
   http://svn.apache.org/repos/asf/webservices/scout/tags/scout-1.2 scout
   ```
   (b) apply the patch named `scout-1.2.patch` (available on the attached DVD) – make sure to change current working directory to the project root directory, i.e. `scout`:
   ```
   patch -p1 < /path/to/scout-1.2.patch
   ```
   (c) compile, package and install into local Maven repository using `mvn install`,
3. using command-line enter the root project directory `mobility-project` and issue `mvn package`.

Note: If the build fails with a `java.lang.OutOfMemoryError: Java heap space` or similar, try increasing memory (heap space) available for build. It can be done by setting MAVEN_OPTS environment variable to `-Xms64m -Xmx128m`. Depending on various factors (primarily architecture – 32 or 64-bit) the values may be higher. A few experiments should suffice to succeed.

A.4.2. Internationalization

In the following text the term "to provide translation for `path/to/file.properties`" means: "to create a new file `path/to/file_<languageCode>.properties`, where <languageCode> is an ISO 639-1 code of the new language, with the same keys as the file identified by the path given but with values translated; preferably just copy the existing file and save it under a new name and translate the contents".

Mobility-client-transport, mobility-client-web, mobility-server-transport applications depend on mobility-wsdl-1.0.jar. You need to update it in each application.

- mobility-client-transport – provide translations for:
  - `RES_ROOT/pl/edu/usos/mobility/client/transport/bundle.properties`,
  - `RES_ROOT/pl/edu/usos/mobility/client/transport/fault/bundle.properties`.
- mobility-client-web
  1. provide translations for:
     - `RES_ROOT/pl/edu/usos/mobility/client/web/bundle.properties`,
     - `RES_ROOT/pl/edu/usos/mobility/client/web/validator/bundle.properties`,
  2. change used locale in `WEB_ROOT/WEB-INF/faces-confg.xml` – change text value of the node pointed by a pseudo-XPath expression: `/faces-config/application/locale-config/default-locale`.

- mobility-server-web
  1. provide translation for `RES_ROOT/pl/edu/usos/mobility/server/web/bundle.properties`
  2. change used locale in `WEB_ROOT/WEB-INF/faces-confg.xml` – change text value of the node pointed by a pseudo-XPath expression: `/faces-config/application/locale-config/default-locale`.
- mobility-wsdl – provide translation for `RES_ROOT/pl/edu/usos/mobility/fault/bundle.properties`.

A.4.3. Adding sample data files to mobility-client-web

Although web client interface is supplied with a few sample data files containing exemplary contents of messages handled by the system, it may be desired to enrich the application with more files of that kind.

Sample data file's filename format is <optionalPrefix><requestName><optionalSuffix>. The procedure is as follows:

1. add new sample data files to `RES_ROOT/xml/client=<clientOrganizationId>/server=<serverOrganizationId>`, where <clientOrganizationId> and <serverOrganizationId> are `organizationIdT` values,
2. add the following definition to `RES_ROOT/pl/edu/usos/mobility/client/web/resources.properties`:
   (a) append an arbitrarily chosen identifier (further referred to as <id>) to `xml-ResourceDefs` key (unique with respect to existing ids) – it needs to be delimited from the others with a colon,
   (b) add entries:
   - <id>.client=<clientOrganizationId>,
   - <id>.server=<serverOrganizationId>,

   (c) for each sample data file being added create an entry with a key formatted as <id>.data.<requestName> with a value indicating the filenames sample data files for a specific request; format is a colon-delimited list of values <optionalPrefix>{0}<optionalSuffix> (actually a sample data file's filename format with <requestName> substituted with {0}).

Example: we need to add sample data files for organizations identified with hei1.eu (client) and hei2.eu (server). Suppose that we have 2 sample data files named: getOrganizationData.xml, sendOrganizationData.xml. The above procedure in this case is as follows:

1. add files getOrganizationData.xml, sendOrganizationData.xml to `RES_ROOT/xml/client=hei1.eu/server=hei2.eu`,
2. an identifier has to be chosen – let it be hei1_hei2,
3. add the following to `RES_ROOT/pl/edu/usos/mobility/client/web/resources.properties`:

(a) append hei1_hei2 to current value of `xmlResourceDefs` key (colon-delimited),

(b) add entries:

- hei1_hei2.client=hei1.eu,
- hei1_hei2.server=hei2.eu,

(c) add entries:

- hei1_hei2.data.getOrganizationData={0}.xml,
- hei1_hei2.data.sendOrganizationData={0}.xml.

