Enhancing Software Engineering Education – a Creative Approach

Mario Žagar
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
Zagreb, Croatia
+38516129617
mario.zagar@fer.hr

Ivana Bosnić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
Zagreb, Croatia
+38516129631
ivana.bosnic@fer.hr

Marin Orlić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
Zagreb, Croatia
+38516129631
marin.orlic@fer.hr

ABSTRACT
Research in software engineering (SE) shows that freshly graduated students are usually not prepared to deal with problems occurring at the workplace. It is important to teach them how to construct the knowledge and solve the problems faced with. In this paper, we present two approaches used in SE courses at the University of Zagreb, Croatia: One approach is a distributed project-based course, where students from Croatia and Sweden work on the projects together, going through the whole life-cycle of creating a software product, while solving different problems, from technical obstacles to handling cultural differences. The other approach focuses on self-constructing the knowledge and presenting it to other students, with students' discussions and a group project in the end of the course. Both approaches enable the practicing of soft skills which in most cases are not adequately represented in SE education.

Categories and Subject Descriptors
J.1 [**Computer Applications**]: Administrative Data Processing – *Education*

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries – *Collection, Standards, User issues*

H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services*

General Terms
Design, Standardization, Documentation

Keywords
software engineering, education, e-learning, constructivism

SEESE'08, May 13, 2008, Leipzig, Germany.

1. INTRODUCTION
Software engineering education is exceedingly difficult to conduct, due to the fast-changing nature of the software engineering field. Students of software engineering are required to master basic skills of computer architecture, programming, system and application design. Every generation of students is faced with massive changes in what is considered to be the current fashion in software world, and every few generations a different set of skills is expected. Aside from technical skills, future engineers are also required to possess social, soft skills – ability to present knowledge, work in teams, learn from independent sources, listen to what others have to say, etc. While classical software engineering education focuses primarily on technical skills, our approach tries to include the basic soft skills directly into educational process and place the students in some of the situations they will face in their professional lives.

We can divide the educational process into four basic units:

- Lectures – tools, technologies and content for direct teaching,
- Exercises – tools, technologies and content necessary for the student to return the results to the teacher,
- Literature – books, manuals, textbook, descriptions, explanations, documents,
- Infrastructure and tools – systems and applications which help to consolidate the previous units and their functionalities.

In the following chapters, individual tools and technologies which help in handling these four course units will be described. The application of these concepts will be analysed on two approaches: the first for courses *Open computing* and *Computers and processes*, the second for *Distributed Software Development* course.

2. LECTURES
Lectures have a central place in the learning process – new concepts are introduced to students, and additional insight is given to the familiar ones. Instead of trying to keep up with each and every one of the new technologies, it is more proficient for

the lecturers to focus on the established base (theoretical or practical) and to interlace the lectures with presentations from researchers and experts from the industry that have more experience with cutting-edge technological developments. Lectures can be recorded and made available – this allows students to repeat the units they missed or found particularly interesting or more difficult. Such recordings should also be translated to other languages and enriched with additional material – experiments, further readings, and links to parts of the same or other courses.

Enriching the lecture materials enables the teacher to change the focus from presenting the subject to mostly passive audience, to discussing the subject with students who have basic understanding of the topic. This seems not much different from the classic *required reading* lecture notes, however, placing the course material in context of vast amount of information available over the Internet can create an additional awareness and ability in the students – awareness of the current developments and their historical significance (or insignificance), together with the ability to navigate through the multitude of information. Students should demonstrate their understanding, not in a form of tests, but presenting the chosen topics to their peers, arguing over the subject and defending their view.

Lectures should be accompanied with a search capability, necessary for creation of anchors on key moments, later used to link lectures with other material – literature, demonstrations, experiments, etc.

Key properties for the enhanced lectures are:

- Availability – lectures should be available anytime, from anywhere.
- Multilanguage – lectures should be subtitled in different languages.
- Added materials – lectures should contain additional material to help with topic understanding.
- Experiments – lectures should be augmented with practical exercises and experiments for students to watch or practise on; students should be able to recreate the experiments demonstrated during lectures and modify their properties.
- Search – the bulk of the lecture material should be searchable; it should be possible to find if a keyword occurs in a recorded lecture unit, and at what time.
- Tests – asking students questions of the subject taught increases the level of understanding, as the students are required to use their own words to explain the topic; tests can evaluate the comprehension of the subject, adoption of vocabulary, technical terms, etc.

