Commitment as an Underlying Principle for Learning

Conny Johansson and Kari Rönkkö

Blekinge Institute of Technology, School of Engineering
Soft Center, SE-372 25 Ronneby, Sweden
Conny.Johansson@bth.se Kari.Ronkko@bth.se

Abstract. The prevailing model of software development on which most educational programs are based is in conflict with general practice in industry. As following paper demonstrates Software Engineering education lacks an approach to teach the skills needed to master coalitions of existing recourses that are hard to control. In order to prepare students to handle unpredictable, non-technical and moving targets, an adequate curriculum is needed. Since 1990 software engineering education at Blekinge Institute of Technology has used commitment as the underlying principle for learning. This principle has made it possible to address the discrepancies between education and industry. This paper describes and evaluates our experience of using six elements of commitment in the education of software engineers.

1. Introduction

This paper addresses the criticism that most educational software engineering programs are in conflict with accepted industrial practice [1] [2]. For more than a decade the software engineering program at Blekinge Institute of Technology has integrated the elements of unpredictability and changeability in industry into its educational program. In this paper we present our experiences and an evaluation of our success rate.

Responsibility and initiative are personal qualities that students are expected to have when taking up a position in industry; they are expected to be able to handle an unpredictable, changeable and messy reality. The prevailing model of software development used in most educational programs is the closed-shop model [1]; this model does not comply with general industrial praxis. Important skills needed to handle emerging requirements, complex economic constraints, end-users, and coalitions of existing resources that are not under control, are not taught in the form of software engineering education which prevails today. Shaw claims that educational institutions must prepare software engineers to handle these forces in the development process [1]. In order to control these often unpredictable, non-technical 'moving targets', complements to the prevailing educational approach are needed. The educational community itself is increasingly moving from the formal lecture to projects and problem solving, allowing students to exercise lessons learned [1]. See reference [3] for a study of learning environments and different approaches to courses that emphasize practical work through projects.

Several educational engineering programs are built on the assumption that knowledge can be effectively transferred from one person to another. This attempt to transfer knowledge frequently occurs in classrooms under well-defined processes, where the teacher has the responsibility to plan in detail what knowledge the student's brains should be fed with [1] [3]. Knowledge is divided into several subjects and the student's progress from one subject to another specializing with clearly defined and separate areas. This might be a successful way to present an overview of the great variety of classical structures used in software engineering, such as different kinds of techniques, plans, models, estimates, reviews etc. But such an assumption does not do justice to the fact that, if subjects are presented as divided and stepwise the resulting body of knowledge also includes its own presentation. Hence, students are forced to use much of their learning time to figure out what the teacher wants to achieve in his/her lectures. The student's problem solving, focus and power in a closely planned learning situation are directed at remembering a line of argumentation that can be reproduced later at an examination. This form of teaching excludes important skills that are required by industry today [2]. The most effective learning occurs when taking action, i.e. when experiencing something. This has been demonstrated to be six to seven times more effective than lecturing [4]. Sveiby has presented a different criticism; he claims that the most common method for transferring knowledge, the lecture, is the least effective [4]. Five days after a lecture most people remember less than a tenth of what they heard. Furthermore, people still learn mainly from following examples and by practicing [4]. In this and the former perspective, the educational method chosen provides a poor learning environment, i.e. the environment does not invite students to apply their own experiences to solve problematical situations in the way professionals do. There is still a lack of knowledge about the different forms and potential of open-shop development model courses [1] [3], and there are few papers which discuss how these should be taught. Having a learning situation organized in the way described above does not stimulate students' ability to take responsibility and initiative (We regret that some of our basic technical courses are still too much organized in this way).

This paper presents our experiences of using the commitment approach based on the commitment discipline concept defined by Humphrey [5]; this is one approach that is able to respond to the forces in an open-shop development model. We have adopted the principle of learning-by-doing [6], where students learn from their own actions and use the theory to repair mistakes and solve problems that occur during their education. This action-oriented learning is studied in [3], and is supported by Argyris [7] and Kolb [8] among others. Project courses with careful and diligent supervision are the tool we use in our undergraduate software engineering program [9]. In our educational approach, commitment is the guiding light throughout the educational program. Being the basic attitude and driving force in everyday work, commitment is something that stimulates creativity by its emphasis on active and voluntary responsibility and the implicit need for each student to take the initiative and to be self-propelled. In this paper we present a first evaluation among students who have graduated from our program.

