Int. J. Engng Ed. Vol. 24, No. 4, pp. 747–760, 2008
Printed in Great Britain.

0949-149X/91 $3.00+0.00

GSEEM: A European Master Program on Global Software Engineering*

PATRICIA LAGO
VU University Amsterdam, The Netherlands. E-mail: patricia@cs.vu.nl

HENRY MUCCINI
University of l'Aquila, Italy

LJERKA BEUS-DUKIC
University of Westminster, UK

IVICA CRNKOVIC and SASIKUMAR PUNNEKKAT
Mälardalen University, Sweden

This paper presents a novel European Master programme on Software Engineering (SE), being put forward by four leading institutions from Sweden, UK, Netherlands and Italy. The Global SE European Master (GSEEM) programme aims to provide students with an excellence in SE based on sound theoretical foundations and practical experience, as well as preparing them to participate in the global development of complex and large software systems. GSEEM has been designed with two aspects of note: 1) the three specialization profiles in which the consortium excels: Software Architecting, Real-time Embedded Systems Engineering, and Web Systems and Services Engineering; 2) an innovative concept of 'shared modules', delivered all together by multiple institutions. Four types of shared modules are foreseen: 'parallel' twin modules, which run remotely between universities, 'shifted' modules, which teach SE concepts incrementally with shifts in study locations and timeline, 'complementary' modules in which complementary SE concepts are taught in parallel through shared projects, and 'common' modules, which share the presentations and the project. The profiles realize 'integrated knowledge' by complementing partial knowledge available at partner institutions. The paper presents some of the important issues faced during the design of the program and explains how GSEEM achieves the objectives of educating global software engineers. The lessons learned from the GSEEM design are of a technical, pedagogical and organisational or administrative nature.

Keywords: European Master Program; teaching Global Software Engineering; shared modules

INTRODUCTION

SOFTWARE ENGINEERING ADDRESSES the development of complex software systems by using a systematic engineering approach, providing theories, methods and tools for solving complex and often interdisciplinary problems. Since software permeates every aspect of our society, its engineering is of crucial importance for successful and efficient development of a large variety of products and services. The increased *globalization* of software development is creating software engineering challenges due to the impact of time zones, diversity of culture and communication, or distance, and requires novel techniques, tools, and practices to overcome numerous difficulties and to take advantage of the opportunities such environments entail. A new term has been recently been established to identify such globalization—Global Software Engineering—and new approaches have been introduced focused on this topic [19].

Based on current challenges and needs in the software engineering field, a growing demand for software engineers with different skills can be clearly seen. The development of large software-intensive systems is essential in many diverse business domains (including telecommunications, industrial automation, avionics and automotive industries, consumer electronics, in-house intelligent devices), different types of information systems, and web-based and service-oriented systems (such as e-government, e-health, traffic control, etc.). This is for example, expressed by the EU initiatives ARTEMIS [2] (Advanced Research and Development of Embedded Intelligent Systems) and NESSI [24] (Networked European Software and Services Initiative), the strategic research platforms for the EU Research Framework FP7. Further, appropriate and coordinated education in these areas is perceived as a major requirement [13].

The main objective of the GSEEM programme is to educate students to become excellent development engineers, service engineers, system architects, project managers, team leaders, information

* Accepted 25 April 2008.

analysts and policy advisors equally in Europe and in other continents. While such positions are very attractive, they are also challenging. They require not only excellet technical expertise but also social competence [21]. Typically, software engineers lead or are involved in team projects, which are often distributed, mobile, and have members with diverse skills, different backgrounds who may speak different languages. Software Engineering must address these challenges not only through the development of new techniques, practices and tools but also by educating a new type of a *global software engineer* with a wider critical view of learning, knowledge and experience.

The GSEEM programme came from teacher–student mobility agreements between partner universities under the Erasmus Socrates programme. Additional impetus came from the Erasmus Mundus programme [12], which supports selected European master courses and aims to prepare European citizens for life in a global, intercultural and knowledge-based society.

The academic programme is offered by a consortium comprising Mälardalen University (MDH), Sweden; University of L'Aquila (UDA), Italy, University of Westminster (WU), UK; and VU University Amsterdam (VU), The Netherlands. By building the consortium, the universities significantly added value to their existing individual curricula in a common curriculum, which covers different but highly related areas that are important for an integrated approach to global software development. The focus of the programme is on Software Architecting, Software Engineering of Real-time Embedded Systems, and Web Systems and Services Engineering. In the development of the common programme, and synchronization and management of differences between the sites, we have encountered many problems and met many challenges of different natures: academic, administrative, organisational and economical. In this paper we discuss the challenges and our solutions, as well as our innovative approach in education of global software engineering.

The outline of the paper is as follows: in the next section we briefly present related work. Then we introduce the GSEEM programme and outline some of the issues. The issues related to creating a joint curriculum and programme (academic, administrative, organisational, and economical) are given as well as a description of GSEEM solutions to such issues. The experience of the partner institutions in conducting shared modules is discussed. Finally, we give our conclusions.

RELATED WORK

Both industry and academia have recognized the need for adjustments in SE education, to effectively train the future generations of global software engineers. We are aware of only a few comprehensive approaches in this direction with isolated efforts and experiences worldwide. The ACM job migration task force stresses in [3] the difficulties in translating an educational response to offshoring into practical curricula, and points out that the US educational system is still in the process of identifying the curriculum changes that are essential to address application domain knowledge and a global workplace in order to keep its innovative edge.

Berglund [5] postulates fundamental questions on the what, why, how and where of the learning experience of computer science students. The case study was based on student interviews carried out in a distributed project course given jointly by Uppsala University, Sweden, and Grand Valley State University, USA, under the 'Runestone initiative'. Here, the focus is on the way that the students perceive their learning environment, to unearth the complex relationships between material taught in the course, and the many ways of experiencing an advanced learning environment like one including global issues. Navarro and van der Hoek [23] experimented with a project involving students with different specializations, hence adding the factor of interdisciplinary backgrounds. The work of Verkamo et al. [28] focused on distributed cross-cultural SE. They carried out a case study in two universities, one in Finland and one in Russia, where they observed that initial cultural differences are resolved by experience during the project.

