Relatório Técnico

Reviewing the Curriculum of Computer Science Undergraduate Courses to Incorporate Communication and Interpersonal Skills Teaching

V. M. Teles
C. E. T. de Oliveira

NCE -26/02

Reviewing the curriculum of computer science undergraduate courses to incorporate communication and interpersonal skills teaching

Vinícius Manhães Teles
Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
vinicius@improveit.com.br

Carlo Emmanoel Tolla de Oliveira
Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
carlo@ufrj.br

Abstract

The major problems of software development projects are not so much technical as sociological in nature. The industry seems to agree very much with this statement while the university seems to give it little importance. The article begins analyzing some related work and proposes change in the computer science undergraduate course to accommodate new ways of teaching and to incorporate professional skills teaching. It also describes a course on extreme programming where some techniques will be used and evaluated.

1. Introduction

The main emphasis in Computer Science courses is on the development of technical skills by the students undertaking the course (IEEE/ACM apud Pham, 97).

This focus on technical skills poses problems for the future professionals when they are supposed to build information systems where their activities are part of a team effort. According to DeMarco et al. [1] since they work in teams and projects and other tightly knit working groups, they are mostly in the human communication business. Their successes stem from good human interactions by all participants in the effort.

Researches conduced in Australia seem to confirm the statement above. In this country, surveys of employers have shown that the qualities they consistently rate most highly in graduates relate to their communication skills, their ability to work together in teams and their technical writing skills, besides their basic technical knowledge. (Business Higher Education Round Table and Department of Employment, Education and Training apud Keen, 98)

The Business/Higher Education Round Table, Australia conducted a survey in 1992 in which both business and universities were asked to rank the desired characteristics of university graduates. The results are shown in Table 1. [2].

Desired Characteristics of Graduates	Rank Business	University
Communication skills	1	7
Capacity to learn new skills and procedures	2	5
Capacity for cooperation and teamwork	3	8
Capacity to make decision and solve problems	4	3
Ability to apply knowledge to workplace	5	4
Capacity to work with minimum supervision	6	6
Theoretical knowledge in a professional field	7	1
Capacity to use computer technology	8	2
Understanding of business ethics	9	12
General business knowledge	10	11
Special work skills	11	9
A broad background of general knowledge	12	10

Table 1. Desired Characteristics of University Graduates (where 1 is the most desired)

This survey indicates that business and universities differ in their ranking of the importance of characteristics in the graduates in two major areas:

1. Communication skills, a capacity to learn new skills and procedures, and a capacity for cooperation and teamwork. In each of these cases the universities rankings and well below those of business, particularly in the area of communication skills.

2. Theoretical knowledge in a professional field and a capacity to use computer technology. In these cases universities have rated these characteristics much higher than has business. [2]

These results are consistent with the views of some authors, like DeMarco and Lister [1], Goguen and Linde [15] and Goguen [9]. They believe that software development is strongly affected by social issues. So, it cannot be treated in a purely technological way.

Other researchers have been studying methods to improve the education of software engineering in order to address these social issues. In this paper, the authors review some recent developments in this area and proposes a review in the computer science curriculum in order to accommodate these methods. In particular, they start proposing the introduction of an extreme programming course which will use some of the techniques presented in this article.

2. Related work

Researchers are working in two different propositions:

- Changes in the teaching of technical content
- Introduction of professional skills teaching

Each of these alternatives are described in the sections ahead.

2.1. Changes in the teaching of technical content

The Software Factory

Tvedt et at. [13] propose the "Software Factory", an eight-semester sequence of courses. These courses expose students to large-scale, team-oriented development in a software development organization. Each course represents a specific software engineering role or job within the development organization.

The eight semester sequence progresses the students through the following roles: (1) Software Factory process and tools trainee, (2) software system testes, (3 & 4) software developer and maintainer, (5) requirements analyst and test planner, (6) software designer and (7 & 8) software project manager.

Students from all courses in the Software Factory sequence meet simultaneously to fulfill their roles in the software organization. The enrollment in the program allows for multiple teams within the organization.

The Software Factory approach is rather interesting since students begin to work in a long term project early, exercise teamwork and face non-technical issues. Furthermore, it seems to bring students closer to what happens in the "real-world". On the other hand, the adoption of this approach has considerable impact on the way faculty staff is used to work and demands a significant infrastructure.

Large-scale software development

Sebern [16] also works with the idea of large-scale software development. It proposes a software development laboratory in which student teams work for extended periods on large-scale, ongoing projects in the context of a standardized and evolving development process. It's composed of a three-course sequence.

Students in this course believe they have achieved the objectives related to teamwork, process improvement and software development practice.

As in the case of Tvedt [13] this sequence reflect much of the experience in the "real-world". But it is also hard to be implemented by faculty staff.