In section 2, commitment will be explained in relation to our educational program; in section 3 it will be related on a detailed level to our experiences. Section 4 provides more guidelines for teaching commitment, while section 5 gives instructions on how to grade commitment. Section 6 presents an evaluation of our model. Finally, a conclusion is given in section 7.

2. The overall educational approach

Since 1990 commitment has been the underlying principle of learning in software engineering education at Blekinge Institute of Technology. It forms the basis of an undergraduate education program leading to a Bachelor's degree. The core of the program consists of three project courses which together comprise nearly one year's full-time studies. First, a project course that provides the basis for individual software development must be passed. Second, a small team project course including four or five students is carried out. Third, a large software engineering project, involving 15 to 20 students organized into several sub-teams, is completed. In addition, students take courses within the subjects of computer science, human work science and business administration, (see details in [9]). Each learning situation within the courses requires its own commitment approach where commitments of different kinds are acquired:

1. Basic pre-requisites. A prerequisite for a commitment culture is that students are given the possibility to take responsibility for their own actions. The basis for this is that during the first year students must have earned self-confidence and self-respect. Students must feel early on in their education that their choice of profession is important, demanding and challenging. They must feel that they have deserved a position of credibility in which they can be treated with respect. This is accomplished by having demanding courses from the very beginning, where students earn acknowledgement by passing them.

2. Individual project course. The individual project course has a budget of 200 hours per student. The lead-time is eight to ten weeks and the goals are to introduce commitment culture, negotiations and contracts, and to make the student familiar with the software development process. One teacher supervises each project and acts as an adviser to the student. The first step in the commitment process is to enable students to feel responsibility and obligation towards their client. A prerequisite for this step is that clients treat students with respect; this can only come about if students have built up self-esteem, as described in the preceding section. The client contact is the first situation in which students' professionalism as software engineers is challenged; they must therefore have the will to be professional. This will follows with their self-esteem and the respect that they will receive from the client. It is important to ensure that the proper prerequisites exist for the successful completion of both the tasks at hand and the overall mission. The trade-off between what is technically possible and what the specific student is capable of accomplishing is a balancing act which requires honesty and respect [10].

3. Small team project course. The small team project course has a budget of 280 hours per student and a lead-time of 18 to 20 weeks. The key software engineering topics in this course are requirements specification, estimation, project planning, progress tracking and system testing. In addition to the first commitment step students consider the difficulty of sharing commitment within a smaller development team. Students must treat each other with respect, and give a fair hearing to other students' opinions and concerns (see for example [10]). They are encouraged to handle the difficulties of agreeing, communicating and organizing, and to continuously reorganize the shared picture of understanding. These commitment qualities must be trained individually, within the team and with respect to the client.

4. Large team project course. The large project has a budget of approximately 700 hours per student and a lead-time of 18 to 20 weeks. The key content is related to the complexity of large-scale product development and large organizations. Pre-study work, including understanding the business goals of the product and evaluation of different alternatives for technical solutions, is one of the focuses of this course. Quality assurance techniques such as reviews, inspections and audits are also important topics. The third step in the commitment process that students are expected to learn is the difficulty of sharing commitments between sub-teams, as well as showing respect to other sub-teams. In order to handle the difficulties of agreeing, communicating and organizing they should ask themselves: "What have we committed ourselves to as an entirety?", "What does that mean to each sub-part?", and "How is this transformed to commitments for each individual?"

To sum up the commitment culture: Commitment is about continuously renegotiating and reorganizing the true shared picture of understanding, both individually, within the team, between sub-teams, to the project as a whole, and towards the client. We believe this model reduces the knowledge gap, i.e. the importance of a topic and the knowledge recently received that students usually have when finishing their software engineering education. Project courses today are offered by several computer science programs [11], and are supported by studies made in [3] [12] [13] [14] and [15] among others. Furthermore, we clearly address topics such as negotiations, management and ethics, the most significant gaps in knowledge according to [16]. We also believe that realism significantly increases motivation. If we can provide students with a realistic environment that corresponds well with reality, we will increase each student's performance. A close co-operation with industry is necessary to fulfill this wish (this is also emphasized by Beckman et al. [2] and Bagert et al. [17] among others). In this way we also reduce, as Shaw describes [1], the discrepancy between academic education and existing industrial practice. We do, however, deviate from reality to some extent, e.g. salaries to students are excluded and maintenance after product delivery is left to the client. In the commitment model projects are run as games where teachers' act as heads of department and representatives from industry act as clients. Design mentors, quality managers and specialist consultants are other examples of roles (see [9] for more details).

