Integrating Code Camp Concept into CSE Curricula

Kari Heikkinen, Jouni Ikonen, Jari Porras

Abstract: *In this paper we present the Code Camp concept and how the concept can be integrated into CSE Curricula. Code Camp results as a short term collaborative learning have been excellent and this paper introduces an effort to apply the concept from single course to the whole curricula.*

Key words: *Code Camp, Collaboration, Intensive, Curricula Design.*

INTRODUCTION

Students learn different skills during their studies. These skills should be in align with the capabilities the student should possess when he/she graduates. In literature, e.g. [1, 2], different categorization, classification or clustering of such skills can be found. Furthermore, literature also provides lot of publications, e.g. [3, 4, 5] that aim to show the importance of these certain Computer Science skills. We have own classification approach (see [6] for more details and definitions for the skills) in which personal skills and knowledge essential to have when graduating are emphasized. In addition to the technical skills and knowledge, the students have accumulated a set of social skills. The personal skills and knowledge category contain skills such as critical thinking (CR), problem solving, design & modelling (DE), methodology skills (ME), programming (PR), project work (PW), written & oral presentation (WO), foreign language skills (LA), and evaluation and feedback skills (EF). The social skills category on the other hand has skills such as leadership (L), negotiation (N), communication & interaction (I), group work (G), and supervision (S). In order to master all these skills the curricula needs to emphasize all of them in various parts of the studies. Technical skills and knowledge is usually taken care of by various course topics but social skills require that teaching methods somehow take them into account. Our code camp concept aims to emphasize these often forgotten skills.

The rest of this paper is structured as follows. Following this introduction we present the fundamentals of the Code Camp concept. Some key figures of our previous code camps are given for reference of the concept. Next the curricula design and code camp concept are combined for gaining large scale benefits. Finally some conclusions and future ideas are given.

CODE CAMP FUNDAMENTALS

Code Camp is a teaching concept, in which students work together for intensive time period, e.g. 24 hours or five days. The event emphasizes heavily collaborative learning aspects, e.g. common tasks, small group learning, cooperative learning, and interdependence, even though the context of the event is build around learning some programming related skill. Students might not have nor are required to have any experience with the selected environment or programming language, but they are required to have sufficient background on programming. The Code Camp normally starts with an introduction to the subject after which students start the hands on phase. In the end of the event each group has to present their problem and the results. This requires students to plan their work, study the subject, collaborate, and even make changes to their plans in order to finish in time. Although set of instructors observe and help students in their problem solving, the main emphasis is on student-to-student collaboration. Students are encouraged to collaborate with each other's and extra points are given in evaluation for active collaboration. A collaborative platform, i.e. Wiki is used to promote collaboration, documentation and written communication. Due to cooperative nature of the work it was felt extremely important that all information is easily available for all the participants and questions as well as solutions are properly documented.

Fig.1 provides an overview of the Code Camp concept. On the top is the selected context to stimulate the learning, i.e. what Communications Software topic and programming environment and language is to be learned and practiced. The middle layer presents the various phases of learning of personal skills and knowledge (i.e. student activities). On the bottom, the same phased learning is supported by instructor's activities. Between the top and middle layer are the checkpoints that help both the students to coordinate their activities and the instructors to support their activities. The code camp always starts with an orientation day (Monday in Fig.1). The orientation is used for introducing e.g. the new software, software tools, platform, and the Code Camp as an event. Social gathering event in the evening of the introductory day builds coherence between students, their groups and between groups. The second day (Tuesday in Fig.1) usually starts with demo sessions and continues with the actual programming. The following days (Wednesday & Thursday in Fig.1) follow the same pattern as demonstrations are given when needed. On the last day (Friday in Fig.1) the students give presentation as well demonstration of their solution.