Favela and Pena-Mora [15] with their distributed SE laboratory share with us similar experiences and observations. They established a similar kind of shared module in which a Mexican and a North American institute allowed students to carry out common projects. Like us, they encountered problems in communication, team building, use of technology and cultural differences. In their report they suggest five critical principles to help the students in dealing with distributed development. From these we learn that the students need to start the project with some kind of warm-up phase where they get to know each other, set their own collaboration rules and play with the communication tools available. This helps them to get prepared for the inter-cultural differences that they will encounter during the project, as emphasized by Kruchten [22]. Damian et al. [8] reported on a SE project involving students from Canada, Australia and Italy, and focusing on global customer–developer interaction. While their experience and findings are very similar to ours, they put more emphasis on tool support and the ability to carry out technical tasks in a distributed manner and in delivering high-quality results.

Although global SE has received recognition as a discipline both in practice and academia (e.g. [19]), the education of global software engineers has not yet reached the SE curricula. The valuable initiatives to update SE curricula on a global level (SWEBOK [6], ACM/IEEE [1]) do not yet address the emerging aspects of global SE. ACM Comput-

ing Curricula 2005 [1] mentions outsourcing, offshoring, and job migration merely in the context of the pace of change in the workplace and fails to acknowledge the fact that students need to be prepared for global software development. Similarly, the Software Engineering Body of Knowledge (SWEBOK [6]) conspicuously does not include any reference to global software development. Inputs from related initiatives should also be taken into account. For instance, human aspects in SE (as in [17]) should be brought into and extended for educating global software engineers. Herbsleb [18] describes a number of challenges in the technical coordination of globally distributed projects. Hawthorne and Perry [16] provide an interesting discussion of the education aspects that should be broadened to cover issues relevant to global SE: examples include ethical issues (not directly modified but exacerbated by distance and diversity), and use of SE techniques in a global context (such as requirements elicitation techniques, architectural design techniques, and project management).

DESIGNING THE GSEEM

The main objective of the GSEEM programme is to educate and train Masters students to become *global software engineers*, with excellent technical expertise and social competence.

For this purpose, the four partners in GSEEM have defined a joint European Masters programme in Software Engineering (beginning in the academic year 2007–2008), which will allow GSEEM students to:

- live and study in different EU countries in order to get accustomed to different working and living environments;
- experience technical challenges and social and cultural diversity;
- master the English language for effective communication;
- learn how to communicate in a global network, in a global team (especially using modern communication tools and media);
- interpret and sensitively exploit diversity in their professional (and personal) lives; and
- learn the challenges and regulations regarding not only a specific country, but throughout the EU (e.g., EU patents, EU privacy and security regulations, EU location-specific SE challenges and ongoing research).

In brief, our goal is to educate global software engineers to succeed in the global workplace. In order to reach such objectives and meet such requirements, the GSEEM programme is being designed to provide an integrated and harmonized list of modules from the four sites, covering the most relevant technical and social aspects of current software engineering. An appropriate framework and rules facilitating students' mobility was defined and validated.

To achieve this, many issues have to be properly addressed. These are of both an academic and non-academic (administrative, organisational, economical) nature. We discuss these issues and the solutions provided by GSEEM in the next two sections.

ACADEMIC ISSUES AND GSEEM SOLUTIONS

Teaching issues

Since its inception in 1999, the Bologna Process has put into action a series of reforms that are needed to make European education more compatible and comparable, more competitive and more attractive to Europeans as well as for students and scholars from other continents. The main goal of the Bologna Process is, by 2010, to create a European Higher Education where students can choose from a *wide* and transparent range of *high quality courses* and benefit from *smooth recognition procedures* [26].

GSEEM has been designed to accomplish the Bologna Process objectives towards the implementation of a double degree programme for the four partner universities. The issues we faced during the GSEEM design process can be summarized in the following.

- *Building a unified, harmonized and compatible view of different courses in different universities.* The four universities in GSEEM already ran recognized degree courses in Computer Science or Software Engineering. When building a joint programme, the main issues were: how to select the subset of modules to be taught in GSEEM, how to avoid modules repetition and how to maximize the professional skills acquired by GSEEM students; how to make this a unique program on the EU-level Masters and not just the sum of existing modules; and how to balance theoretical lectures with lab work and practical group projects.
- *Breaking the barrier of distance.* Communication is one of the most important issues when building a distributed Masters programme between four EU universities spread over four countries. We enable communication in two complementary ways: (1) by encouraging communication between the sites using modern internet-based communication technologies, and (2) by enabling the exchange of students and teachers. So far, the most relevant issues here have been: how to handle student mobility between the four universities; and how to adjust study periods, schedules and thesis preparation while limiting students' travel expenses.
- *Impact of cultural differences on the way courses are taught.* Students from different universities in different countries have different prior education, cultural backgrounds, and attitudes. Students in different universities have different

degrees of objectives, study independence, competitiveness, and project management abilities. Such diversities have to be understood, critically analysed and strategically used so as to make cultural differences an added value.
- *Breaking the barrier of communication: student–teacher (and student–student) relationships in an international study programme.* The student–teacher relationships vary from one country/university to another. They can be very formal and hierarchical or informal and egalitarian, student-centric or teacher-centric. Even the relationships between students can vary, when students from different countries are expected to work together. These issues have to be analysed in order to break down any barriers in communication.

GSEEM solutions

The four main issues listed above have been carefully analyzed and the following subsections describe how GSEEM handles them.

Building a unified, harmonized and compatible view of different courses in different universities

This has been one of the most important issues and our solutions are based on the three key concepts of (1) integration, (2) profile orientation and (3) shared teaching.

1. Integration. Software Engineering is a very broad area and very often SE programmes focus on particular aspects or domains. The idea of the integration approach is to effectively use the expertise accumulated at the four universities and build a common programme that can cover large areas but at the same time retain the depth and detail. From a technical perspective the programme offers an overall and integrated knowledge in Software Engineering. The integrated approach is achieved by two measures: (1) the partner institutions must provide a common basic knowledge that is implemented by a set of similar modules and similar teaching styles and procedures, and (2) by combining the specialties of the partner institutions in the Web Services and Embedded Systems with Software Architecture as the integrating factor the integration of these areas is a clearly visible trend in industry and research.