2.1 Synchronous distance learning

Live lectures are delivered synchronously, to local and distant audience, using a wide range of technologies – both for personal (NetMeeting, Skype, …) and institutional (Polycom teleconferencing) use. Course professors, guest lecturers, and student presenters alternate in front of both audiences.

We are currently working on a system to support short quizzes given to students during the lectures or breaks between the lectures. This will help us gain insight into adoption of knowledge among students during the semester, and hopefully increase the level of attention during the lectures as well.

2.2 Prepared lectures

Prepared lectures are developed to cover the enriched lectures guidelines, consisting of everyday materials created using available equipment suited to the task – recordings (video/audio), audio converted into text (Croatian and English), presentations, images, drawings and animations, comments and references (links).

As a result of the use of SMIL format based on XML, the World Wide Web Consortium recommendation for multimedia representation on the Internet, all the materials are integrated into multimedia resources which can be viewed through players such as RealPlayer or another SMIL player. Video and PowerPoint presentations are displayed in two windows, together with additional information and links, while the window at the bottom displays text (in Croatian or English, whichever the user prefers). This solves the problem of multilingual instruction as well as accessibility for the users with special needs.

During the development of multimedia lectures, all the speech is transcribed, in order to make the presentations accessible to persons with special needs (hearing impaired), as well as to enable multilingual support, taking into consideration the cooperation with a Swedish university and joint instruction on a course. In addition, this enables detailed search of such presentations.

The subject matter was prepared sequentially, in small modules, and is suitable for organisation in various orders – learning paths. The content can be viewed in the local work mode (on a PC) but it is primarily intended for and integrated into LMSs such as WebCT and Moodle. The concept alone is not specifically related to any LMS, and it can be adapted to any selected system.

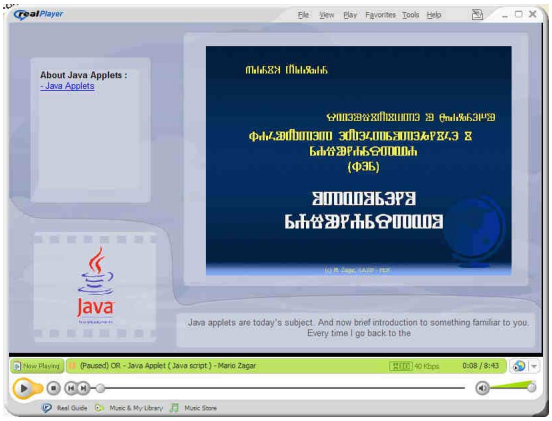

Figure 1. Screenshot of a multimedia presentation

This approach to the development of educational materials also enables searching of the content, in accordance with which a presentation indexing and searching system was developed. Full text search allows the user not only to find the presentation, but also the exact moment in which a certain word or expression is

mentioned. This enables fast access to the necessary information in the lectures.

The indexing/searching system enables the authors of SMIL presentations to define the scope and to group the materials for indexing. The teachers can have multiple search engines which they can personalise and choose what content they wish each search engine to contain. A search engine is simply introduced into the existing page within any LMS or CMS wherever the teacher desires, using client-side technologies, with a simple design adaptation system to the design of the current page.

Figure 2. Searching the multimedia lectures

3. EXERCISES

Exercises form a practical angle of students' experience, providing the ability to see in practice what they have heard at the lectures. Exercises should be presented in a credible remote experimentation environment for students to work in. Experiments created in such environment should not differ significantly from actual situations students might find themselves in – simulation/emulation is a valuable tool but students must be exposed to a real, non-simulated situation at some time. Classical lab approach is not always possible – due to space, personnel or maintenance constraints. Additionally, students might want to see and try for themselves the experiments demonstrated. The exercises should be available remotely, anytime, anywhere. Even further, the student's progress with the exercise could be automatically monitored and graded – either with quizzes regarding the topic (to test the theoretical knowledge), or with automated tests checking the validity of accomplished work. Computer as a mediator between the experiment and the student can also form a safety net for the student – computer can act as a personal trainer on equipment usage, recreate the examples given during lectures, check students' inputs for errors that would otherwise injure the student or damage the equipment, and so on. Key properties for such exercises are then:

- Availability – depending on the available equipment, lab should be made available to students anytime, anywhere.
- Credible remote environment – student experience should be based on real equipment, not only simulated one.
- Experimentation – all demonstrated experiments should be available to students to play with.
- Tests – student comprehension can be tested prior, during and after lab exercises, automatically.
- Added materials – portions of lectures and other materials can be integrated with the exercises, so students can move back and forth between them to gain better understanding of subject's theory and practice.
- Multilanguage – with much less text than lectures or literature, the exercise descriptions can easily be translated to other languages; all support systems should be multilingual, however.

