e-mentor
No 2 (19) / 2007
www.e-mentor.edu.pl/eng

Increasing Interdisciplinarity by Distance Learning: Examples Connecting Economics with Software Engineering, and Computing with Philosophy

Gordana Dodig-Crnković, Ivica Crnković

This paper presents two distance courses aimed at promoting interdisciplinarity. The first one was an internet-based distance undergraduate course in software engineering and management of software development projects for students of management and economy. The goal of the course was to bridge the gap between disciplines of economy (management) and software engineering, transfer knowledge and provide necessary technical background for future managers who very likely in their careers will take part in software intense projects. Both the interdisciplinarity and the advanced e-learning technology of this course made it challenging. The second was a specialized level Swedish National Course in Philosophy of Computing and Informatics for students of computing, philosophy and design, which was a combination of a campus-based and a distance course involving several Swedish universities, with a group of distinguished teachers from both Sweden and abroad. The critical challenge of this course was the establishing of a new inter-discipline and overarching the gaps between traditions of disciplinary thinking.

Introduction

As education systems today in general do not offer training for interdisciplinarity and multi-disciplinarity, distance education can play an important role in providing additional degrees of freedom and facilitating communication between disciplines. This paper presents two case studies based on experiences with distance courses intended to promote cross-disciplinarity. The first one was a distance undergraduate course in software engineering and management for students of management and economy. The second was a specialized level course in Philosophy of Computing and Informatics for students of computing, philosophy and information design. The goal of the first course was to provide necessary technical background in software engineering for future economists and managers who very likely in their careers will take part in software intense projects. Both the advanced e-learning technology and the transdisciplinarity of this course made it challenging. The second case study will present The Swedish National Course in Philosophy of Computing and Informatics which was a combination of a campus-based and a distance course involving several Swedish universities (Dodig-Crnković, August 2006). The course engaged a group of distinguished teachers from both Sweden and abroad and had basically mail and web support, and a virtual library. The critical challenge of this course was primarily on the conceptual level. It was not about training philosophers in computing or training computer scientists in philosophy, but the focus was on establishing a new discipline and bridging the gap between disciplinary traditions of computing and philosophy.

The Case Study I – Software Engineering Course for Future Managers

One of the fundamental roles that education plays in the society is to enable students for their future professions. Managers and economists often have strategic positions in corporate organizations. Nowadays administration, but also business and technology projects are software intensive, so management is heavily relying on software use, maintenance and production. For managers in such organizations a good command of the language of computing technology such as software engineering

and a basic knowledge about software intense projects characteristics is a big advantage. When teaching non-specialists a very broad field in a compressed form, there is always a problem of the balance between the depth and the breath. Experiences from holistic approach (Berglund, 2004) have shown that there is a real challenge to achieve understanding without remaining on a superficial level.

In this paper we describe a case study, based on (Crnković et al., 2006): the course designed to provide students of management and economy with a basic knowledge in software engineering which goal was to contribute to the holistic approach to learning by bridging the gaps on several levels: (i) Presenting a software engineering course to students of management and economy; (ii) applying pedagogical approach from one country (Sweden) to another (Croatia); (iii) making a combination of distance and local learning. We will present the concept of the course, its performance and the results with emphasis on the challenges and lessons learned.

Goals and objectives of the course

Basics of software development. One of the goals of the course was to give an insight in the basics of software development to students of management and economy. By completing the course the student should be able to understand basic characteristics, processes and some of technologies for software design. Since software design and development are not trivial it would be naïve to expect achievement in deep understanding of all aspects of software development. Rather the goal was to (i) make students aware of these aspects, (ii) make students capable of understanding the basic principles, (iii) train students to distinguish different solutions based on different technological assumptions, (iv) prepare students to successfully participate in software development projects and (v) train students in communication and formation of their view of different aspects in software development projects.

