Technical and Managerial Principles of a Distributed Cooperative Development of a Multi-Lingual Educational Course

K. Bothe[1], K. Schuetzler[1], Z. Budimac[2], K. Zdravkova[3], D. Bojic[4] and S. Stoyanov[5]

[1] Humboldt-University Berlin, {bothe, schuetzl}@informatik.hu-berlin.de
[2] University of Novi Sad, zjb@im.ns.ac.yu
[3] University "Sts. Cyril and Methodius", Skopje, keti@pmf.ukim.edu.mk
[4] University of Belgrade, bojic@EUnet.yu
[5] University "Paisii Hilendarski", Plovdiv, stani@ecl.pu.acad.bg

Abstract. Seven universities of four countries are developing a joint course on software engineering which is web-based and has to cope with the challenges coming from different languages of the participating parties as well as with the distribution of work over the internet.
This article describes the experience of this ongoing project in a generalized form to offer advice to similar forthcoming projects.

1 Background

There is an ongoing educational project with participants from seven universities of four countries with the goal to develop a joint course on software engineering. This course is web-based, i.e. all course materials are available over the internet. Due to participants coming from Germany, Serbia and Montenegro, the Former Yugoslav Republic of Macedonia and Bulgaria, two main problems had to be solved: the four languages involved and the coordination of distributed work.

This project has been realised with the support of the "Stability Pact for South Eastern Europe" and has started in 2001. The original of the course material already existed at Humboldt-University of Berlin: However, in German and adapted to the specific situation at Humboldt-University. Nevertheless, this material proved being useful as a starting point. In particular, reuse of this material resulted in reduced efforts needed to develop slides and in a transfer of methodological principles. By now, the first version of the course has been finished and first lectures based on this material have been held successfully. As hoped and expected, involving several partners has contributed to improvements and extensions of the original course material.

This article evaluates the experience of this project and describes the principles on which the work has been based. To support similar projects in other fields besides software engineering, we will not confine ourselves to the actual work done. We will instead present a generalized list of principles which may prove useful in similar projects, too, and we will give explanatory details from our project for particular points on this list. We will classify the items on this list on being either technical or managerial principles. Before discussing these priciples in detail we will at first cover the involved course material in the next section.

2 Overview of the Course Material

The project web-site contains different kinds of information connected with the joint course. This information concerns the course as well as aspects of project organisation. Figure 1 provides an overview of the different fields involved.

Information about the participants, the schedule, basic principles, FAQ and the discussion forum serve the project organisation. The slides make up the core of course materials. There is a close connection between the slides and the involved case studies, assignments and recommended literature. An overview of the contents is provided through the topics and in more details throughout the syllabus.

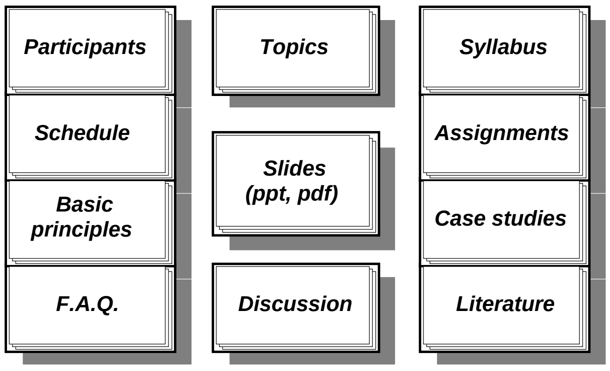


Fig. 1. Overview of the course materials and project organisation

The title slide of each topic covers project and management information: topic contents, project name, participants with their logos and the document version. As an example, see Figure 2.