The Activity Weekends

Ratcliffe et al. [11] describe the adoption of an integral activity weekend as part of the faculty's introductory software engineering course and a second weekend to reinforce industrial awareness in the student's second year.

These weekends have been developed to improve motivation and staff-student relations, emphasizing on life skills and adaptability. The idea is to introduce students to the concept of team skills. With specific attention to personal challenge and team dynamics, these weekends where carefully designed to both improve motivation and enhance the general employability of the students.

They follow a series of specially tailored outdoor activities that are designed to promote self and inter-personal skills through a series of shared group experiences. The activities are personally challenging and are heavily teamwork oriented. They cannot be carried out successfully without group co-operation and group encouragement.

Ratcliffe et al. [11] state that the response from the weekends is overwhelming. The students obviously enjoy themselves a great deal but more importantly they learn a great deal about themselves and working with others. It seems that the students learn far more about team working in one weekend than could have been taught to them through a whole series of class projects.

Although the approach is very interesting and effective, the costs are high. Considering the limitation of resources that affect many universities, it's necessary to find a way to implement these weekends with lower costs.

Case Studies

The use of case studies, a pedagogical approach which has been used successfully for many years in a variety of business schools (Alfred Aho apud Joan Krone et al., 02) is also proposed by some authors.

Krone et. al. [17] developed a course in which students work with both an industrial partner (or an industrial case study) and a faculty member to apply theory and current research to real problems. This approach goes far beyond the usual internship by setting up a partnership in which students, faculty and industrial partners work together, each bringing a special perspective to a particular problem.

The course is heavily based on case studies which have to be prepared in advance. The industrial problems are documented as case studies, using a specific format. The problem statement and background material are created along with a proposed solution.

This course seeks to tighten the loose connection between theory and practice in computer science education by utilizing a partnership between academia and industry to document industrial problems with non-trivial technical issues and to transfer that knowledge into the classroom.

Fuller et al. [18] also use case studies to teach technical content. But they focus on the teaching of software risk management. They propose the use of case studies, based on the history of real projects. The case studies will be drawn from industry and students will be asked to perform risk assessment based on data that was available at certain times throughout the project. The students' assessment can then be compared with actual outcomes. In this way the student constructs their own experiential background, becoming progressively more familiar with all kinds of risks and their impacts on particular types of projects.

Open Ended Group Projects

Daniels et al. [10] describe the use of Open Ended Group Projects (OEGP). OEGP is a form of experimental learning (Kolb apud Daniels et al., 02) which can, in principle, be used to advantage to teach any subject with a practical application.

Daniels et al. [10], state that in addition to supporting knowledge acquisition, OEGP can be used to help the students gain and improve skills. The most obvious skill areas which are involved are interpersonal communication and group working. However, a suitably designed OEGP can ensure that students must consider the problems of communication with manager and client and can help improve both report writing and presentation skills. OEGP also assist in getting students to analyze problems and synthesize solutions while examining, and trying to mitigate the risks of things going wrong, all valuable skills for the software engineering project managers of the future.

The experience of group project work prepares the students for their subsequent careers where group working is the norm. Undertaking open ended projects also appears to have the benefit that they force the students to think about the problem rather than spending time searching for the 'correct' answer.

It is also noted that OEGP appear to have measurable beneficial effects on student performance in other academic subjects. Improved motivation and greater enthusiasm seem to carry over into general performance, confidence levels go up and problem solving skills improve so that students are more willing to attempt difficult tasks. OEGP can also be used to encourage students to apply theory which should lead to a better understanding of the theory and thus to improved performance in examinations.

Daniels et al. [10] describe several projects run in the classroom and they describe the students' reaction. Firstly, feedback from the students has been generally very positive. In all of the OEGP with which the authors have been involved there has been positive feedback from the students both during the module and afterwards. It is also noticeable that the levels of motivation of the students appeared to be higher with better completion rates, less plagiarism and very few drop outs or failures.

But OEGPs raise some criticism. Daniels et al. [10] explains that the main concerns that are expressed when the use of OEGP is suggested relate either to the use of group projects at all ("weak students get 'carried', good students get 'pulled down'") or to the fact that the outcome for an OEGP is inherently unknown i.e. that there is no 'right' answer. The necessity for group working is, however, becoming more widely accepted now (Ford apud Daniels et al., 02), thus it is the concerns about the open ended nature of the project and the need for fair assessment based on problems for which there is a correct answer, which are addressed here.

The obvious counter argument is that OEGP mirror real life software engineering projects which do not usually have known 'right' answers and there is a need to assist students to learn this before they start to work. Part of this learning process includes the intrinsically difficult process of finding out what the client thinks is required (Veryard apud Daniels et al., 02), negotiating with the client to agree what can be done and, later, explain what has actually been done and how it relates to the requirements. An alternative argument is that, ultimately, all criteria are established and judged by people and are, therefore, subjective. Objective criteria are only regarded as objective because there is agreement about the 'correct' way in which something should be done, or said.