The RASIP remote laboratory was developed for students to perform exercises on various devices – computers, microcontrollers, etc. – where the students can work on a special platform, experiment with examples, do exercises and, finally, be graded in the comfort of their own room. Remote laboratory is also referred to as 'virtual' as it is designed to serve as a platform to access both physical devices, as well as simulated ones.

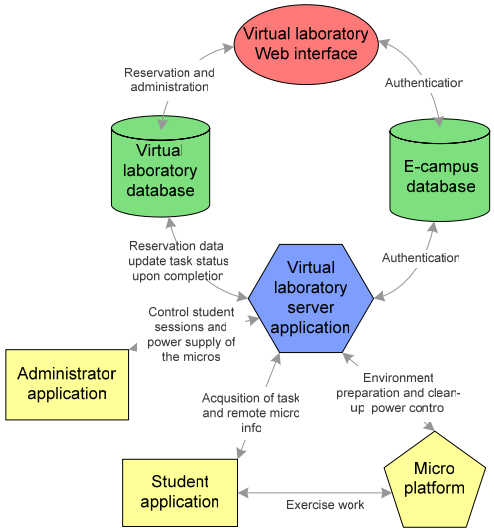

Figure 3. Virtual laboratory scheme

Currently available remote platforms are based on several microcontroller platforms such as network (TINI, Rabbit) and classic microcontrollers (Atmel). A program that was written can be translated, saved, performed, its performance controlled and evaluated. It is important to notice that, while doing so, the students are not working on a platform simulator, but on a real microcomputer system (which they can see by using a video camera on the Internet), in the time slot reserved for that student alone.

4. LITERATURE

Literature is perhaps the easiest to convert to a new, modern form. Paper books have long been converted to digital form, made available on course web sites and expanded to include links to other sources or multimedia materials. Even with such extensions, books are still static, preformatted, previously prepared units. Software engineering books, for instance, contain code examples, usually included on a CD. Such examples would be easier to understand if an example could be, while reading a book, executed with just a few clicks, and then, perhaps, modified and tested again. A book like this must be a digital online book integrated with its samples, adaptable to reader's preference. Enhanced books should therefore be:

- Accessible – readable visually or with text-to-speech synthesizer.

- Adaptable – to reading agent/device, screen size, user's preference of colour scheme, font size, formatting, pagination, etc.
- Enriched – complete with semantic annotations to make it searchable and usable as a data source for outside referencing.
- Live – integrated examples should be available to reader to inspection and experiment with.
- Based on standards – integration with other data sources or sinks is possible only if widely adopted standards are used for all levels – topical, presentation and semantic data.

The concept of the digitally distributed RasipBook is the step in this direction – the book now also includes separation of the content from the presentation, selection of style, format, device or media it is used in (PDAs, mobile phones, print), *live* performance of the examples from the book (UNIX commands, HTML), performance of students' examples based on the example templates (HTML, XML, PHP, Java), reading of the text (text-to-speech) and links to multimedia content and vice versa.

The book can be adapted to the user preference according to various parameters. Along with all the supplements, such a book represents a dynamic structure.

Figure 4. The book adapted for use on a PDA

Figure 5. The same book, adapted for use on a mobile phone

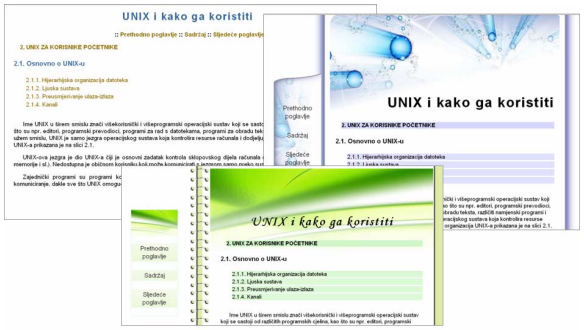

Figure 6. Different layouts of the book

Apart from distributed textbooks, RasipBook format is also used for writing papers and project documentation. We try to make the production as easy as possible, so the authors do not have to create content files directly in XML (although that is possible). The easiest way to create RasipBook materials is to write in Word, using a special template and save it in RTF (Rich Text Format). Such a document is uploaded to the content management system, which converts the RTF file into RasipBook format and adds transformation files and styles defined in advance.

