

Software Engineering: A Common Playground for Computer Science and Surveying Students

Rudolf Müller^{1,3}, Andrew Frank¹ and Stefan Biffl²

¹Department of Geoinformation
Technical University of Vienna. Austria
Email: {mueller, frank}@geoinfo.tuwien.ac.at

²Department of Software Engineering
Technical University of Vienna. Austria
Email: Stefan.Biffl@tuwien.ac.at

³Spatial Information Research Centre
University of Otago, New Zealand

Presented at SIRC 99 – The 11th Annual Colloquium of the Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
December 13-15th 1999

ABSTRACT

Geographic Information Systems (GIS) has established itself as an increasingly important area in the field of information technology (IT). Many enterprises are now trying to include geo-related or spatial information into their systems or to build new GISs. This process is often hindered by the fact that GIS is a relatively new field of application for computer scientists and therefore most specialists in Software Engineering (SE) are not familiar with the special requirements of a GIS. This knowledge is often provided by surveyors, who are often not familiar with the principles of Software Engineering.

A number of courses for surveying and computer science students exist at numerous universities in which the students are taught facts which affect both fields of studies, e.g., data structures, spatial database and spatial analysis issues. We believe that these courses are not always sufficient to teach both parties the required knowledge needed for the successful completion of GIS projects, as these courses most often only teach facts and are not meant to teach students of different fields of study how to work together in a team. This shortcoming may lead to problems in real projects because the two parties are not familiar with the other's vocabulary, principles, techniques etc.

In this paper we propose a combined Software Engineering course for both surveying and computer science students, in which the students "experience" a real software engineering project while working in a team consisting of people with different expertise and background. This course will be based on the existing courses "Software Engineering 1 & 2" for computer science students at the Technical University of Vienna.

Keywords and phrases: Software Engineering, combined courses, Geographic Information Systems

1.0 INTRODUCTION

Geographic Information Systems (GISs) have become increasingly important within the area of information technology (IT). On the other hand, the "traditional" businesses of surveyors, like measuring, have become easier and easier, as more sophisticated and yet user-friendly instruments become available. The job profile for surveyors has been shifting more towards a "spatial data broker", that is, the surveyors have become more of an

expert in the "analysis of spatial data" rather than "data gathering". There is also a growing opportunity for surveyors to participate in the development of GIS as experts on spatial data [Frank, 1995]. Many universities have acknowledged this by incorporating courses on topics traditionally associated with computer science into the curriculum of surveying students [Curriculum TUV, 1999]. Such curricula include introductory courses on GIS, computer graphics, databases, networking and software engineering. Some of these topics, like introductory courses on GIS, are taught in interdisciplinary courses for surveying, computer science and information science students. Such a setup is not sufficient to teach the students from different fields of study the skills required to successfully implement GIS projects. These skills include project management, quality assurance and work organisation and are taught in courses on software engineering. Unfortunately many departments involved in educating surveyors do not have sufficient resources to administer a software-engineering course which includes the implementation of software development projects.

Software engineering, on the other hand, is an integral part of the curriculum of Computer and Information Scientists. Courses on this subject have been taught for at least several years at most universities. The expertise and experience gathered in these years is incorporated into existing courses. Therefore it seems to be advantageous to re-use these course setups for surveyors.

Surveyors are not supposed to do the work of computer scientists, such as implementation of projects, but they should be able to communicate with them and to understand their problems. Therefore we propose a combined software-engineering course for computer scientists and surveyors based on an existing course for computer scientists. In this course the surveying students will act as responsible persons for quality assurance and / or experts on spatial data within a software development project.

This paper is organised as follows: Section 2 summarises the software-engineering course for Computer Scientists. Section 3 outlines an existing course for surveying students. Section 4 describes a possible way to combine these two courses and Section 5 gives some conclusions.

2.0 SOFTWARE ENGINEERING FOR COMPUTER SCIENTISTS

This section gives an overview of the existing courses "Software Engineering 1 & 2" for Computer Scientists, as it is currently held at the Technical University of Vienna (TUV). It does not aim to give an exhaustive description of these courses, as this can be found in [Biffel and Grechenig, 1998]. This section gives a summary of the principles incorporated in these courses and describes their applications in a real course setup.