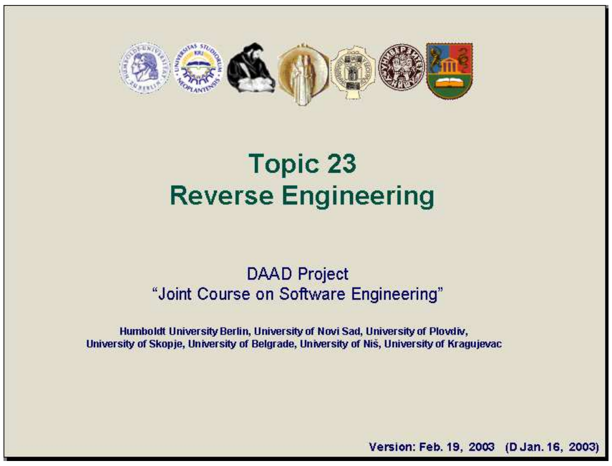
3 Technical Principles for a Distributed Course Development

1. Define the subject, the contents, the structure and the syllabus of the course

This is an important part of the requirements specification of each software project adapted to the development of e-learning material. In our project we oriented ourselves by the contents of acknowledged software engineering textbooks [1, 3] as well as by the CC2001 curriculum [2]. Division of our course into parts was done according to general software development phases.

2. Determine the prerequisites of the audience and the place of the course in the informatics curriculum

There is a big difference whether software engineering is tought in the beginning of a study or later in the curriculum. In our case we decided to offer software engineering



Topic 23
Reverse Engineering

DAAD Project
"Joint Course on Software Engineering"

Humboldt University Berlin, University of Novi Sad, University of Plovdiv,
University of Skopje, University of Belgrade, University of Niš, University of Kragujevac

Version: Feb. 19, 2003 (D Jan. 16, 2003)

Fig. 2. Example of a title slide

courses after students got familiar with programming, especially with object-oriented languages. Thus, our software engineering course should be offered to third- or fourth-year-students.

3. Be aware of the specific conditions concerning the offered basic curricula at the participating universities.

Software engineering has to deal with all aspects of software development. Therefore it should be analysed, for example, if the students are familiar with the concept of object-orientation or with coding style guides. On the other hand, complementary lectures in the existing curriculum[1] should be identified in order to decide whether the software engineering course should contain them or not. In any case, differences between universities can be solved by declaring certain parts of a lecture as optional, so that they can be left out when needed.

4. Determine the prerequisites of the proposed lecturers of the course

There are two kinds of users of the course: students and lecturers. In our project we even had to be aware of different kinds of lecturers. All of them are experts in general informatics. However, the degrees of familiarity with certain aspects of software engineering differ. Some of the lecturers even are experts in software development through having conducted larger industry projects.

5. Develop a course as a whole - not only lecture slides

Introducing a new course demands more than just the lecture part. In addition we need appropriate assignments, case studies, examination guides, and literature recommendations. Especially software engineering lectures can get boring, if they are just straight-forward and are not accompanied by interesting case studies and motivating

[1] e.g. lectures on UML, OOA/OOD, software testing etc.

exercises. These additional parts have been included in our course and can be found in Figure 1.

6. Reuse and adapt existing core material

Developing software is expensive - there is no exception with developing e-learning "software" like, for example, ppt-slides. It can easily take several person-months or even person-years. In our project we could start from a stock of about 1200 German ppt-slides which have been adapted and extended by the project participants.

7. Choose a suitable intermediate language for the course material

Partners from four countries are using in our project four different languages (see Figure 3). Although the actual lectures should be held in the respective native language, there is a need for an intermediate communicational language which should be, by default, the English language.