A.4.4. Adjusting transport to non-Oracle RDBMS

While the server part (mobility-server-transport) does not require any specific changes as long as HEI's student management system expects a PL/SQL procedure call, the client (mobility-client-transport), which is designed to work with Oracle AQ, may need to use a different connection factory in `jmsBean` definition located in `RES_ROOT/mobility-client-transport-context.xml`.

A.4.5. Modifying WSDL

Considerations

Modifying the WSDL document does not break the code, if the changes do not violate conventions assumed (described in Section 6.3). Apart from that, it is essential not to remove `errorT` type elements `code` and `message` as well as the type itself because the code uses `ErrorT` class created during `wsdl2java` code generation process.

Steps

The path to the WSDL document is `src/main/resources/wsdl/MobilityService.wsdl`, relative to the root of the mobility-wsdl project. Recompilation of the whole project is required, excluding mobility-server-web.

A.5. User's guide

A.5.1. Mobility Client web interface

Getting started

Open a web browser and enter URL address of the application. A window requesting a username and password will appear. Enter credentials and accept, shall the introduction page load.

Web service methods

In order to invoke a web service method, choose a method and click its name from the *Web Service Methods* menu – a view of the application after clicking the item *getOrganizationData* is shown in Figure A.2. Choose client and server from drop-down lists on the top. In the editor area of the *Request* section a valid XML containing the message body should be placed. Achieving that may be aided by the *Sample Data File* drop-down menu. It contains a list of samples, if there are any samples provided for the chosen combination of client and server

Figure A.2: Web service methods test functionality – *getOrganizationData* chosen and sample file loaded into the *Request* section

organizations. For instructions regarding how to add samples, please refer to subsection A.4.3. It is also possible to use a sample stored locally through *Upload a file* facility.

You can always validate prepared request against WSDL-generated schema to see whether it is syntactically correct. To use that feature press *Validate against schema* button. The result of validation will show up underneath.

The invocation of the method occurs when *Invoke* button is pressed. If it is successful, the contents of response message appear in the *Response* section. Otherwise, an error message is displayed under the *Invoke* button.

UDDI Registry

View mode Application can be used to edit UDDI Registry. In order to make use of this functionality click *Edit* menu item under *UDDI Registry* category in the menu on the left. Table with UDDI entries shows up in the main area. There are two modes of operation: view and edit mode. In the view mode (presented in Figure A.3) you can browse the details of any entry by clicking entries in the table. You can switch to edit mode by:

- pressing *Add* button which basically adds a new entry but it shows an editor to immediately provide values for the newly created entry, or
- pressing *Edit* button (an entry in the table needs to be selected) which shows the same editor but populated with current values of the selected entry.

Edit mode When the editor is visible and you click into a field, a small counter of the number of characters left appears on the right side of the field. The field *Name* has a subfield *Language* which contains an input field enriched with a convenient list of language codes conforming to ISO 639-1 norm. The list is visible when that field has focus (see Figure A.5).

Button with *Save* caption allows to save the state achieved in the editor to the UDDI Registry. The data is saved after clicking on this button, unless the data provided is not valid – in that case a short explanatory message is placed on the right side of any field which contains erroneous data. The fields are validated according to the rules imposed by the corresponding data types definitions in the WSDL document. *Cancel* button leaves the editor without saving anything. Both *Save* and *Cancel* buttons return to the view mode.

A.5.2. Mobility Server web interface

Application enables viewing messages received by the server side (i.e. a mobility-server-transport instance). Provided interface allows you to:

- view the total count of messages shown,
- change the number of items visible on single page,
- toggle a message view from collapsed to expanded and vice versa.

Figure A.6 shows the only screen available in this application.