2.1 Organisational Background

The TUV provides a five-year computer science (CS) curriculum that aims at educating scientists and engineers. The courses "Software Engineering 1 & 2" are compulsory courses in the curriculum for Computer Scientists. Both courses consist of lectures (2 and 1 hours per week) and workshops (2 and 4 hours per week). The courses are recommended for students in their third and fourth semester. Each year 300 students complete the software engineering (SE) course.

A typical student is 20 years old, owns a fairly new computer, and can program in at least one structured programming language from the first year in computer science. The students have no experience with real projects or large systems and have never worked in a team of computer scientists before. Furthermore they have never written technical documentation nor do they have any work-experience with commercial relational databases.

Staff involved in this course are five professors together with 20 senior student supervisors guiding 50 teams of 5 to 7 people. During the course the professor takes the role of middle management, the supervisor acts as the quality assurance manager, and the team members work as project management and software engineers.

2.2 Goals

The goals of the workshops are to educate students to become fit for commercial software development projects. In a SE-course the following skills should be taught:

- The ability to understand new *SE methods and tools* supporting these methods within a fair time, e.g., four to six weeks.
- An above average familiarity with the *current major paradigms* of SE and information technology.
- The understanding of the whole *process of software engineering* from analysis to maintenance, including quality assurance and project management.

- The ability and modesty to *do any job within that process*, even if its outside reputation may not be high, like maintenance.
- The ability and modesty to *do any job within the team* by understanding their own role as well as the role of other team members while respecting their responsibilities.
- An understanding of and respect for the *value of actual experience* of senior software engineers and project managers, while seeing their own knowledge of state-of-the-art tools and methods in realistic context.

The last three requirements are personality factors, which cannot be taught in a lecture. They have to be taught by the actual experience of doing software projects and by experiencing the problems which arise while working in a team.

2.3 Structure

The complete course takes eight months and consists of three main parts, organised in two workshops, see Figure 1 for the timeline:

1. *Phase A* in October is a four-week entry "technical fitness" test, where the individual students have to show or acquire a minimum level of programming skills to be able to pass the course.
2. *Phase B* is a three-month team workshop from November to January, where the students form teams and complete a pre-designed product to get acquainted with teamwork, SE methods and tools. Part B is the most critical part of the workshop, since here individual students and groups acquire work patterns, which they apply for the rest of the workshop. The quality of these work patterns shape success or problem situations.
3. *Phase C* consists of a four-month team project from March to June, where each team has to find a suitable commercial task with a real customer outside the university and develop a complete system from specification to acceptance test. Part C repeats and strengthens the methods learned in part B and enforces active immersion into analysis and design.

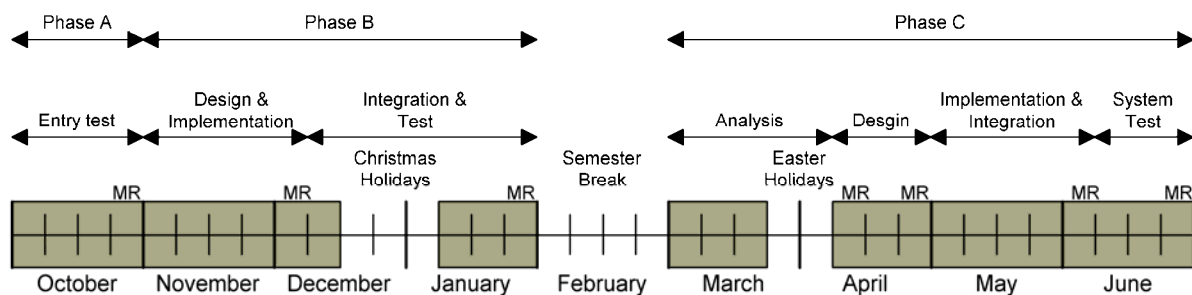


Figure 1: Timeline of course phases.

There are seven *management reviews* (MR) at the end of major phases where the teams present their products to a professor. Management reviews are useful checks of the teams' progress for the major project goals.