2. Profiles. The GSEEM programme provides Master education in Software Engineering with a focus on the domains that are strategically important for IT and IT-intensive products and services that are characteristic of today's society of and for the future. GSEEM provides insights into these domains both separately and in an integrated way by offering modules that cut across these areas. GSEEM has three separate Profiles:

- The *Software Architecting (SA) Profile* comprises the general principles for the analysis, design and management of large and complex software systems. These principles can be applied to various types of software systems and include different development approaches and techniques. GSEEM focuses on the modern trends that will dominate the future of software development including dynamic architectures, component-based software engineering, service-oriented architectures, interoperability and global system development.
- The *Real-time Embedded Systems Engineering (ES) Profile* is related to the domains of embedded and real-time systems. Such systems are increasingly present in all aspects of human life, as practically all of today's products include embedded software (from mobile phones, sensors and communication devices, to large products such as vehicles or aeroplanes). Such systems require knowledge of Software Engineering to provide solutions related to reliability, safety, resource usage, timeliness, predictability and similar properties. The Profile is focused on modelling and analysis of these properties and real-time embedded systems life-cycle.
- The *Web Systems and Services Engineering (WS) Profile* focuses on Web and Internet distributed applications and services. Like embedded systems, this class of applications is rapidly growing and plays an increasingly important role in industry and in everyday life. Web applications require specific solutions related to concerns such as security, integrity, timeliness, performance and usability. The focus of this Profile is on architectural engineering, interoperability, implementation, testing and deployment using component-based and model-based technologies.

Table 1 shows how the four universities contribute to the three Profiles.

3. Shared teaching and training. From a global perspective, the GSEEM students will experience the technical challenges and the social and cultural diversity of global development. The modules with accompanying projects are carried out in an inte-

Table 1. GSEEM Profiles

Profile	MDH	UDA	VU	WU
SA—Software Architecting	X	X	X	X
ES—Real-time Embedded Systems Engineering	X			X
WS—Web Systems and Services Engineering			X	X

grated way across a number of universities, either by running them in parallel and letting the students interact on a distributed network, or through mobility, by allowing the students to bring the technical and cultural experience that they gained at one university to another one.

The consortium is realizing the innovative concept of shared modules, each delivered jointly by two or three institutions, and resulting in common teams built from both students and teachers from different universities. Some modules intensively use distributed teaching and teachers' mobility.

Communication, collaboration, and coordination are the three main challenges in global SE. Co-located teams can collaborate using traditional data-sharing devices (e.g., whiteboard, projector, flip-board) and computer supported cooperative tools (e.g., shared knowledge database, project management tools, groupware). However, in a global and distributed context, collaboration becomes more difficult and novel means are needed [19]. In the GSEEM context, teachers and students will use innovative tools to participate in the shared modules (this method has been successfully tested in the ongoing collaborative teaching at MDH and is starting at both VU and UDA).

Four types of shared modules have been designed (see Fig. 1):

- *parallel twin modules run remotely between universities;*
- *shifted* modules, teaching SE concepts incrementally with shifts in study locations and timelines;
- *complementary* modules in which complementary SE concepts are taught in parallel through shared projects, and
- *common* modules, which share the presentations and the projects. The presentations are performed on site and sent online to other sites.

Different types of shared modules emphasise different aspects of global SE. Common modules correspond to virtual local organisations but they are physically dispersed with an intensive communication over the sites. The complementary modules correspond to typical solutions in global organisations in which different sites develop different parts of a product. Similarly, in shifted modules results from one site are acquired by another site. The parallel modules are not directly related to the global SE, but they indicate the diversity of the possible solutions.

The shared modules approach enables one to learn the technical and cultural aspects of SE by working in internationally distributed development teams. The GSEEM students learn both 'technical' Software Engineering and Software Engineering in a global setting. In other words, the programme combines the technical profiles demanded by the European IT market with the global knowledge demanded worldwide. We aim at educating software engineers who are global citizens aware of cultural, geographical, economical and environmental differences, who can sensitively exploit this diversity in their profession. The teachers are also able to experience diversity in education by participating in the teaching of different modules at different locations.

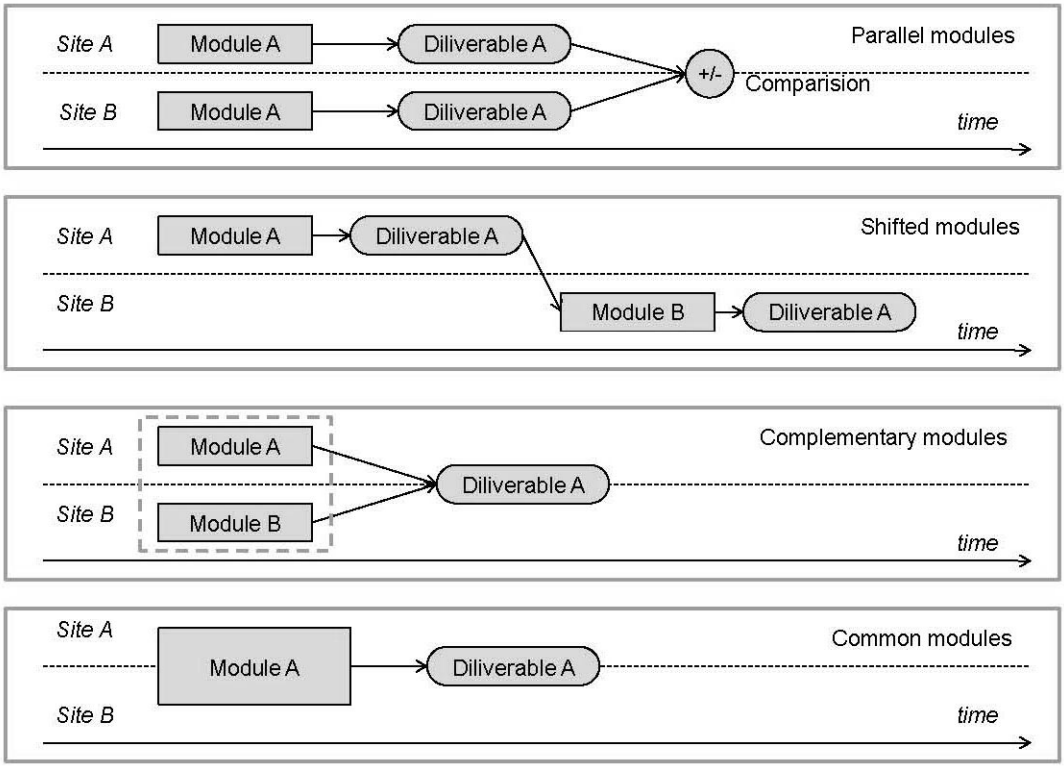

Fig. 1. Types of shared modules.

Breaking the barrier of distance

The GSEEM enables a great flexibility in mobility. In the application, the students indicate their preferences regarding Profile and mobility. The GSEEM Selection Committee chooses the enrolling university and the mobility path of each student, giving consideration to their stated preferences and merits, and the principle of equal distribution of students in each university.

During their studies the students study at two universities and obtain 50% ECTS at each university. After the first year at the enrolled university, the students move to another university, depending on which profile the students have selected. This is shown in Fig. 2.