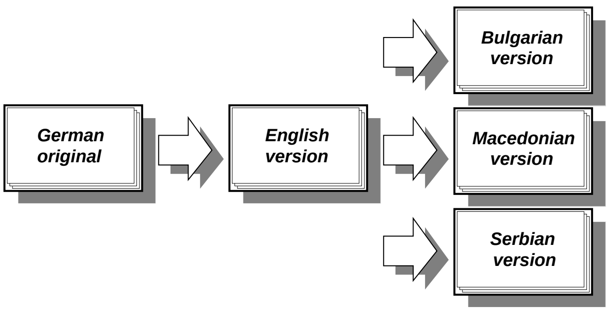


Fig. 3. Translation issues in our project

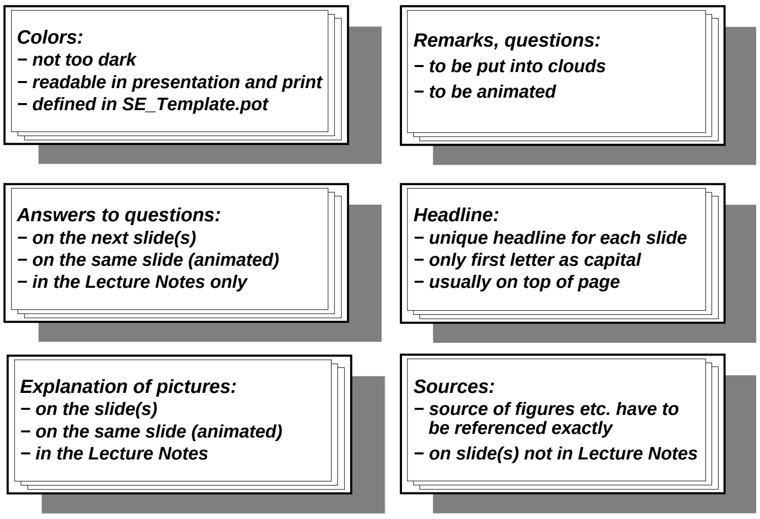
8. Define style guides for a unified appearance of the course material

In our project ten persons were involved in developing the course material. To insure a unified style there should be style guides for the slides appearance in the same way as there are coding style guides for programming tasks. In Figure 4 we present some of our project style guides. The unified appearance of the topic title slides (see Figure 2) is a result of them.

9. Provide lecture notes with additional structured information for the lecturer

As mentioned in point 4 of this section we have to take care of different preconditions of the lecturers. That is why it is a crucial point to deliver not only the pure slides but also additional information. This information covers the slide contents as well as methodological hints, e.g. on how to involve the students through questions to the audience. The lecture notes section of each title slide includes special management information.

Figure 5 gives an excerpt from lecture notes in tabular form. Note that the slides and the lecture notes are both part of the same ppt-document from which this table has



Colors:
– not too dark
– readable in presentation and print
– defined in SE_Template.pot

Remarks, questions:
– to be put into clouds
– to be animated

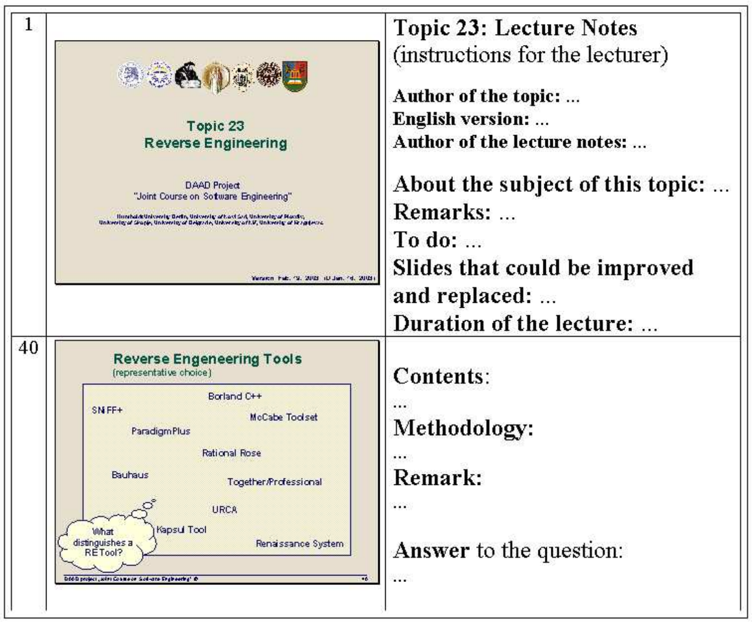
Answers to questions:
– on the next slide(s)
– on the same slide (animated)
– in the Lecture Notes only

Headline:
– unique headline for each slide
– only first letter as capital
– usually on top of page

Explanation of pictures:
– on the slide(s)
– on the same slide (animated)
– in the Lecture Notes