3. The elements of a commitment culture

Humphrey argues that the foundation for software project management is the commitment discipline [18]. Commitment discipline must be supported by plans, estimates, reviews and tracking systems. It is committed people who meet commitments and not procedures and tools alone [5]. Commitment is supported by formal and informal processes and structures. Being an obvious part of company culture, commitment is the basic attitude and driving force in everyday work. It stimulates creativity by its emphasis on active and voluntary responsibility and the implicit need for each employee to take initiative. The overall mission for a commitment culture is: 'Keep your promises'. In short, in a commitment culture each individual actively seeks new commitments, agrees on these, and does his utmost to fulfill them. We have chosen to use the concept of commitment as the underlying principle for learning as it gives the students a realistic, professional and challenging approach to their future profession. We have identified six elements of the commitment culture which pervade the whole education program. The elements are presented below in separate sub-chapters.

3.1. Voluntariness

Unlike in an enjoinment culture, (An enjoinment culture is characterized by that people, primarily managers, are ordering and commanding others on what to do), each commitment must be voluntarily accepted [5]. Our projects start by giving each team, (or each student in the individual project), a tender which includes the initial client requirements for a new software product. The task of the teams is, on the basis of their own initiatives, to act upon these tenders and present proposals for changes in order to adapt the product development to the predetermined time and cost limit. It is our belief that every tender will be changed to some extent before the contract is signed. If the tender is not changed in the beginning, it will be changed later. If it is accepted without further negotiations, the clients are recommended to be skeptical and sometimes they even have to encourage negotiations. Since students in general have little experience of project work, they tend initially to commit themselves to doing more than they are capable of. Software engineering projects are not easily estimated, planned and tracked by experienced people; it is thus very difficult for inexperienced students to make realistic estimations.

We appoint a teacher from the university to continuously supervise the projects. In our role-play, students are employed at a software house where the supervisor is the Head of Department (HoD). The supervisor supports his/her students and the teams to ensure that the final product is delivered on time, and without deviation from the agreed functionality and characteristics. In a professional situation, however, the HoD also has the additional responsibility to ensure that the project keeps within the agreed cost budgets. Over-commitment [18] is common in the first project course, and is responsible for exceeding the budget. Since a real software project should keep within the agreed budget, it is essential that our projects do the same. The tool the HoD uses is to take samples from planned progress vs. actual progress for commitments throughout the project.

Under-commitment in our context occurs when students deliberately take on project activities and allocate an unreasonable amount of time to be used as a buffer. It is necessary that the supervisor is aware of the problem of under-commitment since students very quickly acquire skills in negotiating. We believe that identifying over-commitments and under-commitments is a matter of experience on the supervisor's part. Our advice is to consider both, even though the problem of over-commitment is more common in the early courses and under-commitment in the later courses. Concrete examples of possible signs of under-commitment which have appeared in previous projects include single activities with a large amount of allocated time, e.g. more than 40 hours, and over-estimations of the time required for meetings and other unspecified activities.

3.2. Agreement

A formal contract is signed between the supplier and the client in a professional agreement. The contract specifies the framework for the assignment including, for example, cost and work limits, criteria for acceptance and fines when applicable. Fictitious contracts with no legal validity are used in our education. The contract is, however, one important criterion for assessing whether or not the students have succeeded in their projects. Furthermore, the contract forms the basis of subsequent negotiations during the project and is the basic criterion for passing the project course. We have noted the importance of frequent supervision throughout the project to ensure fulfillment of the contract. The HoD must have a long-term focus since students generally have a short-term focus. The chance of identifying

when renegotiations are necessary increases significantly if contract fulfillment is discussed frequently. A signed contract usually has high priority at the beginning of each project. The supplier has, for example, a vested interest in being able to charge the client as soon as possible. In general, people do not usually want to live in uncertainty; they want to know the requirements for the product being developed. The commitment perspective specifies that each project must be given the necessary time for preparation in order to be able to make commitments [5] [18]. What we need to strive for is to have the contract signed neither too early nor too late. The guideline for contract signing is that the contract should be signed before 15% of the project time has passed.