History suggests that most such agreements change over time and current 'right' way may well be revised later.

A different perspective on the fair assessment of OEGP can be provided by considering the way in which science and engineering are advanced. All research projects have unknown outcomes but the methods used to undertake and present research are common. Thus it is possible to provide a fair assessment process for OEGP by focusing on the process which the students use rather than the product they produce [10].

2.2. Introduction of professional skills teaching

Lamp et al. [3] state that changes to the traditional systems development life cycle towards use of package software, prototyping, distributed computing, JAD etc have all placed a further demand on interpersonal skills as opposed to technical skills.

They described a major revision of the undergraduate teaching programme of the Department of Computer Science at the University of Tasmania, Australia.

This review introduced the teaching of professional skills. The objectives were:

- to introduce students to a range of professional skills considered essential for their effective operation as IT professionals;
- to develop skills and attitudes in students appropriate to IT professionals;
- to ensure that all times the acquisition of these professional skills are seen by the students as relevant to the technical and theoretical programmes which they are concurrently receiving, and as being essential for the well grounded graduate.

The second objective ensured that the professional skills training was a genuine skills-based programme, and not just an attempt to impart knowledge. Students are required to actively participate and to acquire appropriate levels of skills through experiential learning.

The programme encompass subjects that are taught along four years distributed as follows:

First Year

- What is the role of an IT professional?
- Ethics in computing
- Study skills
- Report writing
- Cross cultural communication
- Legal Issues

Basic Communications Concepts

- Information gathering
- Interviewing
- Organizations: groups & teams
- Meetings & decision making
- Marketing, presentation skills
- Presentations by students

Second Year

Working in groups and teams

- Groups and Teams
- Information Gathering
- Interviewing and ethics
- Interviewing
- Interviewing Exercises
- Metaphors
- Groups dynamics
- Meetings
- Team decision making
- Group problem solving
- Presenting a proposal
- Group presentations

Third Year

- Teamwork
- Team leadership
- Presentation skills
- Assertiveness
- Negotiation, conflict resolution
- Career visualization
- Skills/education for life
- Contract negotiation
- Detailed case study analysis

Honours

- Advanced ethics, legal issues
- Critical analysis
- Research skills
- Presentation skills
- Thesis writing
- Organizational contexts

According to Lamp et al. [3] the inclusion of the professional skills programme complements the social and human aspects of information systems by giving real experience in organizational and team based activities, and focuses understanding of the impact of information technology at the interpersonal level.

3. Proposal

Based on the evidences given before, the authors of this paper are working on a research project in order to change the way computer science is taught at Federal University of Rio de Janeiro.

They will try some of the techniques presented in this paper both changing the way technical subjects are taught and introducing professional skills teaching.

At the present moment, experiments are underway in a technical course on Extreme Programming. The authors are using concepts presented in this paper such as:

- The adoption of a large-scale project
- The involvement of an external client
- Open Ended Group Project

Their approach is to change the curriculum gradually. They will experiment the techniques in different subjects and introduce new ones especially for the professional skills teaching.

3.1. The Extreme Programming Course

This is a semester long course for a class composed of 20 students. The content is based on the books of Beck [4], Beck and Fowler [14] and Jeffries et al. [7].

The introduction of extreme programming teaching to computer science undergraduate courses is not new. It has already been described in the works of Shukla and Williams [12] and Müller and Tichy [6].

Shukla and Williams [12] describe a course based on a 16-week semester class where the students completed four Java programming projects during the course of the semester. Three of the projects were completed as the students were learning and using more traditional software development practices. These practices were based on the Collaborative Software ProcessSM (CSPSM) developed by Williams.

They found that one semester is not long enough to teach two very different methodologies nor for the students to perform meaningful assignments using two very different methodologies.

Furthermore, the students didn't work at all times co-located and with a customer on site. So, the experience didn't reflect the work of a real extreme programming project where co-location is extremely recommended and the customer should be present.

Müller and Tichy [6] developed an extreme programming course where, in the first three weeks students solved small programming exercises to familiarize themselves with the programming environment and to learn XP practices. The exercises introduced jUnit (the testing framework used throughout the course), pair programming, the test practices of XP (write test cases before coding, execute them automatically with jUnit), and refactoring. The remaining eight weeks were devoted to a project on visual traffic simulation. The course language was Java. All students had experience with Java from their early undergraduate courses.

As in the case described by Williams, the students didn't work at all times co-located and with a customer on site which brought some problems.

The proposal of this paper tries to overcome the problems found in both cases. Firstly, it focus only in one methodology, the extreme programming. Secondly, the students will work only in one project through the course. And they will develop a software during the classes, so they will always be co-located and will always have the customer on-site.