Several internal reviews (IR) prepare for each MR. Internal reviews are to trigger an early start for most work packages. Here the supervisor gives feedback to the team in the form of a discussion of the products submitted and tries to solve problems with minor rearrangements, like shifting work deadlines for up to a week. In addition the supervisor asks questions to determine the team members' understanding of the product and SE methods.

The final products of Phases B and C are administrative applications that typically help a group of users in a small office domain. Most of the systems developed in the workshop are given away to the customers and some 10 to 20 percent of the teams receive a nominal reward.

A weekly lecture, loosely coupled to the workshop, discusses the theory of methods and their use in industry. The supervisors show and discuss small examples in a string of tutorials in Phases A and B to help students start working on SE methods in time. Furthermore a lecture and workshop on quality assurance to encourage reflection on the events during the SE workshop is provided.

2.4 Contents

The software development phases structure the timeline of parts B and C and put the range of SE methods into perspective. This section gives an overview of the tasks the teams have to perform in Phases B and C and relates them to SE methods. Figure 2 shows an overview of the course content.

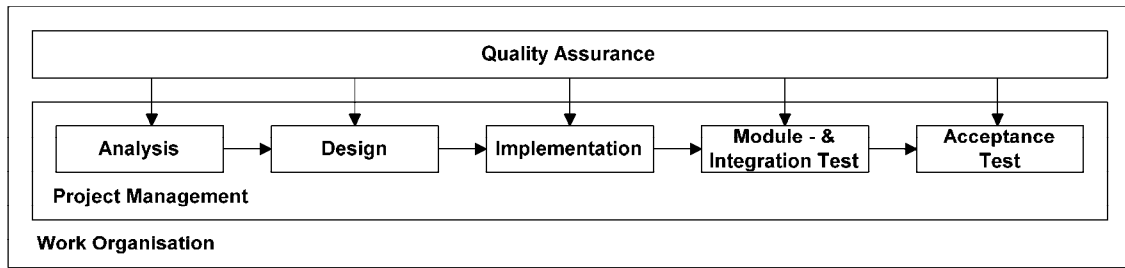


Figure 2: Course Content

The students are required to develop a sound analysis of the project. This analysis results in the specification of a software product. This specification is then used to design an application. After implementing the application, the students have to apply tests (module and integration tests) to ensure the quality of the application. The project is accepted after the product passes an acceptance test, which has been negotiated after the completion of the specification. Quality Assurance and Project Management are an integral part of the whole project in order to ensure the successful completion of a product of high quality. Work organisation of the team forms the background of all activities.

2.4.1 Phase A

In Phase A, which lasts for only four weeks, each student has to complete a simple database application. This application consists of 2 entities, a m:n relationship between these two entities, methods for the manipulation of the data and a few simple reports.

The students have to implement this application using an object-oriented environment (Borland Delphi) and a standard relational database (Informix). As most of them have very little or no experience at all with object-orientation or relational databases, they have to reach a certain standard in these subjects rather quickly in order to be able to contribute in a team. Software engineering methods learned in this phase are Extended Entity Relationship Diagrams (EERs) and Object-Oriented Analysis and Programming (OOA and OOP).

2.4.2 Phase B

In Phase B the teams have to complete a given framework with frozen interface specification. The information available to students are a system specification, design documents, such as interface specification, and a infrastructure, which implements some basic data access methods. The specified system consists of 15+ database entities and 14 relationships between these entities, including 6 m:n relationships. The students have to use the prepared infrastructure and have to stick to the specified user-interface. Their tasks are:

- *Analysis*: the teams have to assess the parts of the system that have been provided. Based on this assessment they have to decide what work has to be done.
- *Implementation*: the example-application can be split into several modules, which can be implemented by individual students.
- *Integration*: the single modules have to be integrated using specified interfaces. Some interfaces between modules have to be defined by the students.
- *Test Planning*: in an early stage of Phase B the students are required to produce adequate test plans for the modules, the different stages of the integration of these modules and the fully integrated application.
- *Quality Assurance*: during the whole project students are encouraged to review their documents before they are submitted to the supervisor. Not doing so might result in the delayed completion of tasks. A typical example for this are test plans, which are not exhaustive and therefore tests based on this plan have to be repeated, thus resulting in delayed completion, and worse grades are received.
- *Project Management*: the team has to organise itself. The students in a team have to negotiate among themselves who is doing what and when the tasks have to be completed. It is neither the supervisors' nor the professors' task to appoint students for a certain job or to monitor the timely completion of sub-tasks. Nevertheless they might suggest a new timeline in case a certain sub-task is delayed. The only strict deadline is the final presentation at the end of Phase B.
- *Work Organisation*: the students have to work in a team and interact with fellow students. Although this might seem trivial, bad communication and animosities between team members are the primary reasons why teams fail to complete the project.

In a regular review process the course participants are asked for feedback. Most students complained that the workshop "Software Engineering 1", i.e., Phases A and B, demanded significantly more work than they were credited for in their curriculum, while conceding that they would be able to use the experience in coming projects.

2.4.3 Phase C

In Phase C the students have to find a project of approximately the same size as the pre-defined project in Phase B (4 to 6 person-years) with a real customer. Their tasks are:

- *Problem Analysis*: in interviews with the prospective customer the students have to determine what they are required to do. This also involves learning another field of application's vocabulary, interviewing techniques and the ability to conceptualise the customer's requirements. Furthermore the students have to be confident in their chosen programming environment and to be able to assess the feasibility of the customer's wishes.
- *Specification*: each team has to develop a sound specification for the chosen application. This involves SE-methods, such as EER and OOA. The product of this sub-phase is a description of the functionality of the application.
- *Design*: during design the students have to specify the architecture of the application. This includes the database structure, the software modules, the user-interface and the interfaces between these components. The students have to use object-oriented design (OOD) methods to specify the architecture.
- *Test Plans*: as soon as the specification is accepted by the supervisor and the customer the teams have to generate test plans for the acceptance test. These test plans are the basis for the final acceptance of the product by the supervisor and the customer. Furthermore the teams have to write test plans for module and integration testing.
- *Quality Assurance, Project Management and Work Organisation*: as in Phase B the students organise their work independently. The supervisor reviews their documents and products and helps the students to overcome technical and organisational problems. Organisational problems often arise when one or more students depend on the timely completion of another sub-task and this sub-task is delayed.

Phase C is a direct extension of Phase B where the teams deepen their experiences with software engineering. They have to apply methods learned in Phase B independently and to interact with a "real" customer.

3.0 SOFTWARE ENGINEERING FOR SURVEYORS

There is a growing opportunity for surveyors to contribute to the development of software for geographic information systems (GISs) [Frank, 1995]. Job opportunities with the major manufacturers and vendors of general-purpose GIS platforms are available. Many smaller companies create and sell inexpensive and more specialised GIS software, thus creating yet another market. [Kemp and Frank, 1996] identify the skills of communicating spatial information, project management and implementing GISs in an organisation as important for a GIS curriculum.

These changes of the job profile of surveyors have been acknowledged in the curricula of surveying courses by many universities, see for example [Curriculum TUV, 1999]. The curriculum for surveying students specialising in geo-information identifies the design, implementation and maintenance of GISs as main areas. This is acknowledged by the following compulsory courses:

- Electronic Data Processing 1 &2
- Introduction to GIS
- Structured Programming
- Software Engineering
- Processing of Geometric Data

Furthermore the students may choose specialised courses, like database systems, computer graphics and expert systems.

Surveying is a 5-years degree at TUV. The course "Software Engineering for Surveyors" is recommended for students in their 7th semester. When taking the course most students are familiar with a structured programming language and have some knowledge about databases.

Obviously the goals for the course "Software Engineering for Surveyors" are the same as for the course "Software Engineering for Computer Scientists", as described in section 2.2 — they are universal for any course

on software engineering. The course for surveyors does not need to cover certain areas, like programming, in as much detail as it is required for computer scientist.