Bridging Educational Cultural Gaps. The second objective of the course was to combine teaching traditions from different educational cultures. The idea was to apply Swedish style teaching to Croatian students. Although in many aspects similar, approaches in Croatian and Swedish education are somewhat different. Swedish education has a tradition in keeping education pragmatic, related very much to the principles of “learning by doing”. A second characteristic of Swedish education is teamwork – a strong feeling for collaboration and sharing of responsibilities. Yet another characteristic in Swedish education system is exploratory type of education, focused on searching for knowledge when needed. Similar approaches are common in modern education theory, this being introduced nowadays also in the Croatian education system. However traditional approaches like focus on theory presented primarily in form of lectures and reading are the primary characteristic of Croatian education.

Getting Used to ICT in Education. The third objective of the course was to train students to perform in different learning environments. The course was a distance course – the lectures and seminars have been held via video conference system.

The challenges of a cross-disciplinary distance course

The concept of the course presented many novelties which were accompanied by a number of challenges – some of them related to the interdisciplinarity and introduction of new concepts and discourses, some of them characteristic for distance learning and some of them typical of software engineering (Crnković et al., 2000). Here are the main challenges met in the course.

The Grand Challenge: Training students to manage something that is usually considered far outside the scope of their primary study. The challenge is the following: Is it possible to provide an overall view, which will also be deep enough to highlight the essence of the problem and at the same time not require a detailed technical knowledge? In the concrete case, the challenge was to give insight in the software development to students whose primary interest lies in economy and management.

Covering many disciplines vs. concentration on particular disciplines of an interdisciplinary area. Software engineering is a large area that includes many disciplines. Teaching even the most important disciplines requires a complete academic program, not just a course. The problem is to select the most

important aspects of software engineering and to put them in a consistent framework.

Striking the balance between theoretical knowledge and practical experience. The main goal was to prepare students for the real world, which is inherently inconsistent and unpredictable. The academic world is often an "ideal" world in which students learn about problems and their solutions in a simplified form without details that later on appear in the real world situation. A common solution to this problem in software engineering education is to design courses based on practical examples from industry (Dawson et al.; Leventhal et al.). In the present case that possibility was not a feasible alternative, it would be an overly complicated step. Simple but realistic examples were desired instead. Further, a dilemma was how to weight the theoretical parts in relation to the practical part. Is it better to give the students a solid theoretical background, which they can utilize later on in the "real life", or to "throw them into the water and let them learn how to swim"? Again, to achieve a balance between the requirement for general high level understanding and the "hands on" feeling was a challenge.

Combining different degrees of independent work and supervising. One of the basic preconditions for the successful learning is to establish good relations between students and their teachers. However, the teachers can not guide students too closely, for there is a risk that students may stop thinking independently and instead rely completely on the guidance.

Establishing new forms of teaching. The course was organized as a distance course using e-learning technologies, which is a novel experience for the class. Moreover, the participants in the course were not familiar with the project and teamwork which constitute the main parts regarding students' engagement. These forms of teaching are neither typical for studies of social sciences, nor much used in Croatian education system. The course did not have a final exam, but the grade was the result of a student's performance during the course (continuous examination).

The Organization of the course

The outline of the course followed an approach typical of a software engineering course, giving an overview of the most important phases and activities of software products lifecycles. The elements that are most important from the management point of view have been emphasized (requirements management, project management and system design), while others (such as software implementation, verification and validation) were only briefly mentioned. Software project development's management aspects have been explored into more detail. The main goal was to train student's ability to (i) understand the customers' requirements, (ii) gain knowledge of the basics of software design, and (iii) to plan follow up and lead a software project.

A balance between theoretical knowledge and practical experience was particularly difficult due to students' lack of experience in software development, and even a limited experience of software usage. We could not count on any previous knowledge in computer science or software engineering, so that terms like *software system, building (implementation), source code, function,* etc. could not be used without careful explanation. In addition to the lectures, laboratory exercises were given in the first phase. The exercises included practical examples of the principles and methods presented in the lectures.