Figure A.3: UDDI Registry in view mode

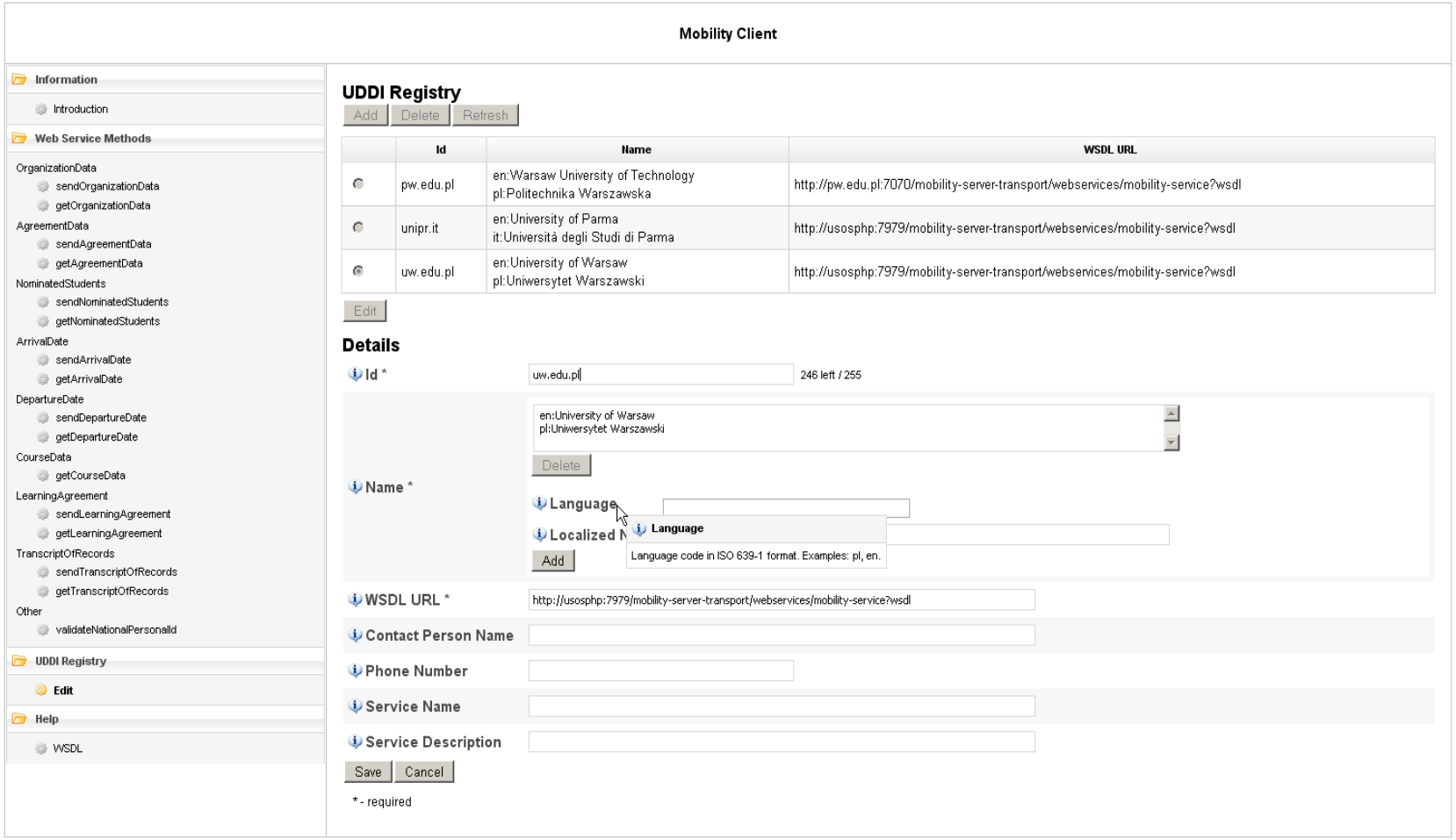

Figure A.4: UDDI Registry in edit mode

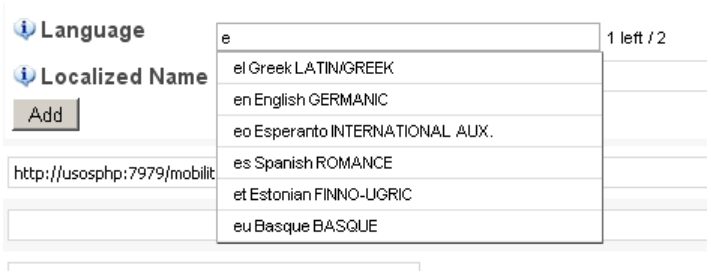

Figure A.5: Language hint list facility

Figure A.6: A view of Mobility Server web interface

Appendix B

DVD Contents

The attached DVD contains:

- this document in two forms: PDF format and LaTeX source,
- WSDL document with:
  - documentation generated with Liquid XML Studio from the XML Schema part,
  - example messages in XML files,
- patch file containing bug fixes for Apache Scout 1.2 source code,
- a set of jars from JAX-WS 2.1.7 release,
- software packaged in war archives and in source form (project tree),
- Javadoc source documentation.

Bibliography

[AMDR09] F. Arcella, J. Mincer-Daszkiewicz, S. Ravaioli, *Web-services for exchange of data on cooperation and mobility between higher education institutions*, EUNIS 2009, The 15th International Conference of European University Information Systems, 23-26 June 2009, Santiago de Compostela, Spain.