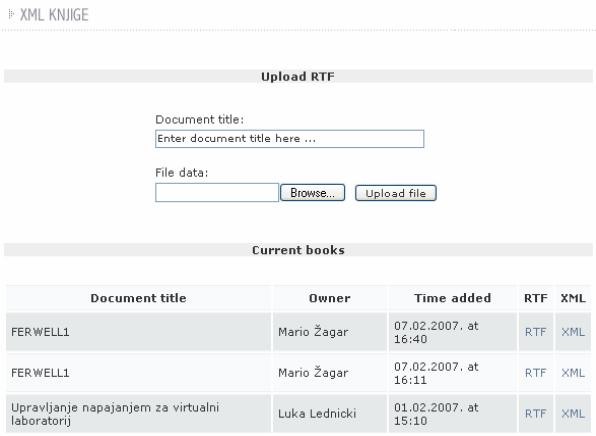

Figure 7. Exporting RTF into XML format

The current research on this topic is integration with an OASIS standard called DocBook, a semantic XML schema for publishing books and other written documents. It is possible to convert RasipBook to DocBook and vice-versa, depending on the user's needs.

5. INFRASTRUCTURE

Finally, the infrastructure used to build all this should exhibit all common properties of other elements – it should be:

- Standards-based – to allow integration in- and outside of the system.
- Internationalized – to allow multilanguage support.

The platform and framework should serve as a basis to build and integrate required tools and elements – search engines, collaboration and learning systems.

The central infrastructure point is the content management system: FER **e-Campus**, an integrated technical infrastructure enabling, providing and supporting information, cooperation and e-education. Since the very beginning, the idea behind e-Campus was to provide the teachers and students with easier use of technology and e-learning tools in instruction. The core of e-Campus is **Quilt CMS** which enables the publishing of news, files and other content related to various forms of instruction.

Apart from Quilt, an important part of e-Campus is tight integration of CMS with various LMSs (Learning Management Systems). At the moment e-Campus integrates three systems:

- **WebCT** – one of the best proprietary tools
- **Moodle** – currently the best open and free tool

- **AHyCo** – e-learning tool developed at FER, based on adaptive hypermedia

Figure 8. Moodle LMS in the Quilt CMS interface

For all three systems an identical interface was developed through which the users (students and teachers) are administered and the courses managed. Integration of all three systems allows for automation of user administration, so most of the instructor's tasks related to adding students to courses have been minimised and are now a mouse click away. Two systems are used intensively: **WebCT** (with which the LMS integration started) and **Moodle**, to which, according to plans, all courses will be transferred in near future. Experimental teaching of the *Open computing* and *Computers and processes* is also conducted through LMS, with all the lectures in SMIL (Synchronized Multimedia Integration Language), as well as assignments, self-tests and communication with the students.

The role of FER e-Campus is particularly important as the backbone of asynchronous communication between the teachers and students of two different universities in the Distributed Software Development course. The multilingual support enables the usage of the system by students from different countries and speaking different languages.

Figure 9. Use of Quilt CMS in distance learning

FER e-Campus is entirely developed in-house using open source tools and technologies and other Web standards, such as RSS and RDF for content syndication or XML-RPC for integration with the other systems.

6. e-RIP AND e-OR COURSES

e-RiP and **e-OR** represent experimental instruction of the *Computers and processes* and *Open computing* courses, in which every year participates about 20 students per course. The courses are designed as a mixed-mode learning, where *online* part includes the multimedia lectures, discussions and self-tests. The other part is face-to-face, including students' presentations of the new course themes, live discussions, and project work.

The lesson cycle in such courses always starts with online lectures made in SMIL format, published on LMS. Such lessons are divided in parts up to 15 minutes long, each containing additional information and links to other knowledge sources. Students can check their newly gained knowledge by taking self-tests, which are not a part of the grade.