In the second phase of the course students worked on projects. Project groups consisted of four to six students. The assignment was to make a project plan, identify the requirements, and provide an overall design of a system. The goal of the project was twofold: a) gaining practical experience in project planning and insight in the software development process, b) getting experience in teamwork. The students had responsibility to organize the project without guidance from teachers. By periodical follow-ups they were asked to present the project status, and to discuss possible problems and proposals for the solutions. This aimed at increasing the ability of taking decisions, and increasing creativity and responsibility in the team work.
Finally the students made individual assignments which consisted in reviewing a selected chapter of a book on software management and writing an essay on a selected topic.

The distance learning techniques used in the course

During the course students were required to attend eleven lectures, finish three laboratory practices in groups of two and do the final group project work in a group of six students.

Fast Internet connection between lecture rooms in Croatia and the video-conference hall in Sweden made it possible to establish two parallel communication channels. The first videoconference communication channel was established with the specialized videoconference equipment which made it possible to transfer highest quality picture and sound. The teacher from Sweden gave lectures and students in Croatia participated through the first videoconference channel. The second videoconference connection was established for presenting of lecture materials given through the separate projector over videoconferencing tool built in Microsoft Windows XP – NetMeeting. In that way students were able to see on the second projector programs used by the teacher during the lecture. This was good for students' practical understanding and solving laboratory exercises, making project work and handling project documentation. The lessons were recorded as Flash movies. Students could download these movies or view them from the course web page.

The WebCT system (www.webct.com) was used as a repository for all data: lectures, exercises, messages. Students, professor and assistant on the course were able to communicate and exchange the files via WebCT. For the purpose of exchanging data and mails among the members of the project, special discussion groups were organized. Within a discussion group each member could post and upload their part of the assignment and see documents and posts that other students from the same group have uploaded.

In order to increase the communication within the class a real-time chat and whiteboard tool was provided by WebCT. Chat was used for messaging and whiteboard for creating project drawings, such as different UML diagrams. Students were encouraged to use any of the VoIP (Voice over Internet Protocol) programs with whiteboard, in case they had broadband Internet connection.

Students were given tutorials in Flash which explained how to use more complicated tools like the one for submitting assignments. The professor and the assistant were also communicating through videoconferencing and VoIP programs (NetMeeting and Skype).

The Results and Lessons Learned

Of twenty students who participated in the course eleven passed the course. Four more students have got the incompletes. This is an acceptable pass-through percentage, although it is not the best one. The students' satisfaction with the course was in general quite high (between 3 and 4 on a 1-5 scale), and the different elements of the course had approximately the same evaluation grade, see (Crnković et al., 2006) for the figures.

Students also provided a list of recommendations for the improvement, as well as the comments about which part of the course they appreciated mostly. The following was typical of what was considered to be on the positive side:

- The course concept including lectures, exercises, projects and reporting.
- Flexibility of the course.

The parts that according to students' opinion presented a problem:

- Distance learning decreases possibility of bidirectional communication.
- Difficulties with motivation for some students that affected the final project group results.

The students' results have been in certain aspects surprisingly good. Results of requirements analysis were comparable with the computer science student's results. In addition, the students demonstrated surprising creativity in finding new requirements. Some of the results have been expected – the system specification was not in the rank of those which students of computer science would do. Some

students had problems in understanding of concepts of object-oriented approach, and some did not manage to provide solutions that followed UML formalism. On the other hand the most reviews and essays have been of higher quality than those of computer science students. The best reviews were so good that the publisher decided to give each student a copy of a book as a gift.

The most unexpected student results came from the project work. The quality and distributions of the results have been expected. What was surprising was the attitude of students toward the team work. Individual interests were strongly preferred to the interest of the entire group. In the majority of the cases the “teams” have been working as groups of independent individuals rather then as coherent collaborative teams. The students have been more accurate in performing the individual tasks (when assigned by the teachers) than in participating in the common work. Even when a common work was explicitly required, students have divided the tasks and made them individually. The project work has clearly showed that additional training in teamwork was necessary. The similar experience the teacher had with the international students visiting Sweden, and an analogous experience has been reported in (Berglund, 2004). The lesson learned here is that introduction of teamwork requires additional efforts. This finding is in line with students’ evaluation and comments. One of the lessons learned from the course was that the distance learning requires a good technical support, but also that the local support will increase the individual’s involvement in the group.