Sources:
– source of figures etc. have to be referenced exactly
– on slide(s) not in Lecture Notes

Fig. 4. Some of the style guides in our project



1	Topic 23 Reverse Engineering DAAD Project "Joint Course on Software Engineering"	**Topic 23: Lecture Notes** (instructions for the lecturer) **Author of the topic:** ... **English version:** ... **Author of the lecture notes:** ... **About the subject of this topic:** ... **Remarks:** ... **To do:** ... **Slides that could be improved and replaced:** ... **Duration of the lecture:** ...
40	Reverse Engeneering Tools (representative choice) Borland C++ SNiFF+ McCabe Toolset ParadigmPlus Rational Rose Bauhaus Together/Professional URCA Kapsul Tool Renaissance System What distinguishes a RE Tool?	**Contents**: ... **Methodology:** ... **Remark:** ... **Answer** to the question: ...

Fig. 5. Excerpt from lecture notes from one topic

been generated. It is of great help for overview and comprehension to have standard keywords to structure the lecture notes.

10. Support adaptability of the slides to different languages

A purely textual slide has two disadvantages in the framework of our project: The whole text has to be translated from the intermediate English to the respective native language and such slides - as a general rule of methodology - tend to be boring to the audience. That is why we generally recommend to prefer pictures with short textual explanations. Figure 6 shows an example of such a slide.

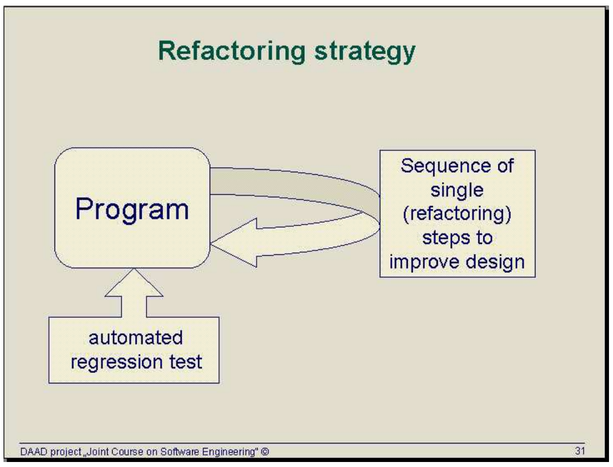

Refactoring strategy
Program
Sequence of single (refactoring) steps to improve design
automated regression test
DAAD project „Joint Course on Software Engineering" ©
31

Fig. 6. Example for using figures instead of long texts

4 Principles of Project Management for a Distributed Course Development

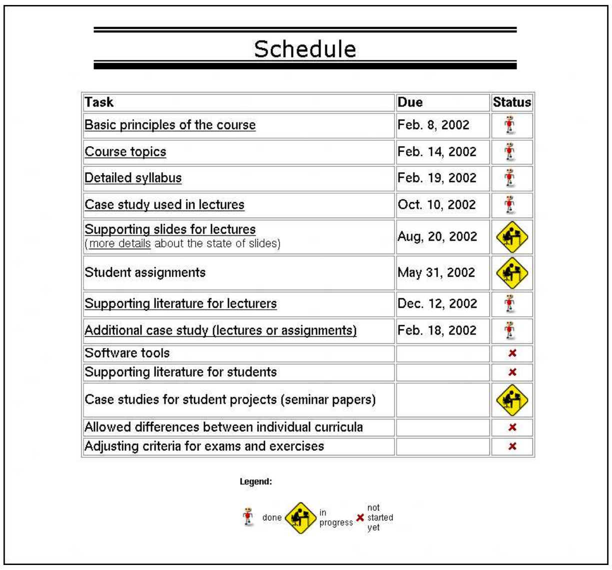
1. Define a schedule of tasks to be done

Developing a course as a whole (see point 5 of section 3) means to be aware of a couple of tasks to be done. These have to be fixed in a schedule. However, this does not necessarily mean to have fixed deadlines, too: In our case we received no additional man-power, so all the work had to be done besides the daily educational work of the participants at their universities.