At present the course "Software Engineering for Surveyors" consists of two parts:

- A one-hour lecture, in which the students are presented papers on SE. The students form groups of two to four students and prepare a short summary of their paper. In each class three papers are presented to all students. These presentations are accompanied by discussions. The goal is to give students an overview of reasons for applying SE methods and different SE techniques.
- The lecture is supplemented by a one-hour workshop. In this workshop the students again form groups and implement a specification in the functional programming language *Haskell*. Functional programming languages have the advantage that they can be used to implement executable specifications. For a more detailed discussion, see, for example, [Frank and Medak, 1997] and [Car and Frank, 1997].

The goal of the workshop is to demonstrate that it is not a trivial task to write sound specifications — especially specifications, which have to be implemented by a person who is not familiar with GIS.

By the end of the course the students should have an understanding why software-engineering techniques are important for the implementation of commercial applications, such as GIS. They should also be aware of the difficulties of writing a sound specification.

4.0 COMBINING THE COURSES

The underlying idea of combining the two workshops on software engineering is to broaden the horizon of both computer science and surveying students. The main benefit for both parties will be that they have to work with experts of another field in a team in order to achieve a common goal.

4.1 Organisation

We propose to offer a course of lectures on SE for surveying students during fall term (October to January), where they are taught the principles of SE, such as software lifecycles, principles of the analysis and design of software products and testing. At the start of spring term (March to June) the surveying students will be introduced to Phase C of the existing courses "Software Engineering for Computer Scientist 2".

4.1.1 *The Role of Surveying Students*

Surveying students are credited one hour for this workshop, whereas the computer scientists are credited four hours. Surveying students are not supposed to do the programming — their job profile demands them to act as experts in GIS (data and functionality) or as customers for software development projects. Therefore we propose to employ one surveying student per team as responsible for quality assurance.

The surveying student's task will be to participate in the specification of the application, design and administer the acceptance test, and to review the other team members' documents internally before they are submitted to the supervisor. This does not mean that they are supposed to work as supervisors — it means they will be responsible for the quality assurance within the team.

The surveying student's most important task is the generation and application of (acceptance-) test plans. These test plans are developed together with the customer. In order to understand what they are about to test the surveying students should also be involved in the specification of the system, which is again developed together with the customer. This means that they will be part of the interface between the customer and the team.

Furthermore the surveying student will have to review the documents submitted by other team-members internally. These documents include, for example, the design documents, module and integration test plans and the help system. This usually involves discussions between the reviewer (surveying student) and the document's responsible person (computer scientist). Only when the documents have passed internal revision are they submitted to the supervisor. If the supervisor accepts the documents they might be sent to the customer.

See Figure 3 for a schematic outline of the surveying students' activities within a team. Tasks symbolised by grey areas are the sole responsibility of the surveying student. The computer science students are responsible for the tasks symbolised by white areas. Although these tasks are the responsibility of the computer science students, the surveying students have to participate, as they have to have certain knowledge about these tasks in order to assess the quality of the resulting products.

This figure shows that the surveying students are primarily concerned with Quality Assurance. Please note that this task cannot be performed by a single person alone, but has to be coordinated with the whole team. Nevertheless, the surveying student will be the supervisor's contact person in matters of quality assurance. The surveying student will also participate in the analysis as an "ordinary" member of the team. He is only concerned with the Design and Implementation as far as Quality Assurance is concerned. The acceptance test, on the other hand, is one of his main tasks, as he is responsible for the presentation of the product to the supervisor and the customer. Project Management and Work Organisation is approximately evenly distributed among all team members.