The Case Study II – The Swedish National Course in Philosophy of Computing and Informatics

The second case study this article presents considers experiences with the specialized level Swedish National Course in Philosophy of Computer Science held at MDH University, year 2004. Participants from a number of Swedish universities joined this cross-disciplinary course that was organized for the first time, with the aim of introducing the research field of Computing and Philosophy in Sweden.

Before starting this course the first step was to delineate and try to define the field of interest. This also implied the discussion about the nomenclature: Philosophy of Computing or Philosophy of Information or Informatics? Here Informatics is defined as synonymous of Informatik / Informatique/ Informatica and alike European terms which are equivalent to ”Computing” in its ACM/IEEE interpretation (i.e. Computer Science, Computer Engineering, Software Engineering and Information Systems). The English term "Computing" has an empirical orientation, while the corresponding German, French and Italian term “Informatics” has an abstract orientation. This difference in terminology may be traced back to the tradition of nineteenth-century British empiricism and continental abstraction respectively.

There is an obvious difference between the two main streams of Philosophy of Information and Computing: computation-oriented and information-oriented. Computation current is particularly focused on the nature of the process of computing, its meaning and its mechanisms. It is much more focused on mathematic and logic than the information-oriented stream which is mostly social and human-centered and has many broad interfaces to humanities (like e.g. library information science). The concept of information itself is so fundamental that it is common of all our knowledge and in a wider sense it comprises every perception and even every physical/ material phenomenon. That is also the reason why a sharp border between the fields is impossible to draw.

The Philosophy of Information is according to L. Floridi, "What is the Philosophy of Information?" (Metaphilosophy, 2002): *A new philosophical discipline, concerned with:*

a) the critical investigation of the conceptual nature and basic principles of information, including its dynamics (especially computation and flow), utilization and sciences; and

b) the elaboration and application of information-theoretic and computational methodologies to philosophical problems

More about Philosophy of Computing, that besides the classical computation represented by Turing paradigm encompasses even the critical analysis of the emerging research field of natural computation; see Dodig-Crnković (September 2006).

Informatics is a modern discipline that builds on science (where the term science also encompasses very central disciplines of mathematics and logic), and technology. In some of its parts (e.g. AI) Informatics has strong relations to philosophy, psychology, ethics, aesthetics and art. At present there is a vital need to formulate and disseminate critical reflections over the foundations of Informatics, its connections to other fields of human endeavor, its prospects and its limitations within the framework of Philosophy of Information.

In that respect, the following proclamation of the Japanese Philosophy of Computation Project is significant: *The mission of the Philosophy of Computation Project is to reconsider various concepts of computation innocently used in Philosophy, Mathematics, Computer Science, Cognitive Science, Life Science, Social Science, etc., and reveal global problems hidden in each realms. We don't only aim to answer particular questions but also to provide universal viewpoints which are thought of important for this new subject.*

It is important to notice that Computing is changing the traditional field of Philosophy of Science in a very profound way. First as a methodological tool, computing makes possible "experimental Philosophy" which is able to provide practical tests for different philosophical ideas. At the same time the ideal object of investigation of the Philosophy of Science is changing. For a long period of time the ideal science was Physics (e.g. Popper, Carnap, Kuhn, and Chalmers). Now the focus is shifting to the field of Computing/Informatics.

There are many good reasons for this paradigm shift, one of those being a long-standing need of a new meeting between the sciences and humanities, for which the new discipline of Computing/Informatics gives innumerable possibilities. Contrary to Physics, Computing/Informatics is very much human-centered. It brings a potential for a new Renaissance, where Science and Humanities, Arts and Engineering can converge to reach a new synthesis, so very much needed in our intellectually split culture. Contemporary trends in the establishing the relation between the Philosophy of Science and the Philosophy of Computing and Information are developing in the direction of replacing the historical relation between Philosophy of Science and Philosophy of Physics.