Figure 7 gives details from the schedule in our project.

2. Organise a broad distribution of work

A lot of work had to be done in our project: translation of German slides into English, adaptation of these slides to the project needs, extention of topics contents, production of lecture notes, preparation of case studies etc.



Schedule

Task	Due	Status
Basic principles of the course	Feb. 8, 2002	✓
Course topics	Feb. 14, 2002	✓
Detailed syllabus	Feb. 19, 2002	✓
Case study used in lectures	Oct. 10, 2002	✓
Supporting slides for lectures (more details about the state of slides)	Aug, 20, 2002	◆
Student assignments	May 31, 2002	◆
Supporting literature for lecturers	Dec. 12, 2002	✓
Additional case study (lectures or assignments)	Feb. 18, 2002	✓
Software tools		✗
Supporting literature for students		✗
Case studies for student projects (seminar papers)		◆
Allowed differences between individual curricula		✗
Adjusting criteria for exams and exercises		✗

Legend:

✓ done ◆ in progress ✗ not started yet

Fig. 7. Excerpt from the project schedule

To cope with this long list of tasks and for a bigger identification of the participants with the project a broad distribution of work is needed. In our case ten colleagues from five universities had been concerned with the slides, another one with the physical management of the web-site and two students implemented a case study.

3. Recognize special interests and particular competence in the distribution of work

To distribute the work means to assign certain persons to particular subjects or topics of the course. Such an assignment is, of course, most successful, if the special interests and competences of each participant are taken care for.

For example, if someone is rather familiar with reverse engineering - possibly through work in industry projects - she or he should take the responsibility for providing slides and lecture notes for that topic.

4. Define roles: For each task roles need to be defined for the participating colleagues

Involving about ten colleagues means to take care of the coordination of their work and to define the competences for each of them. This means that there is no essential difference to ordinary software projects where persons are assigned to roles. In our case we distinguished the following most important roles:

– Project manager: technical and organisational coordination
– Web-site administrator: publication and update management of the documents
– Developer (of slides): translation, adaptation, improvement, extention and testing of slides for particular topics
– User: application of the developed materials in lectures, provision of feedback through reviews

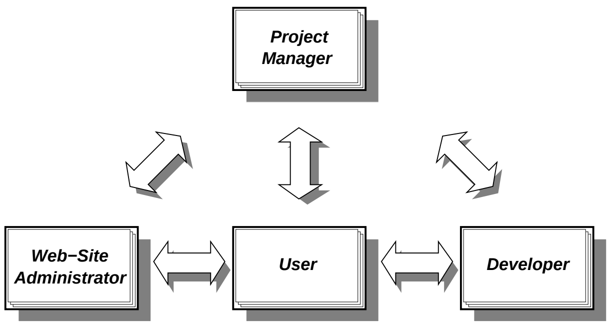


Fig. 8. Overview of roles in our project and their typical communication

Figure 8 provides an overview of the roles in our project together with their respective communication paths. Note that there is typically no direct communication between the developer and the web-site administrator. It is also worth mentioning that the communication between the web-site administrator and the user only considers the technical aspects of accessing the documents via the internet.

5. Organise a peer-review procedure for the course material developed in the project

Topics developed by a certain person need a review of other independent persons in the same way as reviews are ordinarily required validation and improvement procedures in each defined software development process. Of course, the best review will still come with the concrete preparation of the lecture by the actual users (the lecturers).

6. Organise a rigorous update and version management during the development phase

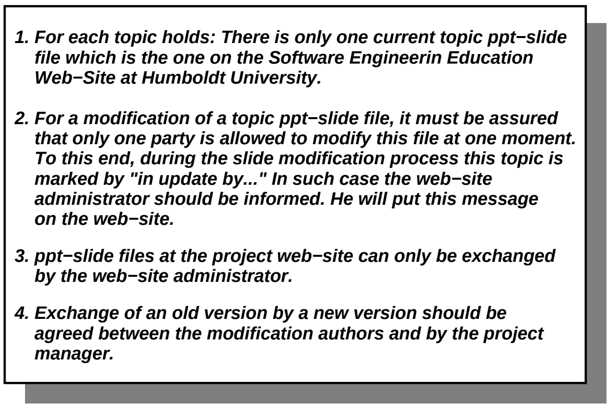
A distributed software project needs special care to keep the documents from inconsistency. In the development phase of our project, a simple but rigorous update management proved to be useful. The principles of this management are shown in Figure 9.