The students do their thesis work in the second university, and they are advised by teachers from both their enrolled and their second university. In some cases, the teachers from all four universities are part of the advising team to achieve maximum use of the expertise available within the consortium with respect to the topic of the projects.

Impact of cultural differences on the way courses are taught

We have studied the students' previous education and background, and the attitudes in the four universities to avoid the risks of misunderstanding due to cultural differences, and to achieve a harmony in the work of the individuals and the teams.

Students' prior education and background. While all students wishing to be enrolled in the GSEEM need to have a Bachelor degree in Computer Science or similar, with (at least) 180 ECTS, we noticed that there are some differences between the courses in the types of modules and duration of project work at Bachelor level. The main differences that we identified consist in the degree of mathematics modules provided by the four universities (especially in WU compared with the other three universities). This could be a potentially relevant issue for WU students who are interested in moving to Italy during their second year of study. In fact, while WU provides 7.5 ECTS in Maths, UDA requires 56 ECTS in Maths (taking into consideration both Bachelor and Master courses) to gain a Master Degree in Computer Science. This disparity could limit the mobility of WU students to Italy, though those affected would only be those who also earned their Bachelor degrees from WU. GSEEM plans to provide sufficient opportunities for such students with these identified deficiencies to compensate by taking additional relevant courses, as feasible on a case-by-case basis.

Further, the VU students already include in their Bachelor's degree courses a considerable number of modules including (or consisting of) practical projects, where they learn project management and experience teamwork. These skills can be transferred to the international colleagues.

Student attitudes. There are some cultural facets that influence the way SE related projects are addressed and carried out by the students. A study has been conducted jointly by the VU and

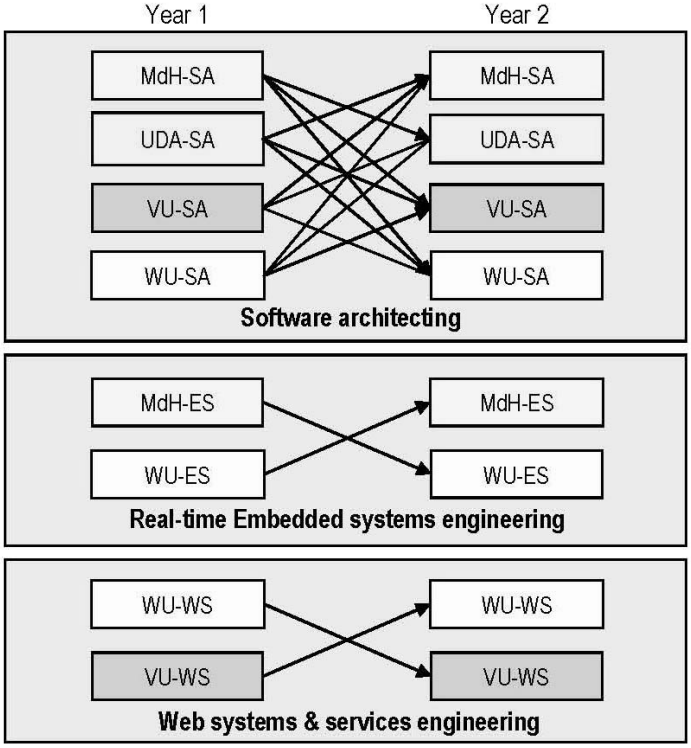

Fig. 2. GSEEM mobility.

UDA universities in order to understand how students are used to working in the two universities. A similar study has been performed at MDH, on the differences in attitude and way of working of Swedish students (students enrolled at MDH) and international students [7] (the 'Experience' section reports on both studies).

One aspect influencing teamwork is that the VU students like being independent. They highly value a certain degree of freedom in the way they manage work, deadlines, assignment of roles and tasks. The same holds for the way that project assignments are formulated: the possibility to autonomously deciding on the specific characteristics of the software application under development, or the technologies used, is highly appreciated. Freedom in the project increases motivation. UDA students are more academic, and more concerned with specific technical issues such as solving a design exercise.

A cultural aspect that sometimes influences the quality of the results in less positive ways is that the VU students perceive requesting help from the teacher negatively, and when they have cooperation problems they insist on hiding them until it is too late. The UDA students, instead, consider it normal (as part of the education) to ask support from the teacher. Lastly, the Dutch do not generally reward high grades. This results in a low level of competition: VU students often expect that completing a large amount of work ensures a sufficient grade, even if the resulting deliverables are of a very poor quality. The Italians, instead, look at the results: during job interviews, it is common for companies to ask for the grades obtained in specialized courses, and the Master degree grade.

Similar to Dutch students, Swedish students do not assume grades to be the most important aspect, but rather the attractiveness of the topic (either as a topic that is of interest for employment, or attractive in itself). They can be engaged during the teaching and even more during the project work if they find it motivating. They expect help from teachers when they ask, and they have no problem in asking for advice or for a help. Since Swedish society insists on equal rights, the students do not feel that they need to hide anything, but also that they have the right to decide for themselves what is best for them. This means that their own private concerns can be a good enough reason to do or not to do something required from the class. On the other hand, the international students are focused on the final results—the grades. So they perform for the best grade, i.e. what they think would be the most important factor in the teacher's opinion. For example, they are less interested in team work if this does not bring the direct results to their grade. As the examination period approaches the international students become more active, especially in their communication with the teachers.

Based on this experience, the GSEEM has been designed to take into account such diversities, and maximize students' learning outcomes from the programme. For example, the students have introductory lectures in which these differences are explained and discussed. We have also noticed that good results can be achieved with this diversity of students' opinions if the students and teachers are aware of them; in most cases the students adjust their styles and their goals according to the established rules (both official and unofficial).

Breaking the barrier of communication: Student–teacher (and student–student) relationship in an international study programme

Over the last two years, the GSEEM organisers have regularly visited the other universities, for dissemination, teaching and research collaborations. During such visits, we learned how student–teacher relationships vary from one university to another. In Sweden, for example, the student–teacher relationship is very informal: students and teachers call each other by their first names, and without any formality. In Italy, the student–teacher relationship is more formal, and students approach teachers with the appellative 'professor', even if not explicitly required and even address Ph.D. students and young researchers as such. At the VU University, students may influence departmental decisions by providing quarterly evaluations of their teachers.

