Better programming skills through Code Camp approach

Jari Porras, Jouni Ikonen, Kari Heikkinen, Kimmo Koskinen, Leena Ikonen

Lappeenranta University of Technology
P.O.Box 20, 53851 Lappeenranta
Tel.+358-5-62111; Fax.+358-5-621 2899; E-mail. firstname.lastname@lut.fi.

Abstract

In this paper we illustrate the use of a Code Camp approach in teaching programming skills. The Code camp approach as an intensive and a social way of learning programming can be seen as a viable alternative to the traditional exercise based approach. Our experience is based on two separate implementations of the code camp method, a 24h and a one week long experiment. Approach is analyzed through extensive questionnaire. Based on the results of the questionnaire, the majority of students see the code camp a better approach than the traditional method. Therefore, it seems highly probable that the Code camp approach will be used extensively in the future for teaching programming skills.

1. Introduction

Programming is definitely one of the basic skills of students as well as professionals in computer science. Though one would not work as a programmer, programming is one of the assets in teaching logical thinking required in the field of computer science. The ability to programme is based on the knowledge of data structures, algorithms as well as programming languages and environments. Regardless of the computer science field, e.g. information processing or data communications, these basic aspects need to be known. The basic skills are usually taught in some very basic courses, e.g. data structures, introduction to programming and programming with C. The contents of these courses are quite similar from university to another, only the methods of teaching may vary. [1-7]

This paper concentrates on code camp as a method of teaching programming (chapter 2) and presents implementations of code camps held at the Lappeenranta University of Technology, LUT (chapter 3). The impact of code camp approach is evaluated through answers to extensive questionnaires (chapter 4). Finally some conclusions are drawn.

2. Methods to teach programming

Traditionally, programming is taught by having lectures and exercises on the selected topics as well as some practical works. Lectures are used for familiarizing students to the topic, exercises for practical training, and practical works for testing what the students have learned. Quite often a single course takes from one period to half a year. The duration of the course is not the main issue but the amount of work required from the students. It is important that students make enough exercises since the

practice is the only way to become a master in the field of programming.

Basic courses in programming are quite often mass courses with hundreds of students. Some approaches have been developed to ease the teaching burden on lecturers and assistants [e.g. 8]. The VIOPE virtual programming learning tool (see Figure 1) has been used to teach the C language to first year university students at LUT for the past three years. VIOPE contains theory, example codes, multiple-choice questions, and programming assignments. The system compiles and tests the student's code, and lets the student proceed after each programming task is done. In a mass course with hundreds of students the VIOPE frees a lot of teaching resources from tedious homework checking. Motivated students learn by writing their own codes in VIOPE and getting instant feedback, for example compiler error messages in a more understandable form.

Figure 1: Screen capture of VIOPE

However, there are problems with the virtual learning approach, for example copying codes made by the others. The fact that a machine and not a teacher checks the codes might make it seem less risky to submit a code made by somebody else. The system itself does not compare the students' codes against each other, so teachers are responsible for detecting plagiarized codes. This year some 15 people out of 300 tried to submit a copy of the same code. This was detected by the teachers and respective response was launched. Another problem is that if students only use VIOPE, they do not learn to compile, test and debug codes in a real programming environment. Fortunately there exist several other courses where the students have the possibility to practice these skills.

In the traditional approaches to programming the time for the work is quite long compared to the amount of work done. In this half a year period the student is quite alone especially if considered that at the end of the course practical works are done independently. In our VIOPE approach plagiarism is reduced by very strict rules and active assistants. This might in some cases decrease the learning results as it is impossible for students to cooperate while working on the problem. In some cases students are even penalized if cooperating.

The code camp approach tries to improve the learning process of programming by concentrating on two aspects, *efficient time usage* and *cooperation*. Many participants would refer to code camps as experiences. The term *camp* refers to situation, where participants get together and live a while together. The term *code* refers to coding, i.e. writing programs. Thus, during a code camp participants write programs together, solve problems related to their work, eat and even might relax in sauna. Intensive time together gives opportunity to work on ideas without interruptions from other tasks, and with the possibility to interact with other people working on the same situation. In fact, the code camp is an intensive and cooperative approach for learning programming skills. By emphasizing the social aspects learning can be done in a more meaningful way.