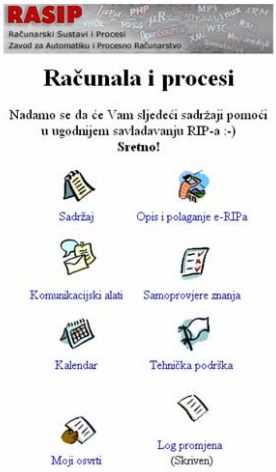

Figure 10. Instruction in WebCT

In the same time, the online discussions are beginning to develop. Usually the teaching assistants motivate the students by proposing themes that build on the theoretical knowledge, but have the practical value at the same time. For instance, students discuss the possibilities of applying some technology, or find other, newly developed technologies which have the similar purpose or roots. These discussions involve a high level of self-learning, by finding relevant information on the Web. Although not a primary objective, students also gain experience in questioning and argumenting their attitudes or proposed solutions, which is an important part in the professional life of a computer science engineer.

As it is hard to keep the course completely up-to-date because of new technologies emerging every day, the next step for the students in each course cycle is to propose a theme, connected to the lectures but not elaborated there, explore it and write a short overview and an analysis, such as good and bad sides, recommended usage, comparison with similar approaches, etc. This part enhances students' professional writing skills and prepares them for writing good reports, elaborates and papers, which is becoming important not only for researchers and managers, but for almost everyone in IT sector.

Whether you are an engineer who is informally presenting his ideas to the group of co-workers, a project manager whose task is to convey information to the upper-management, or a manager who is about to sell a product to the customers, presenting is ubiquitous. It is obvious that in the curricula of Croatian computer science faculties, such soft skills are somewhat neglected. These courses make students practise presenting the knowledge gained by writing overviews to their colleagues in the end of a lesson cycle (once a month). The teaching assistants try to advise them on issues such as badly written presentations, stage

frights, proper gestures, etc. A live discussion is being brought after the presentations, where students can note other persons' opinions and reformulate their own. Students highly regard the opportunity to present their views to the "public" and find it as a good practice for the future. The disadvantage of this approach is that the students have a high level of freedom regarding the choice of topic of their presentation, and often lose focus on the course subject.

Constructing knowledge is a matter which should be practised and encouraged. While constructivism model of learning is still not very popular in Croatia, some courses tend to include it, at least in a part of the activities. Constructivistic methods can be observed in the previous learning activities, but the main constructivistic point comes near the end of the course. The students are divided in small groups of about 8 persons, and they are given "a real" task - a project which in the end becomes a model of a concrete product. In a course called *Computers and Processes* this project can become quite challenging. The examples include:

- designing a model of a "hi-tech" remote-controlled car. Students decide on features of a car, including different sensors, LCDs, remote control, RF transmitters/receivers, microprocessor, etc. They should take care that the components can fit together on all criteria. The financial side of the story should be regarded also, so the components' cost is analysed, based on the actual data acquired from the local electronic stores or Web.
- designing a model of a mobile phone. As mobile phones can have all the features imaginable, it is important to define the target audience and analyse their needs first. Components of a phone depend on each other, as a wireless module would make the battery time shorter, a nice, big LCD would make it hard to put the mobile in the pocket. A GPS module could be handy, but is it really needed for this class of phones? How will it affect the price? How are real products with similar characteristics actually priced?

In every group project work, students encounter similar problems regarding the exchange of ideas, distribution of work, combining different knowledge, handling conflicts, etc. They are also to take over the initiative, in order to make a good project. At first it seems overwhelming, especially for those who didn't have the group work experience before, but in the end, things line up a lot, and the better work process quality can be achieved. A detailed project documentation is required, with an analysis of the system and financial costs.

A final group project presentation takes place in the end of a course – a chance for all participants to show their best. A commission of about 5 teaching assistants listens to the presentations, which sometimes include some marketing ideas, such as giving leaflets of a new mobile phone model. The project evaluation is based on different criteria such as presentation delivery, technical characteristics, feasibility, usability or price-market relation. The final projects in *Computers and Processes* course aim to combine software engineering with more high-level hardware approach, while the projects in *Open Computing* course is strictly software-based.

The final course grade is a combination of all the learning activities, together with a mid-term and a final exam.