Such differences have been carefully taken into consideration when designing GSEEM. Two tutors were assigned to each student, each one lecturing in one of the two universities visited during the GSEEM. Our intention is to keep the student–teacher relationships informal, following the Swedish model, but maximal flexibility will be allowed. Our current experience shows that in most cases the students and teaching staff adjust to the new situation quite easily and quickly if they are aware of the cultural differences. We have noticed that this adjustment tends go more smoothly with European students than, for example, Asian students. Some other aspects require more effort in explaining their importance, for example a sense of time, which varies between different cultures. From the teacher's viewpoint, the adjustments are not such a problem, since in most cases the teachers are also international researchers and are aware of the diversity in cultures. To reduce the risks for cultural mismatches we also prepared seminars for teachers given by guests from the partner universities in which the education systems and cultural differences are discussed.

NON-ACADEMIC ISSUES AND GSEEM SOLUTIONS

During the preparation of the common, integrated programme we faced a number of problems and challenges. Some of them were the result of

'objective' or permanent differences, because of differences in time, such as the start of the academic year, or in the differences in the performance of the classes, such as the division of the academic year into two, three or four semesters or study periods. Some differences were in admission requirements, some in examination requirements. The other challenges were related to the recognition procedures of programmes from other universities. While the academics were in general enthusiastic in building the common programme, the university administrations have approached the programme with more considerations and sometimes scepticism. Our goal was to provide one common programme that is transparent to the specifics at different universities, and at the same time compliant with the existing rules at each university. This required great flexibility in building the programme and mutual understanding between the sites. Fortunately, the Bologna Process was a great help: several of the partner universities were in the process of acquiring the Bologna system, and this process certainly has been going in the direction of decreasing the gap between the universities. Also, the fact that the universities were in a process of change has helped in the acceptance of some changes required from academics in order to make the programme as transparent as possible.

Here we discuss some of the issues we encountered and our solutions.

Administrative issues and solutions

The administrative issues were related to different types of recognition criteria: (1) admission criteria, (2) recognition of the Master's degree and (3) recognition of the dual degree. Again our goal was to achieve the same results from the students' point of view independent of which path and universities they choose.

Admission

The admission criteria are common for all four universities and are independent of the degree, selected profile and institution of enrolment. The criteria are as follows.

- The student must have a Bachelor degree or equivalent degree in Computer Science, Software Engineering or equivalent subjects with a minimum of 180 ECTS or equivalent points.
- The list of selected modules is controlled to make sure that any prerequisite modules have been passed. These modules (at least 60% of all modules) must belong to the fields of Computer Science, Software Engineering and Mathematics.
- They must have a proven competence in both written and spoken English to the university standard.

In addition to these standard criteria, we have identified a number of qualitative criteria. Our aim is to select the best students and this is to be achieved by judging their skills through their grades, CVs and the covering letters. Our experience is that a personal letter describing individual interests and vision differentiates students with similar grades. Furthermore, we prioritise those students with additional record of excellence (such as thebest students at the local university, possible involvement in research projects, visits to other universities and similar).

Recognition of modules and the Master degrees

Recognizing particular modules was not a problem since this had been resolved in principle by signing Erasmus/Socrates agreements.

Recognizing the programmes and the dual Master degrees is, however, a much more difficult task. To make the entire process possible we have divided it in two phases: (1) identification of the required activities and their formal approval from all sites; (2) carrying out identified activities. The only way to achieve this was through peer-to-peer recognition. While the identification of the activities went relatively smoothly, carrying out identified activities required different effort from different universities—from simple and fast recognition to complicated and long process.

The GSEEM programme is founded on joint curriculum design, which forms a basis for mutual recognition of the programme. Faculty Boards on all the sites have signed bilateral agreements that include the recognition of the joint programme and recognition of the modules in the programme and approval of the double degree. According to these agreements each party will approve the Master degree for a student who has obtained at least 50% of ECTS required for the Master degree at the local university and at least 50% of ECTS required for the Master degree at the partner university.

Recognition of the dual degree

The principle of the recognition of the dual Master degree was based on the principle that the students meet the requirements of the particular universities in which they study. These requirements differ slightly from university and university, and the differences are identified pairwise. Since students attending two universities must fulfil the requirements of both universities to obtain the double degree, they have the possibility of tailoring their curriculum by selecting different optional modules. In some cases, students can study at two universities but cannot obtain the double Master's degree without successfully completing additional modules. For example, requirements for a number of obtained ECTS in mathematical modules differ between universities. These requirements have been identified and are available to the students. In order to help the students to obtain the double degree, their mentors help them to devise their individual study plans.

Organisational issues and solutions

In addition to providing compliance with the particular universities and to use local existing

support as much as possible, the common integrated programme required the organisation of support for the specific needs of the students and coordination of collaborative provision. We have tried to introduce minimal additional administration. The GSEEM organisation is small and 'loosely coupled'; the representatives from each university—the GSEEM Local Chairs—constitute the Consortium Board chaired by a Coordinator (each year from a different institution).

Other organisational issues are related to students' mobility: their acceptance at the first and the second universities and their seamless integration into the new environment. Before the students move to second university, they are informed about the new university, the student life and general social context by a guest teacher from the new university. Also, through shared activities in the shared modules, the students acquaint themselves with the new environment. Upon arrival at the second university, the students also attend special introductory seminars organised by the universities. Several of the four universities provide such seminars for all exchange international students with a very positive result. These seminars usually conclude with a student party that aims to help with the assimilation of new students. The GSEEM Local Chairs together with the local Student–Staff Committee organise periodic meetings (twice a semester) with the students to discuss their work and any possible problems, and prepare and help them with moving to the other university. Furthermore, each student is allocated two personal tutors who monitor the student's progress throughout the course, offering advice and encouragement, highlighting problems, and determining jointly with the student any necessary remedial action.

Economic issues and solutions

Economic issues have been the most sensitive part of aligning the principles. Our primary goal that students are treated in the same way independently at whichever university they enrol at was impossible to fulfil. The tuition fee policy is different in different universities—from a case where there is a high tuition fee to a case where there is no tuition fee at all. For this reason the selected GSEEM policy is that the students pay the tuition fee to the university at which they first enrol. The students are counted as 'regular' students enrolled at that university. No money transfer takes place between universities. However, partner universities are responsible for ensuring that the number of visiting GSEEM students at the university is equal to the number of GSEEM students moving to other universities (so that the total number of GSEEM students present at the university is the same). This may cause some problems, in the case of an imbalance (more students either want to go or to come). Since one university without tuition fees can actually receive more students than it sends, and other three universities that have tuition fees can send more students than they receive, this problem can be solved.

To help students financially with additional costs when moving, we will try to utilize other possibilities of funding, such as Erasmus/Socrates and some national funding. For example, MDH has a funding for guest students in cooperation with local industry.