3.3. Infrastructure

The commitment infrastructure defines the requirements, estimations and plans in the project. The contract refers to this infrastructure, which consists of requirements specifications, project estimation sheets and project plans. In the individual project the focus on plans and estimations is rather small because of the short lead-time and limited budget; also, other issues (such as commitment to the client) are more important in the early stage of learning and practicing the commitment culture. We do, however, encourage and recommend that students plan and estimate, and we also provide them with some basic literature on the subject. We estimate that these recommendations are used by less than 10 percent of the students attending the first project course due to students' exaggerated anxiety about not being able to solve the technical problems in the project. After concluding the individual project, most students find that their work has been too ad hoc. They recall the recommendations regarding planning and estimations and become at this stage more motivated to apply it in their future work. For teachers it becomes less difficult to request plans and estimations later on. We have discovered that it is advantageous not to require formal planning and estimation documents. Despite the numerous tools available on the market, we have not found any tool that is flexible enough for keeping plans, estimates and progress status up-to-date. Our experience is that the tool problem is the main reason for not having updated plans and estimates. After numerous cases where plans and estimates were frequently outdated, we decided to recommend that students work informally. By using flipcharts, whiteboards, stickers etc., on the wall in the project room, we have improved planning and estimating. The method is simple and increases visibility making it easier to keep plans updated; it provides a good basis for the process of building up and keeping commitments both within and between sub-teams. We hold supervisory meetings in the project rooms; students and teachers stand up and discuss actual plans and estimates. The aim is to teach the students to master "Who does what and when will it be done". This model for planning, estimating and progress tracking is suggestive of the models used by agile methodologies such as Extreme Programming [19] and Scrum [20].

3.4. Openness

A commitment must be openly discussed and publicly stated [5] [18]. The challenge is to create an environment, a culture, where students feel that it is possible to discuss problems openly with other team members and the teachers without failing the course. It is important to note that we have put fulfillment of the commitments as one of the criteria for passing the course. A situation where the student fails to fulfill the commitment but has reported it at an early stage may lead to a pass if the hypothesis about how to succeed with the problem is reasonable. The student will fail, however, if he/she does not report problems. This is a common situation. It is thus important to remind students to report problems throughout the

project. Openness is best created in a relaxed, informal and light-hearted environment. Sometimes it is necessary to drop the teacher/supervisory role and talk with students as friends. This method should not be over used though it is the simplest method of achieving mutual trust between students and teacher. If trust is achieved it is more probable that problems will be reported.

3.5. Fulfillment

The next element of the commitment culture concerns making those responsible more inclined to meet commitments. According to Humphrey, the person responsible tries to meet the commitment even when help is required [5]. Being responsible for something implies that it is you yourself who must take the initiative to seek out and validate new alternative solutions. With commitment as the underlying principle for learning, we have created an environment in which it is natural to do one's utmost to fulfill commitments and help others to fulfill theirs. Students sometimes ask teachers for help, but more often than not this is received from other students, tool suppliers or discussion forums. In addition, we must stress that the majority of support for solving problems and keeping commitments seems to come from oral discussions between two or more people. As a teacher it is important that you capitalize on opportunities to initialize such discussions as a means of speeding up the problem-solving process. An interesting issue is the relation between credibility and commitment [22]. As software engineers you start with a disadvantage since software companies frequently fail to meet delivery deadlines. When starting a project we do not always have credibility; in order to get it, we must earn it. The only way we can earn credibility is to consistently meet commitments [22].

3.6. Renegotiating

Every project is renegotiated to some degree after the contract has been signed, as the world is constantly changing and more knowledge about details is acquired as time goes by. We fully agree with the statement that we need flexibility to respond and adapt quickly to change, see, for example, agile software development [20] [21]. If the requirements specification that is the basis of the contract with the client is at a suitable abstraction level, it is unnecessary to sign a new contract every time a detail in the agreement is changed. Whether or not the commitments are fulfilled in such cases is always a matter of judgment, but details such as change of formats, error handling and layout, for example, should in most cases be handled informally. The teacher, the supervisor, wants continuous evidence that the commitments can be kept. Students are thus trained in practical and real progress tracking in order to provide evidence of good commitment status. A real professional will, with the help of simple progress tracking, identify needs for renegotiations, and provide good background material such as estimations that confirm the need for renegotiation. Major renegotiations must be made in good time, (this is also a question of judgment), before the final delivery, otherwise the team or the individual will fail the course. Renegotiations less than two weeks before delivery are clearly unacceptable.