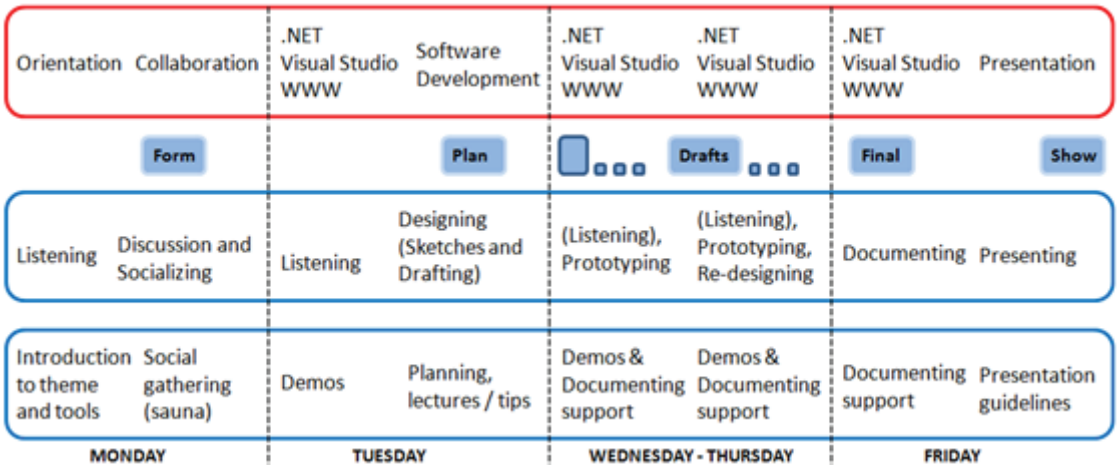

Fig.1. Code Camp structure

The main emphasis of the Code Camp is to promote skills and knowledge of the selected technical area. Table 1 presents a comparison of traditional courses and two Code Camps [7]. More detailed presentation is available in [7]. Code Camps are considerably shorter while producing better results, i.e. grades and passing percentage.

Table 1. Comparison of the code camp concept to the traditionally taught courses

	Course A	Course B	MUPE	J2EE
Type	Traditional	Traditional	Code camp	Code camp
Length	14 weeks 2+2 hours/week	14 weeks 2+2 hours / week	5 days 12-24 hours/day	4 days 12-24 hours/day
Students	170	80	98	33
Answering %	87 %	70 %	60 %	39 %
Study time	-	-	4,8	5,1
Initial self evaluation	2,25	1,875	2,2	1,5
Final self evaluation	3,7	2,4	2,8	2,6
Course grading criteria	Exam Practical work Viope	Exam Viope	Practical work: Idea Implementation Documentation Presentation CC spirit	Practical work: Idea Implementation Documentation Presentation CC spirit
Average course grade	3,5	2,6	4	4,1
Course passing %	41 %	65 %	88 %	100 %

In addition to the technical skills Code Camp concept emphasizes also social skills. Short time requires students to collaborate in order to finish in time. Course grading criteria (see Table 1) is also set in such a way that social skills are emphasized. In our setting, in particular, the students should practice:

- A foreign language skill as course is held in English and English is not their mother language.
- Group work as work is carried out in groups.
- Project management as they have to work in project manner and the project has a deadline to meet to do deliver their project.
- Social interaction skills as they have to collaborate in a group and with other groups.
- Presentation skills as each group have to present their results (in multiple occasions).

CURRICULA DESIGN AND CODE CAMP CONCEPT

Curricula and the courses included in the curricula define the skills and knowledge the students will gather in their studies. Various proposals for the curricula has been presented in the literature, e.g. [8]. Our curricula do not fully match directly any established curricula due to the reason that the research areas of different laboratories (machine vision, software engineering and communications software) are too far apart. However, by analyzing and comparing (see Table 2) our curricula to ACM/IEEE CS2008 curricula [8], we can argue that the match is very good in 8 areas (57%), good in 4 areas (29%) and poor in 2 areas (14%). In comparison, the match to ACM/IEEE IT2008 [9] was slightly worse, as only 6 areas were very good (46%), 3 areas were good (23%) and 4 were poor (31%). It should also be noted the core hours as listed in ACM/IEEE cannot be easily analyzed reliably with local teaching concepts or procedures.

Table 2 ACM/IEEE CS2008 and our curricula comparison

CS2008 Area	Curricula Match
Algorithms and Complexity	Very Good
Architecture and Organization	Good
Computational Science	Good
Discrete Structures	Very Good
Graphics and Visual Computing	Good
Human-Computer Interactions	Poor (do not exist)
Information Management	Very Good
Intelligent Systems	Good
Net-Centric Computing	Very Good
Operating Systems	Very Good
Programming Fundamentals	Very Good
Programming Languages	Very Good
Software Engineering	Very Good
Social and Professional Issues	Poor (do not exist)