EXPERIENCE

Although we are in the early phase of implementation of the programme, each partner institution has ample experience in global education, especially in implementing the complementary and common modules.

Common module

For the last five years the common module Distributed Software Development [7] has been successfully used between MDH and the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (FER). Between 15 and 20 students participate each year from each site. Interestingly enough, the module has been shown to be especially attractive to international guest students; usually there are students with five to ten different nationalities participating in the module, which provides an additional multicultural dimension to carrying out the software projects. The module has been designed as a combination of lectures, guest presentations and distributed projects, executed jointly with the help of Internet-based tools. All elements in the module are preformed together by the two sites; the lectures are common and equally distributed between the sides; lecturers from both sites give the lectures and the lectures are transmitted using the video-conference system. Students from both sites participate in them. Finally the projects are undertaken together by both sites.

The lectures cover three areas: (1) technical, which includes modelling and design of software systems, in particular distributed and pervasive systems, (2) project management, which includes managing distributed software projects, project organisational issues and their relation to software architecture, and (3) cultural differences between people (in everyday situations, in business, in education). The guest lecturers are from international companies such as ABB, IBM, SAAB, and Ericsson Space, presenting their experience in distributed development, the challenges they meet, the lessons learned and best practices. Typically 3 to 4 guest lecturers give presentations.

The lectures give their basic insights into distributed development. The main focus of the module is project development. Each year the module offers several middle-size projects that are supposed to be performed by a group of 6 to 8 students over eight weeks. The projects include all phases of the soft-

ware development (requirements specification, design, implementation, integration, delivery and acceptance tests). The structure of a project is shown in Fig. 4. The students that work on a project are from the two sites: 3 to 4 students from each site. One of the students is the project manager who has the overall responsibility for the project. For larger groups a local project manager is often defined and they have the overall responsibility for the local site. One of the teachers acts as the customer (typically a younger researcher), and in addition there is the project steering group that is made up of teachers (typically senior researchers) from both sites.

The projects are performed in the following way: the project customer outlines the requirements, which are then negotiated and specified in detail by the project team. The project team reports the project state to the project steering group every week in common meetings sent over the video-conference system. The project members have the possibility of communicating using standard tools such as Skype, chat programs and, to some extent, the video-conference system. There is a common project structure based on CVS and the projects have their own web pages, which include additional support for sharing documents, discussion groups and similar means of communicating over the Internet. In general the main goal in the project is to remove the problem of the distance factor as much as possible.

The experience and repeatable analysis of the module, based on the students' performance, the project results, the project reports, data measured from the project database and the inquiries and student interviews, have indicated which elements of the module require to be specially considered, what are the difficulties, and what is the added value.

We have found that the following elements are more difficult to perform or achieve.

- The coordination of distributed projects, especially in the early phases is difficult and there is a risk that the project could start too late. This is because of the additional time that the project group needs to establish the relations and communication required for an efficient performance of the project. The reason is the unawareness of a local group of the other side, and a weaker sense of team spirit. We addressed this issue by introducing a warm-up phase to all students projects (as mentioned by Favela and by Pena-Mora [15]) where the students exchange contact information and availability, and got to know each other and assign roles and responsibilities.
- An efficient students' performance is threatened due to possible misunderstanding and cultural differences. The students may have different understanding of priorities, accuracy, the meaning of the verbal expressions and timing performance. For this reason we have introduced a procedure in the warm-up phase in which all team members explain their personal perception of the work to be done. For instance, they explain what they plan to contribute and what they expect their colleagues to provide in addition to the given support.
- The means of communication are still limited and cannot fully replace the direct communication. However, in most of the cases the problems are outweighed by higher student motivation for cooperation, and most of the results (the deliverables and the project performance) are of higher quality than one would expect. From the anonymous enquires made we have learnt that the students are very enthusiastic about the module, not only about the subject itself but also the fact that they can collaborate with people they never meet in person. On two occasions several students visited another site on their own initiative, which clearly illustrated

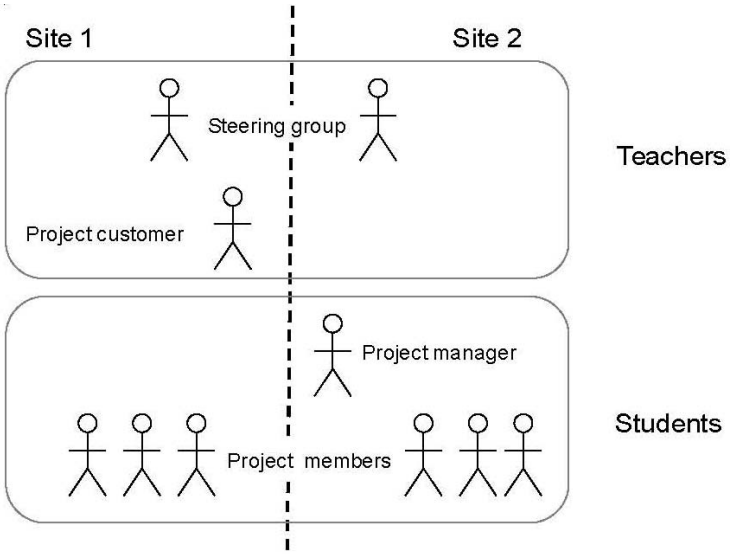

Fig. 3. DSD project structure.

their enthusiasm. More information about the module (and students' evaluations) can be found in [9]. Although during the classes we inform the students about the typical communication risks, and the ways to resolve them.

Complementary module

A complementary module, composed of modules 'Modelling of Web Applications' (MWA) and 'Advanced Topics in Software Design' (ATSD), has been taught since the academic year 2005/2006 at UDA and VU. Its main objective is to teach complementary SE concepts in parallel through common projects, and let the two student populations experience the issues around global development. The MWA module, for instance, teaches how to use UML for Modelling Web Applications. It covers twelve lectures teaching UML, and twelve lectures on specific Web modelling methods. Much attention is given to the quality and thoroughness of the delivered design, the ample usage of the taught modelling concepts, and traceability among different models. The ATSD module, instead, includes three lectures providing the basic knowledge about effective design documentation, design rationale and the rationale for the selection of UML diagrams, and five additional lectures on specialised design topics, amongst which is Web design. A great deal of attention is given to the quality of the design document, documentation of the rationale, correct and motivated use of the selected topic.