In a very enlightening way Philosophy of Informatics (PI) brings together phenomena and methods otherwise completely disparate. A future project of synthesis, a new Renaissance with human in its centre, can be accommodated within the methodological and conceptual space of PI.

One of the goals of the PI project was to shed more light on the foundations of Informatics and its future possibilities. The project is building on scientific traditions and relates problems of Informatics to the classical sciences in order to widen the perspective and explore the sets of values and ethical grounds for a discipline. It does not imply that Informatics itself can be reduced to a science. The ambition is to explore to what extent and in what ways it builds on scientific (again inclusive mathematics and logic) traditions and what other traditions may be used in order to better understand the present and future development of Informatics.

The PI-network. Goals and objectives of the course

The project started by initiating the PI-network (Philosophy of Computing and Informatics Network) in the year 2003. The aim was to prepare the Swedish National Course in Philosophy of Computer Science, and was supported by Knowledge Foundation, KKS. Network members include Ulla Ahonen-Jonnarth and Jan Odelstad from Gävle University; Björn Lisper, Peter Funk, Jan Gustafsson and Gordana Dodig-Crnkovic from MDH; Torbjörn Lager, Göteborg University and Joakim Nivre, Växjö University.

The course consisted of lectures given by specialists in different fields of philosophy and computing, class discussions and writing a research paper. Issues covered philosophical foundations such as the fundamental nature of computation, methodology of Computer Science and the scientific ideal of Physical sciences, modeling and simulation issues and ethical, societal and artistic aspects of computing.

We addressed conceptual foundations of Philosophy of Computing/Philosophy of Information, including critical examination of the concept of computing, its models and metaphors, from data types and programming languages, programs to processes, architecture to abstraction. Several parts of the course explored the use of computational paradigm in the related fields.

Answering the question: *Why Philosophy is important for Computing?* led to several suggestions, such as: Philosophy is providing ”thinking tool-box” and access to paradigms, metaphors, historical examples (knowledge capital). It can improve communication – within computing community and also more widely, making the context of the computing field explicit – both its conceptual and cultural framework. It was also pointed out that humanist dimensions of higher education are important. In the Knowledge Society with automated production, organization and even automated discovery, genuine human thinking abilities will make all the difference.

We also asked the opposite: *Why is Computing Important for Philosophy?* That was answered by several suggestions. Computing is a way to implement simulated or experimental philosophy with experiments “in silico” (or alternative constructed cognitive/computational systems), an innovative extension of ancient tradition of thought experiments. Computing makes possible application of computational modeling schemes to questions in logic, epistemology, philosophy of science, philosophy of biology, philosophy of mind, and so on. Computing paradigms and metaphors have shown interesting for cognitive science and also for philosophy.

We hope to see the network activity as well as a graduate course develop in the future, possibly as a distance course in collaboration with other universities in Sweden and abroad. This will certainly broaden our experiences and allow identifying further relevant topics that may be included in PCS.

The course was held during the period January – May 2004, with the following syllabus.

I. PHILOSOPHICAL FOUNDATIONS 22/01 – 23/01
Introductory lecture: What is PI?, Luciano Floridi, Oxford University
Physics as a traditional model of the ideal science for Philosophy of Science, Lars-Göran Johansson, Uppsala University
Philosophical Foundations of Computation, Gordana Dodig-Crnković, MDH

II. METHODOLOGY, MODELLING AND SIMULATION 04/03 – 05/03
Methodological Foundations of Computer Science, Erik Sandewall, Linköping University
Methodological and Philosophical Aspects of Modelling, Kimmo Eriksson, MDH, and Lars-Göran Johansson, Uppsala University
Critical Analysis of Computer Science Methodology, Björn Lisper, Jan Gustafsson, MDH