Our laboratory (communications software) is responsible of roughly one third of the curricula for the Bachelor's degree. The responsibility area focuses on the fundamentals of networking and network security. Thus, our part of the curricula includes bachelor courses that contain fundamentals of the net-centric computing. These courses are for the purpose

of this paper named as introductory courses. Currently these introductory courses include Introduction to Telecommunications, Laboratory Course of Telecommunication Software, Basics of TCP/IP, Wireless Communications, Introduction to Information Security, Information Networks and Security and Web Programming. All these courses are mandatory for the Bachelor's degree. In the Master's degree each laboratory has its own major, i.e. own curricula. The biggest challenge is combining a common Bachelor's degree and specialized Master's degree. The introductory courses should give a good background for the specialized major. Tables 3 and 4 illustrate two major research areas in the light of curricula and teaching. Table 3 presents how network engineering and protocols track is built by combining the introductory courses and Master's level courses. Table 4 presents the same for service-oriented communications.

Table 3.
Track #1: Network Engineering and Protocols

CS2008 Area (Net-Centric)	ROLE
TCP-IP	Introductory Course, Bachelor
Wireless Telecommunications	Introductory Course, Bachelor
Information Networks Security	Introductory Course, Bachelor
Network Programming	Standard Course, Master
Communications Software, Protocols and Architecture	Standard Course, Master
Local Area Network - special course	Advanced Course, Master
Wireless Service Engineering	Advanced Course, Master
Secure Communications	Advanced Course, Master

Table 4.
Track #2: Service-Oriented Communications

CS2008 Area (Net-Centric)	ROLE
Communications Software Laboratory Course	Introductory Course, Bachelor
Introduction to Telecommunications	Introductory Course, Bachelor
Web-Programming	Introductory Course, Bachelor
User-Centric Service Design	Standard Course, Master
Service-Oriented Architectures	Standard Course, Master
Parallel Computing	Advanced Course, Master
Cloud Computing (not yet held)	Advanced Course, Master

The courses in the curricula are taught so that the personal skills and knowledge can be obtained. The methods have been optimized so that every instructor is aware of the methodology used in other courses and can adapt in his implementation of the course. For both tracks, one particular skill has been emphasized, i.e. programming. It is fundamental skill of a Communication Software engineer. It has to present in each course. The programming language may vary but each course contains some programming tasks, whether done alone or in a group.

In this paper, the good experiences of both learning results and spirit of learning in Code Camp concept are being integrated into curricula. The concept should be firstly (gradually) applied in some introductory courses (bachelor courses), e.g.. Web programming and TCP/IP and thus Code Camp as a method is familiar and provides support as "the learning philosophy". The concept could easily be applied for example to group works of these courses meaning that these group works are done together at some specified time. As students then progress and enrol to the next course, e.g. network programming, the concept is applied to a new context, yet the learning philosophy remains

the same. And then when the student further progresses to advanced courses, the learning philosophy is still intact. Thus students inherently learn to cooperate and learn those social skills various curricula proposals emphasize.

Table 5 shows a plan (not yet carried out) how Code Camp concept can be integrated into a period long course (7 weeks). The concept is scaled in time and in task settings. The first three weeks are focused on the design; the next two mixed with design and implementation, and last two focuses on the implementation. Each week lectures and exercised have embedded introduction to the week theme and how to proceed / operate within that week. Each week has also demonstrations etc. if necessary to enable the project and group work to go forward. The students are divided into groups of four. In each phase (group work) the leader is changed inside the group. Also other roles inside the group should be changed. Now every student has to be in every role (project manager, evaluator, documenter and communicator) they are supposed to be. In addition to the group work, the students have a personal track, in which they practice under same theme but with different methodology, e.g. in GR1 the students do Context of Use study and in P1 everyone collects a diary.

Table 5. Integration of Code Camp concept into Period long course. (critical thinking (CR), problem solving, design & modelling (DE), methodology skills (ME), programming (PR), project work (PW), written & oral presentation (WO), foreign language skills (LA), evaluation and feedback skills (EF), leadership (L), negotiation (N), communication & interaction (I), group work (G), and supervision (S)