[AMQ2114] [#AMQ-2114] Failover transport should not hang on startup if it cannot connect, `http://issues.apache.org/activemq/browse/AMQ-2114`

[APACHELIC] Apache License, Version 2.0, `http://www.apache.org/licenses/LICENSE-2.0`

[AUPWG] CAUDIT Technical Standards Committee auEduPerson Working Group, `http://wiki.caudit.edu.au/confluence/display/aafaueduperson`

[CEDEFOP] Cedefop – European Centre for the Development of Vocational Training, `http://www.cedefop.europa.eu/`

[CEN] CEN – European Committee for Standardization, `http://www.cen.eu/`

[CENWSLT] CEN Workshop on 'Learning Technologies' (WS/LT), `http://www.cen.eu/cenorm/businessdomains/businessdomains/isss/activity/wslt.asp`

[CINECA] CINECA – Interuniversity Consortium, `http://www.cineca.it/`

[CWA15555] Guidelines and support for building application profiles in elearning, `ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/WS-LT/cwa15555-00-2006-Jun.pdf`

[CXFWST] Apache CXF – WS-Trust, `http://cwiki.apache.org/CXF20DOC/ws-trust.html`

[DCES] Dublin Core Metadata Element Set, Version 1.1, `http://dublincore.org/documents/dces/`

[DCMI] The Dublin Core Metadata Initiative, `http://dublincore.org/`

[DCMIG] [The Dublin Core Metadata Initiative] Glossary, `http://dublincore.org/documents/2001/04/12/usageguide/glossary.shtml`

[DCXML2003] Guidelines for implementing Dublin Core in XML DCMI Recommendation. 2003-04-02, `http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/`

[DCDSXML] Expressing Dublin Core Description Sets using XML (DC-DS-XML), `http://dublincore.org/documents/2008/09/01/dc-ds-xml/`

[EBXML] ebXML – Enabling A Global Electronic Market, `http://www.ebxml.org/`

[ECERA] European Commission – Education & Training – ERASMUS for Students – experiencing Europe from a new perspective,
http://ec.europa.eu/education/erasmus/doc1051_en.htm

[ECTS] European Commission - Education & Training - lifelong learning policy - European Credit Transfer and Accumulation System (ECTS),
http://ec.europa.eu/education/programmes/socrates/ects/index_en.html

[ECTSUG] ECTS Users' Guide,
http://ec.europa.eu/education/lifelong-learning-policy/doc/ects/guide_en.pdf

[EDUPER] Internet2 Middleware Initiative: eduPerson & eduOrg,
http://middleware.internet2.edu/eduperson/

[ELM] European Learner Mobility, http://wiki.teria.no/confluence/display/EuropeanLearnerMobility/European+Learner+Mobility

[EPXSD] Europass XML Schema v 2.0, http://europass.cedefop.europa.eu/TechnicalResources/XML/xsd/europass_XML-schema-v2.0-description.pdf

[EPMOBXSD] Europass Mobility XML Schema draft,
http://europass.cedefop.europa.eu/mobility/MobilitySchema_DRAFT.xsd

[FEBXML] freebXML Registry – A Royalty-free Open Source ebXML Registry Project – OASIS ebXML Registry RI, http://ebxmlrr.sourceforge.net/

[FVUSPEC] Finnish Virtual University: Standardization of educational data,
http://palvelut.virtuaaliyliopisto.fi/vy_standardization_eng.asp

[GPLCOMPAT] Various Licenses and Comments about Them – GNU Project – Free Software Foundation (FSF), http://www.gnu.org/licenses/license-list.html

[ICEFACES] ICEfaces – Open Source Ajax, J2EE Ajax, JSF Java Framework,
www.icefaces.org

[INFOQ] Tomcat used by 64% of Java Developers,
http://www.infoq.com/news/2007/12/tomcat-favorite-container

[IEP] irisEduPerson Working Draft v: 0.2.1 - 2008-04-30, http://wiki.rediris.es/gtschema/index.php/Iriseduperson

[ISCED] ISCED97, http://www.uis.unesco.org/publications/ISCED97

[ISO3166] ISO 3166 code lists,
http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm

[ISO8601] ISO 8601:2004 Data elements and interchange formats – Information interchange – Representation of dates and times

[ITUFI] ITU-T Rec. X.891, http://www.itu.int/rec/T-REC-X.891/

[ITUT] International Telecommunication Union - Telecommunication Standardization Sector, http://www.itu.int/ITU-T/

[JAXR] JSR 93: Java API for XML Registries 1.0 (JAXR),
http://jcp.org/en/jsr/detail?id=93

[JAXWS-JAVA6] Metro Guide – Using JAX-WS 2.1 with JavaSE6, http://jax-ws.dev.java.net/guide/Using_JAX_WS_2_1_with_JavaSE6.html

[Kra06] M. Krawczyński, *Uniwersytecki System Obsługi Studiów. Biuro Wsółpracy z Zagranicą: umowy i przyjazdy.* Master's thesis, Institute of Informatics, University of Warsaw, 2006.