III EHICAL AND SOCIETAL ASPECTS 13/05
Ethics, Professional Issues, Gordana Dodig-Crnković, MDH
Computers in Society - Culture and Art, Gordana Dodig-Crnković, MDH
AI and Ethics, Peter Funk

IV MINI CONFERENCE - Presentations of research papers written by course participants.
More information about the course may be found at
http://www.idt.mdh.se/personal/gdc/PI_04/index.html

The discussions within PI-network about the course content were based on the books given in the list of references, along with the web resources that may be found on the course Virtual Library web page, http://www.idt.mdh.se/~gdc/PI-network-library.htm

Course literature

Apart from the Virtual library resources given on the course web page, http://www.idt.mdh.se/personal/gdc/pi-network.htm the following books were recommended and used in the course:

Luciano Floridi (Editor), Blackwell Guide to the Philosophy of Computing and Information, 2003

Terrell Ward Bynum and James H. Moor, The Digital Phoenix: How Computers are Changing Philosophy, 1998

Luciano Floridi, Philosophy and Computing: an Introduction, 1999

Timothy R. Colburn, Philosophy and Computer Science, 1999

James H. Moor and Terrell Ward Bynum, Cyberphilosophy: The Intersection of Philosophy and Computing, 2003

Results of the course

Participants from five different universities in Sweden (Blekinge, Dalarna, Mälardalen, Skövde, Uppsala) have taken part in the course. They have presented their research papers at the Mini-conference. Several articles written for the course have been accepted for international conferences (see the list in the next chapter).

The course demonstrated how PCS may be taught to different student groups with heterogeneous background and what questions course participants found most relevant in connection to their own research fields. Ten papers related to the course published in conference proceedings, journals and as PhD thesis chapters confirm high quality of the course outcome and its impact on the related research fields.

Members of PI-network, teachers and course participants have shown enormous enthusiasm and interest in the work in the course. The course evaluation results were very encouraging. We hope to see the network activity as well as the course develop in the future as a distance course in collaboration with other countries. This will certainly broaden our experiences and allow identifying further relevant topics that may be included in PCS. One of the results the course has achieved was publishing of ten research papers in journals, conference proceedings or as chapters of PhD theses.

The list of published articles, written in connection to PI course:

Rikard Land, *Understanding Evolution of Information Systems by Applying the General Definition of Information*, Proceedings of 26th International Conference on Information Technology Interfaces (ITI), Cavtat, Croatia- IEEE, June 2004

Sandra Ijeoma Irobi, *Correctness Criteria for Models' Validation - A Philosophical Perspective*, Proc. Models, Simulations and Visualization International Conference (MSV'04)], Las Vegas, Nevada, United States.

Imad Eldin Ali Abugessaisa, *Ontological Approach for Modeling Information Systems*, Proceedings of The 4th International Conference on Computer and Information Technology Wuhan, China, 2004 and will be published as IEEE CS.

Baran Çürüklü, *Early Stages of Vision Might Explain Data to Information Transformation*, Proceedings of the Engineering Of Intelligent Systems (EIS 2004), Madeira, Portugal, 2004

Christina Björkman, *Crossing Boundaries*, Focusing Foundations, Trying Translations: Feminist Technoscience Strategies in Computer Science, BTH, Dissertation Series No 2005:02, 2005, (A PhD thesis chapter)

Gordana Dodig-Crnković, Virginia Horniak, *Togetherness and Respect - Ethical Concerns of Privacy in Global Web Societies*, Special Issue of AI & Society: The Journal of Human-Centred Systems and Machine Intelligence, on "Collaborative Distance Activities: From Social Cognition

to Electronic Togetherness", CT. Schmidt Ed., Vol 20 n°3, 2006 (includes elements of Virginia's article).