3. Realization of code camp in LUT

Three different code camps have been arranged in LUT in the past two years. Different types of experiments have been run. Twice within the annual summer schools of the department (http://www.it.lut.fi/wawc) a 24h code camp has been organized, and due to good results of the code camps, a one week practical course on coding during this spring was carried out.

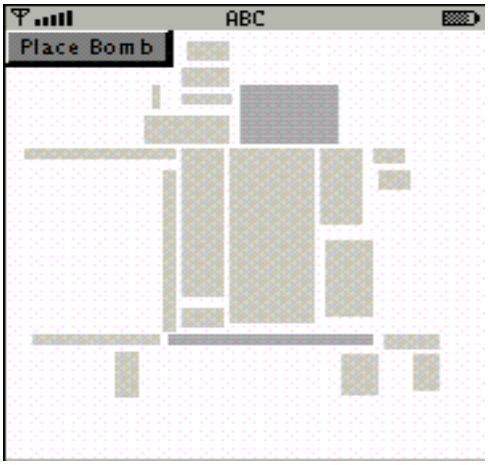

Figure 2: Bomberman-game

The first code camp was arranged in the 12th summer school on telecommunications in Lappeenranta as a practical part of the summer school. Symbian programming was selected as the topic of this code camp. Students were given a few hours of lectures and demonstrations concerning Symbian and the programming environment before the code camp. Students formed groups of three persons and started their own projects. All of the groups managed to produce a simple game working on the Symbian platform. In the 13th summer school on telecommunications the code camp was arranged for the second time. This time the Nokia MUPE (Multi User Publishing Environment) platform [9] was selected as the programming environment and location aware mobile games as the topic of the code camp. Once again groups of three students were created and appropriate introduction of the environment was given. This time cooperation was even better than in the first code camp, and all groups managed to produce a working location aware game (see example in the Fig.2) for the MUPE platform.

These 24 hour code camps have been very challenging, as topics have been chosen so that the students have had very little experience with the selected topics. This is due to the fact that summer school offers something extraordinary for the participating students. However, this type of an approach has required a closely planned program for teaching. Code camps have started with a short 1-2 hour introduction to the subject by experts from the selected field. After that the participants have moved to the *hands on* phase, where they have given example program codes, which they have compiled and run to get to know the used environment. Normally students have started from the example codes and made their own realizations by extending and applying them on new domains. As participants are new to the subject, there have been experts on the topic available all the time to answer their questions and help them if they have problems.

One example of a 24h code task is writing an application for a mobile phone which uses the Symbian operating system. Students have to learn what are the tools for writing software for mobile phone, how Symbian is coded, how the applications are tested, and moved from the simulator to the actual cellular phone. This has been very rewarding for students, as they have in a short time been introduced to a world, which is completely new to many of them.

Due to the successful implementation of location aware applications in the 2nd 24h code camp, a separate code camp style course on programming was launched this spring. Once again the MUPE platform was

selected as the programming environment (due to its simplicity as an environment, and earlier experience of the teachers) but location awareness was extended to context awareness. Location was one of the contexts offered for the students. There were two possibilities for the location information, namely cell based location (according to WLAN access point) and exact location (according to several WLAN access points and Ekahau positioning engine). In addition to location information, also weather information (temperature, wind), heart beat and exercise information (speed, acceleration) from a bicycle were offered for the students.