7. Recognize that there is a distinguished version management in the consolidation phase (maintenance)

During the development phase there is the particular situation that only one developer is responsible for one topic at the same time. In the consolidation phase the situation is different: Different universities will apply the course material and will come out with a couple of proposals how to improve or extend the material. To handle this situation a different approach to version management is necessary: The proposals have to be collected and discussed for some time and then from time to time a new version of a topic will be produced.

8. Try to organise an e-mail "hotline" and a discussion forum (mailing list) for lecturer's urgent questions during the "hot" lecture preparation phase

At least during the first time a lecturer prepares teaching the newly developed course she or he will have a lot of questions and remarks concerning the slide material.


1. For each topic holds: There is only one current topic ppt−slide file which is the one on the Software Engineerin Education Web−Site at Humboldt University.

2. For a modification of a topic ppt−slide file, it must be assured that only one party is allowed to modify this file at one moment. To this end, during the slide modification process this topic is marked by "in update by..." In such case the web−site administrator should be informed. He will put this message on the web−site.

3. ppt−slide files at the project web−site can only be exchanged by the web−site administrator.

4. Exchange of an old version by a new version should be agreed between the modification authors and by the project manager.

Fig. 9. Main rules for update/version management in development phase

To deal with this it proved useful in our experience to have an e-mail "hotline" between the lecturer and the developer of the slides.

9. Take care of an effective feedback of lecturers' experience with the course

This feedback should be organised by a structured questionaire. It should cover the following problem areas:

– Technical errors
– Slide presentation errors
– Problems with the lecture notes
– Proposals for topic improvement

10. Try to give the first lectures using English slides - however, speaking your native language

The first (English) version of the course will not be completely stable. Besides errors there may be the need to improve the style of the slides including the order of animations. To start too early with the translation into the respective native language may cause a lot of rework if the original English version changes. Therefore a good solution would be to use the English slides during the first lectures.

11. Last but not least: Organise workshops

This principle should not be the last step of project organisation. It is very important that the project participants get to know each other, discuss fundamental tasks and present results not only through e-mail.

An ideal event for such issues is a project workshop. In our project two workshops in Novi Sad (2001) and Plovdiv (2002) constituted the basis of work. The planned third workshop in Ohrid (2003) will summarize the work done so far.

5 Current Project State and Future Plans

By now, the first English version of the course material has been developed [4] and the first course at Novi Sad have been held. There was a lot of feedback from these lectures which led to the improvement and extention of the material. It was possible to present the first lecture in Novi Sad with English slides. Coordination was easily achieved through an e-mail "hotline".

The introduction of the course at other participating universities will lead to higher requirements concerning the management of questions and the maintenance of the teaching materials.

There are several plans concerning the improvement of the course contents: For example, the participants will introduce new case studies to the course, new topics will be included, particular topics will be extended and the like.

6 Acknowledgements

We acknowledge the support of DAAD (German Academic Exchange Service) under the auspices of the Stability Pact for South Eastern Europe through which the work reported here is being funded.

References

1. Balzert, H.: Lehrbuch der Softwaretechnik, Vol.1, 2nd Edition. Spektrum Akademischer Verlag, 2001.
2. Engel G, Roberts E, editors. The Joint Task Force on Computing Curricula, Computing Curricula 2001, Computer Science, IEEE Computer Society & Association for Computing Machinery, December 2001.
3. Sommerville, I.: Software Engineering, 6th Edition. Addison-Wesley, 2001.
4. Web-Site of "Joint Course on Software Engineering" HU Berlin. http://www.informatik.hu-berlin.de/swt/intkoop/see/.