Here are some student experiences regarding these courses:

"...presentations are the move of the year. The attempt to listen to myself (am I making those silly gestures, touching my face and staring at the floor? and (what the heck!!!) am I sitting while I am holding a presentation?? aaaaaaa!!!)... Truly wonderful,"

"On a scale of 1 to 10, I would grade the whole idea with 9 or 10… because it stimulates creativity, enables group work and presentations of the topics that were covered."

"Innovative and creative instruction delivery in which the information is presented to the students, but they are not required to repeat word to word what they heard, rather to integrate the acquired information into their point of view."

Figure 11. e-RIP: Developing the final project

7. DISTRIBUTED SOFTWARE DEVELOPMENT COURSE

Distributed Software Development (DSD) course was developed as a result of cooperation between the Faculty of Electrical Engineering and Computing from Zagreb, Croatia (FER) and University of Mälardalen (MdH) from Västerås, Sweden.

Figure 12. DSD course - Croatia and Sweden

DSD is a distance learning course, where lectures and practical work are conducted remotely, since the course deals with distributed software planning and development. Demand for such knowledge and skills on the market is extremely high, as a

large number of IT companies and research teams in Croatia work either with or for foreign companies in software development. Such knowledge is essential for large global international companies such as IBM, Siemens, Ericsson, Microsoft, ABB and others. The reasons range from economic to the fact that projects are to become more multidisciplinary, which makes it hard to gather all the skills required for a project in one place.

There are many obstacles for successful software development, particularly if the teams are distributed over several geographic locations – ranging from another faculty within the same university to another continent. Some of the obstacles are technical, while others concern cultural and language differences. The course analyses the obstacles and enables the students to acquire experience on how to recognise them through practical work with programmers from another environment.

The beginning of the course is quite short, and this is the only time when the "lectures" come to the stage. There are three kinds of lectures:

- standard lectures about the course, technology and methodology about to be used, delivered by professors from Croatia and Sweden
- lectures delivered by the guests from industry, who take part or lead distributed projects, sharing their experience with the students
- lecture about the "human" problems in distributed projects, with an emphasis on the cultural differences

The course is conducted through synchronous Internet videoconferences in teleconferencing rooms and asynchronous possibilities of the FER CMS system. Besides video transfer, the application sharing is available also.

The course achieves a two-way communication on several levels: not only in the delivery of instruction, in which the professors from Croatia and Sweden teach to both the Swedish and Croatian students, but also through the cooperation of students on joint practical software projects.

There are three types of projects:

- A large project consisting of subprojects on which students from both countries participate
- Projects consisting of students from both countries
- Smaller projects consisting of students from just one country, who are supervised from both Sweden and Croatia, in order to simulate a market situation in which project work and development are conducted in one country, and managed from another.

In the moment of project start, the roles of teachers are changing: the professors become "customers", teaching assistants become "supervisors". Students become "workers", led by one of them, appointed as a "Project Leader". The projects simulate the "real-world" environment, where the project is not defined in details. Developers should gather the requirements, create the project plan, analyse the risks of project failing, define milestones and deliverable dates, etc. Of course, everything the students promise is expected to be delivered also.

During the course, project presentations are delivered on a regular basis, to present the project state and persuade the customers that the project is "worth buying". Besides a great product, a great marketing can also help to sell it better, as in a real world. As the whole course is in English, which is not the mother tongue of any of the participants, communicating and presenting in a foreign language adds to the course complexity.

Once a week each group can use teleconferencing resources for an hour in order to enable synchronous communication with the members of their teams from another country, while the rest of the communication is conducted through the FER CMS, e-mail and various instant messengers. As DSD is about software engineering, standard methods for distributed development are put in practice, such as regular use of versioning software (CVS, SVN...).

Besides usual communication problems, such as choosing the right means of communicating, cultural differences are an extension of challenges to be solved by both students and teaching staff. As a lot of foreign students enrol to Mälardalen University, it is not unusual to have 5 to 10 different nations involved in a course per year. Problems with language barrier, understanding of different customs and ways of achieving the goal can arise, They should be solved during the course in an optimal way. Students are expected to experience how cultural differences can affect the project work and realize that diversity should not be something "to overcome", but to be integrated in a real-life situation.

The project ends by presenting – "selling" the final product and deploying it for the customers to test it. The product is accompanied by a handful of documents, such as installation manual, technical documentation, requirements document and test cases for quality assurance. The students are taught that the sole product is not the only thing that matters in software engineering, but also the documentation, testing and support.