4. Teaching commitment – balancing with enjoining

High motivation reinforces the commitment culture. Motivation increases with continuous use of professional language in student meetings. The most obvious example is to use the word 'supplier' instead of the word 'student'. Furthermore, student motivation is closely

related to the ability to take initiative. If, as teachers, we have confidence in our students' ability to handle difficult situations on their own, they mature quickly and their self-confidence increases significantly. Examples of such practical attainments are client negotiations, the ability to find alternative solutions on their own, and tool supplier contacts. As a consequence of the need for increasing students' ability to take initiative, we must beware of how teachers approach students' questions. The basic principle guiding our projects is that we will not present an answer to a question, at least not before the students have presented some possible answers. The following questions are useful: "What do you propose for…?", "What alternatives are there when…?", "How does it work when…?", "If the deadline is May 17, when do you need to…?", "Why do you…?" etc.

It is easy to adopt an authoritarian attitude since as teachers we have the responsibility of judging the students' fulfillment of the criteria for passing the project course. We advocate that the supervisor takes a supporting role because it is important that students feel that the supervisor is on their side and not against them. The supervisor's role should be discussed continuously and confirmed by a handshake. It is also necessary to consider what kind of criticism you as a teacher communicate to a student. Sometimes, for example, it could be bad timing to offer criticism of suggestions if the team/student is much occupied with solving core problems. On the contrary, it might be useful to focus on the 'good' parts. Encouragement could urge students to solve the hard parts of a problem themselves. Students focus more often on commitments that are short-term and close in time. The teacher should focus his/her supervision on long, complex commitments with a long lead-time. You need early focus on the final delivery date and continuous focus on contract fulfillment. Sometimes the teams try to hide or delay commitments that seem hard to fulfill.

5. Grading commitment

The problems in grading group projects have been widely discussed; they seem to be a universal problem. There are few articles on the topic, but Huffman Hayes et al. [23] provides a good summary of the relevant work in their paper describing one approach to grading individual effort. We have faced a lot of difficulties in trying to make an objective assessment of whether or not each individual fulfills the set criteria for passing the project course. We have come to the conclusion that grading must be based on a student's performance in fulfilling his/her commitments, team as well as individual. The grading system we use is divided into two parts in the team projects: for the team as a whole, and for each individual. The basic criterion for passing is that all commitments must be fulfilled. All team members, and not only the project manager, are responsible for the success of a project from an educational point of view; all students have a responsibility for the success of the team as a whole. The foundations for approval are: - The team/individual must openly report difficulties in fulfilling commitments; - The team/individual must demonstrate the ability to negotiate and renegotiate; - The commitment agreed upon in the last re-negotiation must be fulfilled.

It is not, of course, possible to remove all the requirements by negotiation. The client and the supervisor must abide by specific minimum requirements in terms of content and difficulties; these must not be violated. Even though it is more frequent that we fail individuals than teams, we have had some cases where the entire team failed. (Note: Around 80% of the students who enter the individual project course pass the course). Approximately five students have failed the team project courses in the past few years. We must also stress

the importance of thorough checking of the qualifications for the team project courses. We have made the qualification requirements more precise with the result that we have few competence problems now.

6. Evaluation

We have carried out an evaluation among students who have graduated from our program, focusing on student assessment of our success with teaching the six elements of a commitment culture. The purpose of the evaluation was to obtain a first statistical indication of how well we are succeeding. The evaluation is based on 74 answers from 150 questionnaires among students who have been in gainful employment for at least two years. The responders have gained the necessary real world experience to evaluate our academic attempts to simulate the real world. The results of the questionnaire are presented in Diagram 1. The numbers represent the estimated success as a percentage 'Voluntariness' has the lowest score while 'Fulfillment' has the highest one. It is notable that the elements with the lowest scores also have the highest standard deviations.

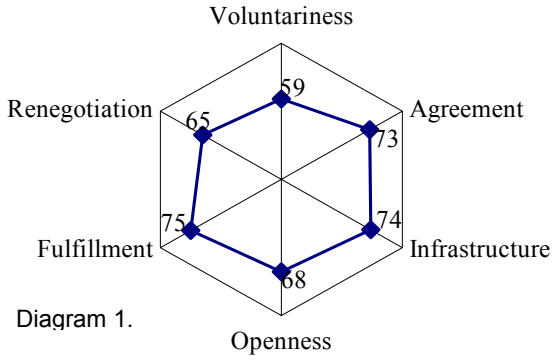

Diagram 1.