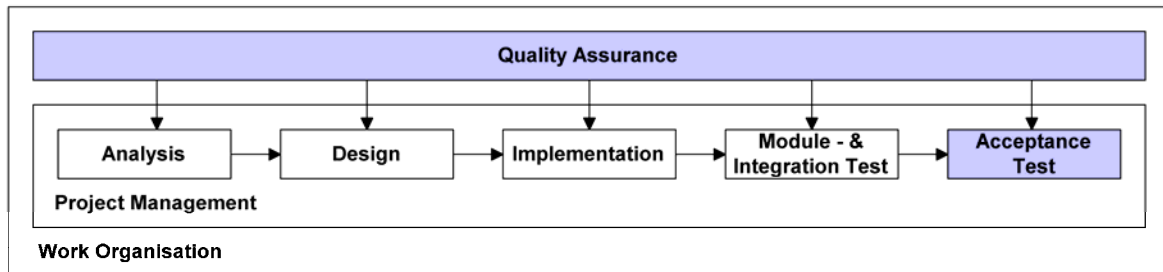


Figure 3: Activities of Surveying Students in Phase C

4.1.2 The Roles of Computer Science Students

The roles of the computer science students remain basically the same as in the current setup. They are still responsible for a sound analysis, design, implementation and module and integration test. The main difference to the currently employed setup is that often neglected aspects of a project, such as quality assurance and acceptance test will be carried through by a person dedicated to quality assurance — the surveying student.

In order to allow for successful quality assurance, all students in the team have to provide sufficient input to the quality assurance person. As it is often the case in real commercial projects, this person is not a computer scientist. Therefore the computer science students have to explain the involved SE-methods to the quality assurance person.

The computer science students also have to discuss their specification, design, test and help documents with the surveying student. It will be the responsibility of the supervisors to ensure that "real" discussions on these documents take place and that the opinion of the computer scientists is not simply imposed on the surveying students. This could be achieved by meta-discussions about this discussion between the supervisor and the team.

4.2 Projects

Much of the success of the projects in Phase C depends on which applications the students are about to build. In this section we consider two scenarios: first, the situation as it is now, where the students are required to find a project in the "real" world with a real customer. As this leads to some problems and does not take into account any GIS-related topics, we will outline another idea how to supply the students with GIS-projects. Nevertheless, both approaches could be implemented in the course to the benefit of both computer science and surveying students.

4.2.1 Present Situation

In the current setup the students are required to find a suitable project of approximately the same size as the project in Phase B. Typical projects are administration tools for small or medium sized businesses.

Although these projects are typically *not* GIS-related, it should be an interesting experience for surveying students to participate in such a "common" project. The field of application would be new for both parties (computer scientists and surveyors) and they have to learn the field's vocabulary and techniques together. As they come from different areas they will provide each other with different perspectives during the analysis-phase of the project. It should also make it clear to the students that software engineering is not "a rather adventurous enterprise for heroes", but rather a project where "a team meets and organises itself in order to engineer a useful service to a real customer" [Biffel and Grechenig, 1998].

Much of the success of the projects in Phase C depends on which applications the students are about to build and who acts as customer. The main reason for failing to complete a project within a semester is underestimation of the project by the students. It is the responsibility of the supervisor to ensure that the students choose appropriate

projects. This, however, is not always easy, as the supervisor is not in direct contact with the customer and therefore cannot estimate the customer's expectations.

Another reason for not completing a system is that some projects are found within the family of a team member, like building an application for a team member's uncle's shop. One problem with these projects is that the customer has often been persuaded to act as such and never really considered having a specially tailored software solution for his business. He is therefore not very interested in the project and usually never gets in contact with the team. The role of the customer is then often taken over by a team member ("the customer's nephew") who acts as de-facto customer. Such a situation is very difficult for the supervisor and the person responsible for quality assurance, as many details in the specification do not correspond to real problems or are just dismissed as "not important for the customer" by the de-facto customer.

Nevertheless, even participating in a non-GIS-related project may have some advantages for surveying students, as outlined in section 4.3.

4.2.2 GIS-related Projects

Another approach is to provide the students with medium-sized GIS-related projects. In this section we consider two possibilities:

- *Companies*: one approach is to contact GIS-vendors and ask them to provide medium-sized projects. The students will then carry these projects through. The advantage here is that we have real customers and real projects. The disadvantages are that there are not many GIS-vendors in Vienna and working with companies based further away might prove to be difficult. An open question is whether we can find companies willing to have rather inexperienced students working on their projects. Maintenance is also an issue here as many teams will not be willing to sign contracts obliging them with maintenance for several years. Another idea is to ask vendors for completed projects and to have the students re-design and implement them. This might be easier for the students, because they would have no obligations after finishing the project. On the other hand it poses more effort on the vendors, as they have to provide at least one "customer", who is familiar with the project. Nevertheless, vendors might be interested in such a solution, because it gives them the chance to come into contact with prospective surveyors and computer scientists on an informal basis.
- *Academic*: if it should prove too difficult to involve GIS-vendors the involved staff-members can offer the students projects related to their research. In such a case staff-members would act as customers. Again there would be a real customer, but the "reality" of the projects might be in doubt.