Christina Björkman, Invitation to Dialogue: Feminist Research meets Computer Science, in Jacqueline Archibald et al (eds) The Gender Politics of ICT, Middlesex University Press, July 2005

Christina Björkman,"Feminist Research and Computer Science: Starting a Dialogue", in Journal of Information, Communication and Ethics in Society (ICES), vol 3, nr 4., 2005

Carina Andersson, R. Pettersson, How can a Design process and Scientific process in Information Design collaborate? Published in R. Rohatynski & J. Jakubowski (Eds.) Engineering Design in Integrated Product Development EDIProD 2004, Polish Academy of Science, Committee of Mechanical Engineering, 2004

Conclusions

It is interesting to compare the experiences of two different courses aimed at strengthening interdisciplinary and cross-disciplinary competence and learning practices.

The course in software engineering for managers was necessarily focused towards implementations and practical applications. The ambition to provide future managers and economists with a holistic view of management of software-intense projects turned out not to be simple to realize. Taking a holistic approach goes beyond exchanging basic facts between different knowledge communities. In many cases the basic facts are not sufficient to adequately describe specific cases of interest. Not only technical matters but also many informal, cultural factors can be expected to constitute barriers to reaching a common understanding between fields which are so far from each other that they usually do not communicate directly.

The Swedish National Course in Philosophy of Computer Science on the other hand was a specialized advanced level course, and the ambition was not to transfer knowledge but to generate knowledge through communication. It has demonstrated the positive potential of cross-fertilizing and synergy that happens in a class of highly motivated and competent learners interested in bridging gaps and sharing their knowledge both within the class and in the discussions with teachers.

We are convinced that holistic approaches which presuppose ability to communicate knowledge cross-disciplinary, trans-disciplinary and inter-disciplinary are necessary in modern education and that they can steadily be improved by learning from practical experiences about the differences and commonalities of research fields which give us possibilities for knowledge generation. Distance courses can play an important role in providing additional degrees of freedom and facilitating communication between various groups with their different cultures and educational practices.

References

A. Berglund, *A framework to study learning in a complex learning environment*, ATL-J, Research in Learning Technology, 12(1), pp 65-79, 2004

I. Crnković, M. Larsson, F. Lüders, *Implementation of a Software Engineering Course for Computer Science Students*, Proceedings, APSEC, Asia-Pacific Software Engineering Conference, Singapore, December, 2000

R.J Dawson, R.W. Newshman, R.S. Kerridge, *Bringing the 'Real Word' of Software Engineering to University Undergraduate Courses*, IEEE Proc. In Software Engineering, 144, 5-6(1997), 44-48.

I. Crnković, K. Aleksić-Maslač, H. Jerković, *Holistic approach in Education – Filling the Gap between Different Disciplines*, 28th International Conference on Information Technology Interfaces, 2006., p 35-40, IEEE, Cavtat, Croatia, June, 2006

G. Dodig-Crnković, *Swedish National Course in Philosophy of Computer Science*, North American Computing and Philosophy Conference, NA-CAP 2006 @ RPI, Troy, NY, USA, Rensselaer Polytechnic Institute, August 10-12, 2006, http://www.cogsci.rpi.edu/conferences/cap/schedule.php

G. Dodig-Crnkovic, *Investigations into Information Semantics and Ethics of Computing*, PhD Thesis, Mälardalen University Press, September 2006

L.M. Leventhal, B.T. Mynatt, *Components of typical undergraduate software enineering courses: Results from survey*, IEE Trans.Softw.Eng., vol SE-131,11(1987) 1193-1198

Information about the authors:

Gordana Dodig-Crnkovic is a Senior Lecturer in Computer Science at the University of Mälardalen, Sweden. Her primary research interests are in computing and philosophy, and philosophy of information. Her background is in theoretical physics; she has a degree in computer science, and teaches on formal languages and automata theory, research methodology, theory of science and professional ethics. She has published on the unified paninformational/pancomputational theory, information and computation semantics, and computer and professional ethics.

Ivica Crnkovic is a professor of industrial software engineering at Mälardalen University where he is the administrative leader of the software engineering laboratory and the scientific leader of the industrial software engineering research. His research interests include component-based software engineering, software architecture, software configuration management, software development environments and tools, as well as software engineering in general.