The project evaluation is based on a detailed table consisting of up to 30 criteria, among which are presentation delivery, usability, documentation quality, project process and communication quality. The overall points are distributed among the group members with the help of the Project Leader. As one of the implicit course objectives is that students have the opportunity to "make mistakes and learn from them", an element in the final grade is a quality of a questionnaire – a document where students through a series of questions describe their experience of distributed software development, problems encountered and the ways of solving them. The big emphasis is on ways of better accomplishing the next distributed assignments, which are supposed to be in some "real-world" company, after the graduation.

The differences in curricula of the involved universities do not pose a significant problem, as the course is held at the final semester of the studies. The differences in student skill levels are resolved by teamwork. At the course beginning the students are given a short poll to determine their skills and competencies. This poll is later used to form balanced teams. Projects are assigned to teams based on their skills and required knowledge.

Students who enrol into the course at FER, by virtue of enrolling also become "Swedish" students on the DSD course and

vice versa (and get ECTS points for it). This course provides a basis for the use of such form of international cooperation in other FER courses, as well in courses on other Croatian faculties.

Here's what DSD students said about the course:

"This course was one of the best that I had on faculty."

"I was scared at the beginning, but now I feel lucky I had been a part of this course."

"Another week and I would have died."

"...I found out that it takes a lot of work to make something work as you want it to work."

"I learned much more than I expected. I learn lot of team working, good organization of work, time and documentation importance."

8. CONCLUSION

The software engineering is supposed to follow the current SE trends, in order to give students the appropriate basis for their future on-spot learning. New media and technologies offer a variety of ways to support the new, creative ways of learning, mostly based on project work done in groups, similar to real-life environment. Such an approach requires a lot of effort from the teaching staff and additional resources, but increases the quality of SE education and gives future young engineers a valuable experience. The depicted methods have proven very engaging for the students, and very demanding for the staff at the same time. Prepared lectures, remote experiments and enhanced digital books are usable with large student groups. The student presentations are most effective for smaller groups of students, when it is not too tiresome to follow the talks given.

9. ACKNOWLEDGMENTS

We would like to thank the Swedish part of DSD team, Prof. Ivica Crnković, PhD and Rikard Land, PhD, for the collaboration and great support in the Distributed Software Development course.

This work is supported by the Croatian Ministry of Science, Education and Sport, under research project "Software Engineering in Ubiquitous Computing"

10. REFERENCES

[1] Karunasekera, S., Bedse, K. 2007. Preparing Software Engineering Graduates for an Industry Career. Software Engineering Education & Training, CSEET '07. 20th Conference on, (July 03-05, 2007), 97-106

[2] Razmov, V. 2007. Effective pedagogical principles and practices in teaching software engineering through projects. Frontiers in education conference - global engineering: knowledge without borders, opportunities without passport (October 10-13, 2007), S4E-21 – S4E-26

[3] Crnković, I., Čavrak, I., Fredriksson, J., Land, R., Žagar, M., Akerholm, M. 2003. On the Teaching of Distributed Software Development. Proceedings of the 25th International Conference on Information Technology Interfaces, SRCE University Computing Centre, 237-242

[4] Čavrak, I., Land, R. 2003. Taking Global Software Development from Industry to University and Back Again, Proceedings of ICSE 2003 International Workshop on Global Software Development (GSD 2003), Portland SAD

[5] Prikladnicki, R., Jorge L. N. A., Evaristo R. 2004. An empirical study on Global Software Development: Offshore Insourcing of IT Projects. ICSE 2004 Workshop, Global Software Development, Edinburgh, Scotland

[6] Vaughn, R.B., Carver, J. 2006. Position Paper: The Importance of Experience with Industry in Software Engineering Education. 19th Conference on Software Engineering Education and Training Workshops (CSEETW '06). 19 - 19

[7] Tomić S. et al. Living The E-Campus Dream. 2006. European Distance and E-Learning Network (EDEN) Conference Proceedings, 644-650

[8] Postema, M., Miller J., Dick, M. 2001. Including practical software evolution in software engineering education.. Software Engineering Education and Training, 14th Conference on, Proceedings, 127-135

[9] Fudurić, D., Žagar, M., Sečen, T., Orlić, M. 2007. Remote laboratory hardware modules based on networked embedded systems. International Journal of Online Engineering (iJOE). 3 (2007) , 3, 1-5