In the context of this complementary module, MWA and ATSD students work in teams in a common project especially designed to let them learn complementary SE topics from both MWA and ATSD modules. Experiments carried out in the last two years have shown that such combination of complementary module with shared projects improve traditional teaching in two ways:

1. *Learning by doing*: by working in joint UDA-VU teams, the students can put theory into practice and learn the contents taught in the other module. Secondly, they gain experience in how global design is carried out in a distributed setting (use of tools for distributed collaboration, effective documentation and communication). Lastly, they gain insight into the core issues specific to global software development, such as working with people of different cultures. In our module, the students from the two student populations work in shared international project teams, where tasks and responsibilities are assigned according to the specific background of each student population, but at the same time ensuring knowledge transfer from one population to the other. For example, the UDA students' main responsibility is using the MWA approach for the detailed system design, whereas the VU students have to analyse requirements, make design decisions, review the UDA design for feedback, and ensure documentation quality.
2. *Learning by osmosis:* by collaborating in international projects, the complementary SE contents are transferred more effectively between the two student groups, and the learning period is shortened. In our module, the material about Web design taught in 8 hours at the UDA is also summarized as one of the specialization topics at the VU (2 hours). Vice versa, the material about design documentation and rationale, taught in 6 hours at the VU, is also summarized to the UDA students in one lecture (2 hours). In this way, the two student populations have their specific in-depth background and a general understanding of the background of their colleagues. Hence, they have the means to learn by osmosis from their international colleagues. In addition, by assigning students from the two institutions to a common project team, they can come into contact with different work experiences, skills and cultures.

In addition, we used traditional classes to teach 'local' (i.e. module specific) SE contents defining the differentiating factors of the populations. A related issue is the need to define common evaluation criteria to be applied in both modules to assess the results of the shared projects. To this end, we defined common grading criteria for (1) the project deliverables and (2) the project teamwork. The first are: correctness, originality, internal consistency and traceability, documentation quality (e.g. UML correctness, essence, relevant decisions), and a link to the selected topic. The second are: teamwork planning (e.g. schedule and allocate work; arrange meetings; consensus on tool usage); team functioning; use of coordination and collaboration tools; role assignment; quality of the assessment of the peer-team work (local/international peers); quality of the support provided to the peers; and a questionnaire. In this way we can differentiate the quality of the technical results from the performance of the team.

Our experience so far led to the following observations.

- The international teams encounter problems in the initial phase of their common project. The main reasons for this were: the physical distance between the team members causes expected communication delays. Nonetheless, these delays were made worse by the fact that the students were locally busy with other modules, which were perceived as higher priority, simply because they were less distant. This was especially true for the VU students, who in their first Master's year have multiple modules embracing practical projects: they need some time to learn how best to distribute their efforts across the different projects. As explained, we introduced a warm-up phase at the start of the projects. In addition, the students are given study material about global software development, and participate in a brainstorming session together with the teachers, to discuss the global issues, both

expected and typical, together with the plans to solve them. The students are asked to reflect upon the global issues in advance, and come to the session with a list of issues and their solutions. In this way, we raise awareness and understanding.

- The international team members heavily relied on the correct and effective functioning of software tools supporting communication between team members. In some cases, problems with e-mail servers or with tool versions caused additional delays, and misunderstandings, resulting in a decreased quality of results. Solutions to this issue are: (1) to make the students aware of it, and (2) to let them fill in an on-line form including all contact information such as phone numbers, e-mail and chat addresses.
- We noticed that the students felt some 'lack of ownership' of their project results. Within a team if, for instance, the UDA students were responsible for the section describing the technical Web design, the VU students assumed that a design problem was not 'their fault' and did not feel responsible for finding a solution. Similarly, if for instance the UDA students completed their deliverable sections, they assumed they were finished and did not worry about the status of the deliverable as a whole. In general, the international teams have been 'too' effective in splitting tasks, and this decreased shared ownership. This also decreased, in our opinion, the learning-by-doing effect. Though this is a very difficult issue, we tried to solve it by introducing roles, especially for facilitating international collaboration and knowledge transfers. An example is the role of 'aligner' to check that, for example, the contributions of the distributed team members are consistent. The aligner must gain the expertise of his cross-border colleagues and must know who is responsible for what (a typical issue in global software development) to inform the relevant person about inconsistencies. By periodically rotating this role, all team members gain the same knowledge and experience.
- Geographical distance and separation of responsibilities further caused 'lack of team building'. This in turn decreased the learning-by-osmosis effect: without a sense of being part of a team, the students were less keen to broaden the interaction with their international partners beyond the purely academic assignment, and hence learned less from one another. Integrating roles like that of the aligner aim at solving this problem, too. Further, shared grading criteria (and grades!) increase the feeling of a 'common fate'. In addition, we are investigating the possibility of involving companies working in global settings to sponsor the project, so that the international team-mates can meet their colleagues in person (simulating the industrial pattern commonly called 'the travelling architect').
- Both the teachers responsible for the MWA and ATSD modules personally taught both the local full classes and the summarizing class abroad. For instance, in 2006 the lecturer from UDA visited the VU and gave the summarising lecture; the lecturer from VU gave the summarising lecture in tele-teaching, and the UDA students studied in advance the suggested material and prepared specific questions. This created a relationship between the lecturers and the students' population: by knowing the person assessing their work, the students were more motivated to participate in shared projects. This teaches us not to underestimate the importance of teaching staff mobility and the personal involvement of lecturers. Further, it seems that informal student–teacher relationships, which are common in North European countries and typical of scientific communities, favours team building.

CONCLUSIONS

Software Engineering is a broad discipline and in a Master's programme it is impossible to cover all its facets. There are two risks: either that of covering too many facets too superficially, or in focusing on particular facets and neglecting some essentials. The GSEEM programme avoids these risks by offering more modules and keeping them tightly integrated. From a technical perspective, the GSEEM provides a broader state-of-the-art knowledge than it would be possible to offer at each individual university. From a global perspective, it offers its students an opportunity to experience the technical, social, and cultural challenges of global development. The specialisation in the selected fields (web and service engineering, component- and model-based engineering, real-time embedded systems, industrial controllers, and mobile applications) makes it possible to obtain a deep technical knowledge and get insights into current and pertinent research issues. With specialised modules in research methodologies and professional ethics being part of the GSEEM curricula, we expect to create an impact on the quality of our graduates and societal relevance of their education. Our goals are that GSEEM students become aware of, and are trained to work with diversity (e.g. cultural, social, and economic), know how to communicate and collaborate in a global networked team, and can interpret diversity and exploit it in their professional and personal lives. To achieve these goals and to realise the idea of global, distributed education, a multi-dimensional perspective that addresses technical, research, pedagogical, social and organisational challenges is essential. Our experience so far and our expectations are that a systematic approach in analysis of these issues and taking appropriate and adaptive measures to solve these problems, makes it possible to build up a successful education and training programme in global SE.