The project will be the development of a software to support the dissertation of a graduate student who will act as the customer all over the course. This approach is interesting, because the teacher won't act as the customer. He will act as the coach of the team. This project will use Java as it's development language.

The class will be divided in two to form two teams of 10 students each. In each team, the students will always work in pairs. They will practice pair-programming at all times following the recommendations of Williams et at. [5][8].

The students will be evaluated in two ways, according to their individual achievements and their behavior working in the team.

Every week the teacher will propose a reading assignment for the students. They will have to read a text and answer the questions posed by the teacher in the next class. The texts are used so that students can learn more about each characteristic of the methodology. This evaluation will represent half of the week grade.

The other half is based on teamwork. Students will be observed while they work in pairs and the teacher will grant the marks according to the way they communicate with the pair and the other teammates.

The teacher won't evaluate the final product, but only the process in which students build the product. So, the student's are not supposed to work on the project when they are not in class.

These are 4 hour classes once a week. In the first hour the teacher will ask the students about the text they've read along the week before the class. In the second hour the teacher will give a lecture on the subject of the class. And in the remaining two hours the students will work on the project.

The authors expect to overcome the problems found in previous experiments of this type. And hope the students will improve their communication and teamwork skills.

4. Conclusion

This paper began describing the concerns of the industry with graduate standards in the areas of communication and interpersonal skills. Both empirical research and anecdotal evidence confirms that industry remains strongly concerned over the teaching of this area.

The authors presented some works that try to address those concerns. They do that changing the teaching of technical content or introducing the teaching of professional skills.

Finally, they propose the adoption of some of these techniques in order to change the teaching of computer science at the Federal University of Rio de Janeiro. They start with experiments introduced in a course on Extreme Programming which is underway.

11. Future Work

This experiment is the first of a series of experiments the authors intend to put in action in their research which looks for new ways of teaching computer science courses in order to shorten the gap between industry expectation and university expectation. The results of this experiment will be available in the future in the form of a new paper.

References

[1] T. DeMarco, T. Lister, *Peopleware Productive Projects and Teams,* Dorset House Publishing Co., New York:, 1987.

[2] C. Keen, C. Lockwood, J. Lamp, "A Client-focused, Team-of-Teams Approach to Software Development Projects", *Proceedings of Software Engineering Education & Practice (SE: E&P'98)*, IEEE Computer Society Press Dunedin, 34-41

[3] J. Lamp, C. Keen, C. Urquhart, "Integrating Professional Skills into the Curriculum", *Proceedings of the First Australasian Conference on Computer Science Education*, Sydney, Australia, pp. 309-316, July 1996.

[4] K. Beck, *Extreme Programming Explained – Embrace Change*, Addison Wesley, 2000.

[5] L. Williams, R.Kessler, W. Cunningham, R. Jeffries, "Strengthening the Case for Pair Programming", *IEEE Software*, vol. 17, pp. 19-25, July 2000.

[6] M. Müller, W. Tichy, "Case Study: Extreme Programming in a University Environment", *Proceedings of the International Conference on Software Engineering 2001 (ICSE 2001).*

[7] R. Jeffries, A. Anderson, C. Hendrickson, "Extreme Programming Installed", Addison-Wesley, 2001.

[8] L. Williams, R. Kessler, "All I Really Need to Know about Pair Programming I learned in Kindergarten," *Communications of the ACM*, vol. 43, pp. 108-114, May 2000.

[9] J. Gouguen, "Requirements Engineering as the Reconciliation of Technical and Social Issues", in Requirements Engineering: Social and Technical Issues, edited with Marina Jirotka, Academic Press, 1994, pp. 165-199.

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

[11] M. Ratcliffe, J. Woodbury, L. Thomas, "Improveint Motivation and Performance Through Personal Development in Large Introductory Software Engineering Courses", *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02).*

[12] A. Shukla, L. Williams, "Adapting Extreme Programming for a Core Software Engineering Course", *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02).*

[13] J. Tvedt, R. Tesoriero, K. Gary, "The Software Factory: Combining Undergraduate Computer Science and Software Engineering Education, *Proceedings of the International Conference on Software Engineering 2001 (ICSE 2001).*

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

[15] J. A. Goguen, C. Linde, "Techniques for Requirements Elicitation", *Proceedings of Requirements Engineering (RE'98), IEEE Computer Society*, 1998, pp. 152-164.

[16] M. J. Sebern, "The Software Development Laboratory: Incorporating Industrial Practice in an Academic Environment", *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02).*

[17] J. Krone, D. Juedes, M. Sitharam, “When Theory Meets Practice: Enriching the CS Curriculum Through Industrial Case Studies”, *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02).*

[18] A. Fuller, P. Croll, L. Di, “A New Approach to Teaching Software Risk Management with Case Studies”, *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02).*