Offering the students GIS-related projects might have some advantages, as described in the following section, but it appears to be more difficult to offer the students appropriate projects.

If a GIS-related project can be provided the surveying students would play a more significant role in the specification of the system — they would be responsible for the correctness of the GIS-related parts of this system, like projections, quality and presentation of the data. The surveying students would act as experts of the field of application.

4.3 Anticipated Effects

This section gives some anticipated effects of the proposed organisation of the workshop. This list is not exhaustive, as many other issues might arise as soon as the setup is applied. The most likely anticipated effects for the students are:

- *Experiencing software engineering*: the main task of surveying students in this workshop is to write test plans for the acceptance test and to administer this test. By becoming involved in testing and understanding the test results the students will learn that software rarely works as planned. Many small mistakes can be found even during acceptance testing. By being involved in the whole process of implementing the software product the students should experience that it is *very* difficult to produce error-free products. This fact is often repeated in software engineering lectures, but students often tend to believe that they are immune to committing these mistakes themselves. It is a valuable experience to see that even carefully designed and implemented projects are not safe from being faulty.
- *Exchange of knowledge*: the surveying student comes from another field of expertise and uses a different vocabulary and is not as familiar with the principles of SE as the computer science students, who have already mastered a semester working in a team. Although the surveying students should be familiar with concepts of SE from the lecture in fall term, they lack the experience and the lessons learned in Phase B of the workshop. This knowledge has to be passed on within each team, requiring the computer science students to reflect on what they did during Phase B and to explain it to the surveying student in a comprehensible way. This involves a critical review of the techniques used in Phase B. An internal review

process should help computer science students understand why they are using these techniques and what advantages and drawbacks they have. It is often the case that the students use SE-techniques because they are told to without reflecting on the underlying principles. Being more aware of the implications of using techniques like EER, testing and reviewing, it will be easier for the students to adapt these techniques to be used in other projects and even other courses. Furthermore, a discussion of software engineering methods between computer science and surveying students might stimulate deeper understanding of these methods in both parties. The supervisor should encourage such discussions.

- *Clash of ideas*: surveying and computer science students will have to work together to build a sound specification. This is a new task for both parties. In this process they have to get acquainted with a new field of application, depending on their project. Discussions during the work on the specification are expected to give both parties new ideas and perspectives.
- *Improved quality assurance*: in the current setup quality assurance is largely the responsibility of the supervisor, who reviews the documents and tries to maintain a certain level of quality in these documents. With a team member as the responsible person for maintaining good quality and organising internal reviews within the teams, the quality of the submitted documents are expected to increase. By reading not only his or her own (test-) documents the surveying student should gain further understanding of how computer scientists think and work. The computer science students have to adapt the style and vocabulary of their documents in such a way that they do not use too many technical terms. Experience shows that this is very difficult for the students. Another advantage for having students as responsible persons for quality assurance is that they are part of the team and have the same goal, i.e., completion of the project and good grades. Senior students acting as supervisors and Quality Managers, on the other hand, are often perceived as not being part of the team by the students. Therefore changes to documents are not discussed on a peer-to-peer basis, but are often perceived as imposed upon the students by a "higher authority".
- *Changing social structures*: teams of computer science students formed at the start of Phase B usually stay together for the rest of the whole course. After mastering a steep learning curve during this phase they often have not only become a team working on a project, but are also working on other courses together — they have formed a social structure in which most of the students feel comfortable. By the introduction of a "stranger" into the team this social structure has to be modified. Roles assumed in Phase B, such as team co-ordinator, tester and programmer, have to be re-assigned. This process is typical for "real" projects, where team members come and go in a matter of months.