REFERENCES

1. ACM/IEEE, Computing Curricula 2005, http://www.acm.org/education/curricula.html, (accessed 1 July 2008).
2. ARTEMIS (Advanced Research and Development of Embedded Intelligent Systems) EU initiative, http://www.artemis-office.org (accessed 1 July 2008).
3. W. Aspray, F. Mayadas and M. Y.Vardi (eds),*Globalization and offshoring of software*, Report of the ACM job migration task force, (2006).
4. P. Avgeriou, P. Lago, P. Kruchten, P. Grisham and D. Perry, Architectural knowledge and rationale: issues, trends, challenges, in *ACM SIGSOFT Software Engineering Notes*, **32**(4), 2007.
5. A. Berglund. Learning computer systems in a distributed project course. The what, why, how and where. Ph.D. thesis, Uppsala University, Department of Information Technology, (2005).
6. P. Bourque and R. Dupuis (eds), *Guide to the Software Engineering Body of Knowledge*, IEEE CS Press, (2004).
7. I. Crnkovic, I. Cavrak, J. Fredriksson, R. Land, M. Zagar and M. Åkerholm, On the teaching of distributed software development, *25th International Conference Information Technology Interfaces*, IEEE, Dubrovnik, Croatia, June, (2003).
8. D. Damian, A. Hadwin and B. Al-Ani, Instructional design and assessment strategies for teaching global software development: a framework, *Proc. of the 28th international conference on Software engineering (ICSE '06)*, (2006) pp. 685–690.
9. D. Damian and D. Moitra, Guest Editors' Introduction: Global Software Development: How Far Have We Come?**,** *IEEE Software*, **23**(5), 2006, pp 17–19.
10. Distributed Software Development Course, http://www.idt.mdh.se/kurser/cd5610/2006, (accessed 1 July 2008).
11. ECTS—European Credit Transfer and Accumulation System, http://ec.europa.eu/education/programmes/socrates/ects/index_en.html, (accessed 1 July 2008).
12. Erasmus Mundus, http://ec.europa.eu/education/programmes/mundus/faq/faq1_en.html, (accessed 1 July 2008).
13. European Commissioner Viviane Reding, ARTEMIS Strategic Agenda Launch Event, 06 March 2006, http://ec.europa.eu/commission_barroso/reding/docs/speeches/brussels-artemis.pdf, (accessed 1 July 2008).
14. European Students' Union (ESU), http://www.esib.org/index.php?option=com_content&task=blogcategory&id=42&Itemid=275, (accessed 1 July 2008).
15. J. Favela and F. Peña-Mora, An experience in collaborative software engineering education, *IEEE Software*, **18**(2), 2001, pp. 47–53.
16. M. J. Hawthorne and D. E. Perry. Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities, *Software Engineering Education in the Modern Age: Software Education and Training Sessions at the International Conference on Software Engineering (ICSE '05)*, St Louis MO, USA, May 2005, Revised Lectures. P. Inverardi and M. Jazaueri (eds), Springer-Verlag, LNCS 4309, (2006).
17. O. Hazzan and J. E. Tomayko, Reflection processes in the teaching and learning of human aspects of software engineering, *Proc. of the 17th Conference on Software Engineering Education and Training (CSEET '04)*, (2004), pp. 32–38.
18. J. D. Herbsleb, *Global Software Engineering: The Future of Socio-technical Coordination in Future of Software Engineering (FOSE '07)*, (2007), pp. 188–198.
19. *IEEE International Conference on Global Software Engineering (ICGSE '06)*, Florianópolis, Brasil, (2006), http://www.icgse.org, (accessed 1 July 2008).
20. I*EEE International Conference on Global Software Engineering, (ICGSE '07)*, Munich, Germany, (2007), http://www.icgse.org, (accessed 1 July 2008).
21. P. Inverardi and M. Jazayeri: Software engineering education in the modern age, *Software Education and Training Sessions at the International Conference on Software Engineering, ICSE 2005*, St. Louis, MO, USA, May 15–21, 2005, Revised Lectures, Springer, (2006).
22. P. Kruchten, Analyzing intercultural factors affecting global software development, *Proc. of (GSD2004) 3rd International Workshop on Global Software Development*, Collocated with ICSE, Edinburgh, Scotland, IEE, (2004), pp. 59–62.
23. E. O. Navarro and A. van der Hoek. Scaling up: How thirty-two students collaborated and succeeded in developing a prototype software design environment, *Proc. of the Conference on Software Engineering Education and Training*, (2005), pp. 155–162.
24. NESSI (Networked European Software and Services Initiative)—http://www.nessi-europe.com/, (accessed 1 July 2008).
25. Stanford Graduate Academic Life Survey, year 2004, http://gsc.stanford.edu/Old/Advocacy/Surveys/GradAcademicLife/, (accessed 1 July 2008).
26. The Bologna Process—Towards the European Higher Education Area. http://ec.europa.eu/education/policies/educ/bologna/bologna_en.html, (accessed 1 July 2008).
27. The GSEEM—An Overview, http://www.gseem.eu, (accessed 1 July 2008).
28. A. I. Verkamo, J. Taina, T. Tuohiniemi, Y. Bogoyavlenskiy and D. Korzun. Distributed cross-cultural student software project: A case study, *Proc. of the Conference on Software Engineering Education and Training*, (2005), pp. 207–214.

Ljerka Beus-Dukic is a senior lecturer at the School of Informatics, University of Westminster, UK, where she teaches courses in requirements engineering, real-time and embedded systems and software architecture. Her research interests include requirements engineering, semantic web, and real-time systems.

Sasikumar Punnekkat is a professor at the School of Innovation, Design and Engineering, leads the dependable software engineering research group, and is responsible for the Software Engineering Master programs at Mälardalen University (MDH), Sweden. His research spans various aspects of real-time systems, fault-tolerant computing, software engineering, software reliability and software testing.

Henry Muccini is an Assistant professor in Computer Science at the University of L'Aquila, Italy. His main research areas are on verification and validation, software architecture-based modeling and analysis, model-based testing, and fault tolerance.

Ivica Crnkovic is professor in software engineering and head of division of software engineering at Mälardalen University. His interests are component-based software engineering, global software engineering, embedded systems, and software engineering education.

Patricia Lago is Associate Professor in Software Engineering at the VU University Amsterdam. Her main research interests focus on software and service architectures, software architecture knowledge management and modeling.