[Lom08] R. Z. Łomowski, *Uniwersytecki System Obsługi Studiów. Moduł wyjazdy.* Master's thesis, Institute of Informatics, University of Warsaw, 2008.

[LSD] R. Willcocks, *Why use data compression for your web service?*, http://www.l-space-design.com/Articles/Why_use_data_compression_in_web_services.aspx

[MACEDIR] MACE-Dir, http://middleware.internet2.edu/dir/

[MLO] Metadata for Learning Opportunities, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/WS-LT/CWA15903-00-2008-Dec.pdf

[MLO-AP] ECTS Information Package/Course Catalogue MLO Application Profile, http://www.cen-isss-wslt.din.de/sixcms_upload/media/3050/Draft%20CWA%20Application%20Profile.pdf

[MUCI] Międzyuniwersyteckie Centrum Informatyzacji, http://www.muci.edu.pl/

[PENTAHOWIKI] Enabling SSL in Tomcat – Pentaho Wiki, http://wiki.pentaho.com/display/ServerDoc1x/01.+Enabling+SSL+in+Tomcat

[RFC1035] RFC 1035 Domain names – implementation and specification, http://www.ietf.org/rfc/rfc1035.txt

[RFC2798] RFC 2798 – Definition of the inetOrgPerson LDAP Object Class, http://tools.ietf.org/html/rfc2798

[RS3G] RS3G (Rome Student Systems and Standards Group), http://www.rs3g.org/

[RS3GWS] RS3G Workshop, Uppsala, http://wiki.teria.no/confluence/display/RS3G/UPPSALA

[SCHAC] SCHAC. Attribute Definitions for Individual Data, Working Draft v: 1.4.0 - 2009-03-26, http://www.terena.org/activities/tf-emc2/docs/schac/schac-schema-IAD-1.4.0.pdf

[SCOUT98] [#SCOUT-98] BusinessLifeCycleManager.saveOrganizations(Collection organizations) does not return exception list if failed to save, http://issues.apache.org/jira/browse/SCOUT-98

[SCOUT99] [#SCOUT-99] OrganizationS returned by findOrganizations() contain at most one name and description even if multiple names or descriptions are present, http://issues.apache.org/jira/browse/SCOUT-99

[SCOUT101] [#SCOUT-101] Faults are not handled properly by RegistryImpl.execute(JAXBElement<?> uddiRequest, URI endPointURI), http://issues.apache.org/jira/browse/SCOUT-101

[SGWV] C. Sgouropoulou, S. Grant, S. Wilson, G. Vangen, *European Learner Mobility Standardization: Sketching The Landscape*, http://www.fs.usit.uio.no/presentasjoner/EUNIS2009/FS-09-092_Sgouropoulou-Grant-Wilson-Vangen-113.pdf

[SITI06] A. Siaperas, P. Tissot, *A Distributed System for Issuing Europass Mobility Documents*. Retrieved: October 26, 2009, from: http://wiki.teria.no/confluence/download/attachments/13697206/Distributed+System+for+EUROPASS.doc?version=1

[SOAP11] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[Ste07] M. Steinbach, *Web Services, Output Formats and GZIP Compression*, http://www.sendung.de/archives/2007/04/09/web-services-output-formats-and-gzip-compression/

[UDDITC] OASIS UDDI Specification TC, http://www.oasis-open.org/committees/uddi-spec

[UIS] UNESCO Institute for Statistics, http://www.uis.unesco.org/

[UNICON] J. Wieland, moveon presentation for moveon Conference 2009 held on July 01, 2009 at University of Stockholm.

[TOMCATSSL] Apache Tomcat 6.0 – SSL Configuration HOW-TO, http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html

[WaBr05] C. Walls, R. Breidenbach, *Spring in Action*, Manning 2005.

[WIKIPEDIA] Wikipedia, the free encyclopedia, http://en.wikipedia.org/

[WSDL] Web Service Definition Language (WSDL), http://www.w3.org/TR/wsdl

[WSS] OASIS Web Services Security (WSS) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[WSSP] WS-SecurityPolicy 1.2, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html

[WST] WS-Trust 1.4, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html

[XMLSCHEMA] W3C XML Schema Part 0: Primer Second Edition, http://www.w3.org/TR/xmlschema-0/