The anticipated effects given so far only take into account a scenario where the students work on a general software engineering project. If it is possible to provide the students with GIS-related projects the following effects should be observed as well:

- *Being an Expert*: If the students work on a GIS-related project the surveying students would also act as experts. This means that, as with the computer science students, they have to develop the skill to explain their knowledge to students from another field of study. The surveying students also have to communicate their knowledge of GIS-related fields, like cadastre, maps, data quality, projections, and so on, in a form comprehensible to computer science students. They have to explain the concepts and techniques, to be used in the SE-project in great enough detail for the computer science students to be able to design and implement such a system. On the other hand, it is not always necessary to explain a certain problem in very great detail — the surveying students have to find a balance between detailed descriptions and more abstract outlines of the problems at hand. This balance may shift from more general explanations of problems to increasingly detailed descriptions as work on the project proceeds.
- *Working with an Expert*: It is not always easy to pay heed to an expert's advice — especially not for young computer science students. Being in a situation where they have to, i.e., working on a GIS-related project, will help them to learn how to work with an expert. It is not always easy to conceptualise new ideas and to admit that one might not have fully understood the presented concepts.

Experience indicates that acting as an expert and listening to an expert might prove challenging for both parties.

It might not be always possible to actually observe all these effects — it might happen that the surveying students' opinions are simply ignored by teams of computer scientists. To avoid this problem, the supervisors have to closely monitor how the teams proceed. The best way to do this might be to ask the students for a report on *how* they achieved their goals in every internal review.

It also seems desirable to have one supervisor dedicated to attend to the surveying students only. This supervisor could communicate problems the students have in their teams to the other supervisors in order to resolve conflicts at an early stage. The concerned supervisors could then influence their teams to make the whole experience more beneficial for all participating students. A further extension could be to organise meetings for

the surveying students on a monthly basis where they have the chance to discuss their problems and experiences with their fellow surveying students under the control of the dedicated supervisor.

5.0 CONCLUSIONS

In this paper we proposed a combined workshop on software engineering for computer science and surveying students. The setup of this course is based on the idea to introduce surveying students as responsible persons for quality assurance into software engineering projects.

The anticipated advantages of a combined Software Engineering course for surveying students are a profound understanding of the process of software engineering through personal experience and experiencing working in a team with students of another course of study. Computer Science students should become more aware of the perception people of other professions have about their profession and how to deal with it. This may help computer science students during their professional career when they often have to work in teams with non-computer scientists.

Two kinds of projects for the students to work on have been proposed: "standard" SE-projects, as they are now used in the course "Software Engineering 2 for Computer Scientists" and GIS-related projects. Standard projects would allow the surveying students to work as quality assurance managers and may give computer science students the opportunity to experience another point of view on issues such as specification, testing and project management and work organisation.

GIS-related projects might be very advantageous for the students to work on, as they allow the surveying students to extend their role as quality assurance manager to that of an expert in the project. On the other hand, such projects might be hard to come by.

BIBLIOGRAPHY

[Biffel and Grechenig, 1998] Biffel, Stefan and Grechenig, Thomas, "Preparing Students for Industrial Teamwork: A Seasoned Software Engineering Curriculum, Experiences, Requirements, and Guidelines", IEE Proceedings on Software Engineering, 1998.

[Car and Frank, 1997] Car, Adrijana and Frank, Andrew U., "Formalisierung konzeptioneller Modelle für GIS - Methode und Werkzeug", Zeitschrift für Vermessungswesen, 1997, 122 (3), pp. 99-114.

[Curriculum TUV, 1999] "Studienplan der Studienrichtung Vermessung und Geoinformation", Technical University of Vienna, 1999 .

[Frank, 1995] Frank, Andrew U., "Surveying Education for the Future", Geomatica, 1995, 49 (3), pp. 273-282.

[Frank and Medak, 1997]. Frank, Andrew U. and Damir Medak, "Executable Aximomatic Specification Using Functional Language - Case Study: Base Ontology for a Spatio-Temporal Database", EJCIMKB'97, 1997.

[Kemp and Frank, 1996] Kemp, Karen K. and Frank, Andrew U., "Toward consensus on a European GIS curriculum: The International Post-Graduate Course on GIS.", International Journal of Geographical Information Systems, 1996, 10(4), pp. 477-497.