Course Timeline	Theme / Context (User-Centric Service Design)	Skills obtained / practiced	Code Camp Integration
Week 1	Understanding Users and the Context of Use	CR, ME, PW, WO, LA, L, N, G	Introduction lectures, Context of Use Exercise with Code Camp concept **GR1 document (checkpoint)** **P1 document (checkpoint)**
Week 2	Understanding the Service Characteristics	CR, ME, PW, WO, LA, L, N, G	Introduction lectures, Service Characteristics Exercise with Code Camp concept **GR2 document (checkpoint)**
Week 3	Building Scenarios	CR, ME, PW, WO, LA, L, N, G	Introduction lectures, Service Characteristics Exercise with Code Camp concept **GR3 document (checkpoint)** **P2 document (checkpoint)**
Week 4	Building Service Front-End	CR, DE, ME, PW, PR, LA, L, N, G	Introduction lectures, Demonstration of Tool as in a Code Camp concept
Week 5	Evaluating Service Front-End	CR, DE, PW, ME, LA, EF, L, N, I, G	Introduction lectures, Demonstration of Tool as in a Code Camp concept
Week 6	Implementing Service	PR, PW, LA, L, N, I, G, S	Introduction lectures, Demonstration of Tool as in a Code Camp concept
Week 7	Implementing Service Front-End	PR, PW, LA, L, N, I, G, S	Introduction lectures, Demonstration of Tool as in a Code Camp concept **GR4 document (checkpoint)** **P3 document (checkpoint)**

CONCLUSIONS AND FUTURE WORK

The Code Camp concept has been successfully applied to various programming courses from 2003 on. The intensive nature of these courses has made it easy for industry

to participate in these courses. As a result both university instructors and students have benefited of the arrangements and the learning results have been better than in traditionally lectured courses. In addition to the technical skills and knowledge the Code Camp concept has emphasized those often forgotten social skills. In this paper we have considered if the concept could be applied to the whole curricula not only to a single course. Both course and curricula level actions are needed as Code Camp concept has so far been applied only to pure programming courses. An plan is given how the concept could be used in user centric service creation course. The plan is in early development phase, and needs further studies, e.g. for particular skill set required or how Code Camp can cope with the problem of individual assessments for work performed in team. If this kind of analysis is done for each course in a curricula, those skills emphasized in e.g. ACM/IEEE recommendations could be better taken into account. Now that we have defined the tracks for our major we need to work on the analysis of the whole track to utilize the best parts of the Code Camp concept.

REFERENCES

[1] Aken, A. and Michalisin, M., The impact of the skills gap on the recruitment of MIS graduates Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: SESSION: IT workforce preparation St. Louis, Missouri, USA, 2007, pp. 105 – 111, ISBN: 978-1-59593-641-7

[2] Computer Science Curriculum 2008: An interim Revision of CS2001, ACM and IEEE, http://www.acm.org/education/curricula-recommendations

[3] M. J. Gallivan , D. P. Truex, III , L. Kvasny, Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising, ACM SIGMIS Database, v.35 n.3, p.64-87, 2004

[4] H. J. Nelson, A. Ahmad, N. L. Martin, C. R. Litecky , A comparative study of IT/IS job skills and job definitions, Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, 2007, pp. 168-170

[5] C. L. Noll and M. Wilkins, Critical Skills of IS Professionals: A Model for Curriculum Development, Journal of Information Technology Education, Volume 1 No. 3, 2002, pp. 143-154.

[6] Ikonen J., Hämäläinen H., Alaoutinen S., Heikkinen K. and Porras J.,From Tacit to Acknowledged Knowledge, 20th Eaeeie annual conference, June 22-24, 2009, Valencia, Spain ISBN: 978-84-8363-428-8.

[7] Porras J., Ikonen J., and Heikkinen K., Code Camp – A setting of collaborative learning of programming, International Journal of Technology for Advanced Learning, Acta Press, 2007, 1/2007, ISSN 1710-2251.

[8] Computer Science Curriculum 2008: An interim Revision of CS2001, ACM and IEEE, http://www.acm.org/education/curricula-recommendations

[9] Information Technology Curriculum 2008: Curriculum Guidelines for Undergraduate Degree Programs in Information Technology, http://www.acm.org/education/curricula-recommendations

ABOUT THE AUTHORS

Assoc. Prof. Kari Heikkinen, D.Sc. (Tech), Department of Information Technology, Lappeenranta University of Technology, Phone: +358-62111, E-mail: kari.heikkinen@lut.fi

Assoc. Prof. Jouni Ikonen, D.Sc. (Tech), Department of Information Technology, Lappeenranta University of Technology, Phone: +358-62111, E-mail: jouni.ikonen@lut.fi

Prof. Jari Porras, D.Sc. (Tech), Department of Information Technology, Lappeenranta University of Technology, Phone: +358-62111, E-mail: jari.porras@lut.fi