The above result came as no surprise to us – we are aware that certain factors must be improved when teaching all six elements of commitment. Furthermore, we have found voluntariness, renegotiation and openness harder to teach than infrastructure and agreement. Gratifying for us is the high score for the element fulfillment. Credibility has a close relationship with commitment, and since credibility is earned primarily by continuously fulfilling commitments we were pleased to see this score. In summary, it was gratifying to note that all means exceed fifty percent. Furthermore, ten percent, fairly distributed between the six elements, of the individual scores received, estimated the success to 100 percent. To conclude, we consider the result from the evaluation as satisfying.

7. Conclusion

Since 1990 software engineering education at Blekinge Institute of Technology has used commitment as the underlying principle for learning. This educational approach has made it possible to address the discrepancies between education and industry. Through actively seek, agree upon, and fulfill responsibilities on own initiative, the students are continuously confronted with situations that polish up the skills necessary to handle issues that seem to be out of their control. For more than a decade six elements of commitment have been developed and tried out as part of the curricula. In this paper is presented the first evaluation of among

the students who have graduated from our program; with the purpose to obtain a first statistical indication of how well we are succeeding. All scores are within the interval of fifty-nine to seventy-five percent, indicating positive attitude towards the curricula. The emphasis for teachers who adopt the commitment culture is on supporting rather than on directing. Even though focus is on support towards the students, it is still possible to formulate and use relevant examination criteria; criteria that clearly relate to the commitments that the team as a whole, the sub-teams, and each individual make.

8. References

[1] M. Shaw, "Software Engineering Education: A Roadmap", Proceedings of the conference on The future of Software Engineering, 2000

[2] K. Beckman, N. Coulter, S. Khajenoori, N. R. Mead, "Collaborations: Closing the Industry - Academy Gap", IEEE Software Vol. 14 Issue 6 pp. 49-57, 1997

[3] C. Johansson, *Communicating, Measuring and Preserving Knowledge in Software Development, Licentiate dissertation*, Blekinge Institute of Technology, 2000

[4] K. E. Sveiby, *The New Organizational Wealth*, Berrett-Koehler Publishers Inc., 1997

[5] W. S. Humphrey, *Managing the Software Process*, Addison-Wesley, 1990

[6] J. Dewey, *Experience and Education*, MacMillan Publishing Company, 1997

[7] C. Argyris, *Knowledge for Action*, Jossey-Bass Inc. Publishers, 1993

[8] D. A. Kolb, J. Osland, I. M. Rubin, *Organizational Behavior: An Experiential Approach*, Prentice Hall, 1995

[9] L. Ohlsson, C. Johansson, "A Practice Driven Approach to Software Engineering Education", IEEE Transactions on Education, Vol 38, No. 5, 1995, pp. 291-295

[10] Software Engineering Code of Ethics and Professional Practice, htpp://acm.org/serving/se/code.htm, ACM/IEEE-CS, 1999

[11] S. Jarzabek (ed.), "Teaching Software Project Courses", Forum for Advancing Software Engineering Education (FASE), vol. 11, no. 6, June 2001, www.cs.ttu.edu/fase

[12] T. B. Hilburn, W. S. Humphrey, "Teaching Teamwork", IEEE Software September/October pp. 72-77, 2002

[13] M. Daniels, X. Faulkner, I. Newman, "Open Ended Group Projects, Motivating Students and Preparing them for the 'Real World'", Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02), 2002

[14] D. Dalcher, M. Woodman, "Together We Stand: Group Projects for Integrating Software Engineering in the Curriculum", Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), 2003

[15] M. Gnatz, L. Kof, F. Prilmeier, T. Seifert, "A Practical Approach of Teaching Software Engineering", Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), 2003

[16] T.C. Lethbridge, "What knowledge is important to a Software Professional?", IEEE Computer , May 2000

[17] D. J. Bagert et al., "Guidelines for Software Engineering education Version 1.0", Technical report CMU/SEI-99-TR-032, 1999

[18] W. S. Humphrey, *Managing Technical People*, Addison-Wesley, 1997

[19] K. Beck, M. Fowler, *Planning Extreme Programming*, Addison-Wesley, 2001

[20] K. Schwaber, M. Beedle, *Agile Software Development with Scrum*, Prentice Hall, 2002

[21] A. Cockburn, *Agile Software Development*, Addison-Wesley, 2002

[22] W. S. Humphrey, "Credibility and Commitment", http://www.sei.cmu.edu/publications/articles/watts-humphrey/credibility-commitment.html

[23] J. Huffman Hayes, T. C. Lethbridge, D. Port, "Evaluating Individual Contribution Toward Group Software Engineering Projects", Proceedings of the 25th International Conference on Software Engineering", (ICSE'03), 2003