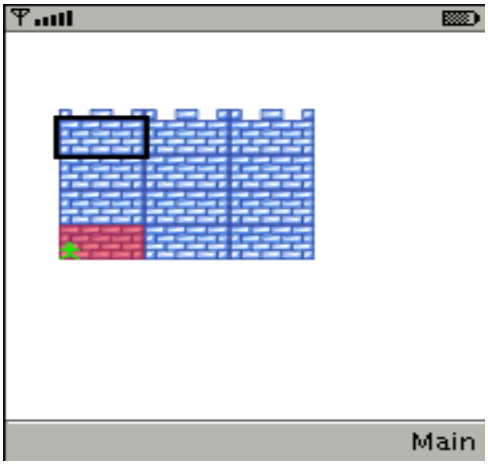

Figure 3: Game prototype in the week-long Code Camp course

Gaming was selected once again as the application area of the code camp. This time the length of the code camp was extended to one week. This was due to the fact that this intensive course was held at the same time with other courses and only evenings and nights were reserved for this purpose. Students had some other courses going on during the day time. As the time was extended, also the number of lectures was increased and students were offered a possibility to pass either the theoretical or the practical part, or both. The theory part contained the lectures and a design exercise of a game as well as an exam. In practical part the students designed and implemented their context aware game (see example in Fig. 3) in the MUPE environment.

4. Evaluation of CODE CAMP approach

Altogether 91 students enrolled into the week long code camp based course. Half of them decided to take the theory option, and approximately half of them implemented their games. One third of the participants made both parts. After this week long code camp students were asked 54 questions concerning the code camp approach. Questions were roughly divided into the following categories:

- Background questions
- Code camp as an approach for teaching programming
- Implementation of the code camp
- Contents of the theory and the practical part
- Programming environment
- Concluding questions

The questionnaire was answered by 58 students. 27 participated in the code camp part. 70% of the students thought that their background was only moderate (2/5) for this type of a course. It is highly probable that this question was interpreted as a question concerning the contents rather than programming, as the averages of the programming skills of the participating students was good (3/5) or better for 90% of students. This lack of background information was expected, as the topic for the course was selected in such a way that it offered something new for the students.

Students were asked to evaluate the code camp as an approach for teaching programming with strict numbers and with free words. 75% of the students liked this approach more than the traditional approach, and 90% considered it better or

equal. Thus, at least students felt that this approach is working. Students graded the teaching methods (lectures and the code camp) with the grade 3,6 out of 5 which is quite a good result. Students were also asked to list the biggest advantages and weaknesses of the code camp approach:

- Strengths of the approach consist of *intensiveness* and *social aspects*
- Weaknesses were *strict timetable*, *lot of new information in short time* and *physical stress*.

It seems that the students appreciated the social aspects of the code camp. Students felt like they would have been working on a real world project where cooperation helps them to achieve their goal. Intensiveness of the course was both strength as well as a weakness. For students it was easier to concentrate on the topic during short time but that resulted in strict deadlines, difficulties in adopting all the new information, and finally physical stress (due to the last night of working while finishing the games).

Students were also asked questions concerning the duration and partitioning of the course. 40% of the students felt that the course was too short. Only 3% would have managed with less. We also asked about the amount of the work done during the course (course included lectures, exercises, game design documents, game implementation documents and the exam). 60% of the students said that the number of credits, i.e. 3, and the hours of work used were in balance. The average time used for this course was 50 hours, which is much less than the 120 hours required for the given number of credits. However, the intensive and physically stressful, as well as the socially open study environment let the students learn more than in traditional course.

The implementation of the course was evaluated based on the opinions concerning the contents of the course. 80% of the participants liked the topic of this course. Especially presenting the game design patterns in the beginning of the course was successful. The use of external lecturers was a good decision. Only the linking of different topics received critics as the students did not see the connection between some lectures. This was mainly due to the students participating in the theory part, as all the others had to work with all parts. The decision of partitioning the course into two separate parts could have been reconsidered. As whole students liked the course and wished that this type of courses would continue.

Students evaluated their programming skills before and after the code camp course. In general, the skills were only moderate. The basic programming was evaluated good (3,3/5) whereas special skills like game applications, context awareness or XML/J2ME were only moderate (1,5/5). After the course the students felt that their skills did improve 0,5 - 1 level in each of the special skills thus resulting on average good skills (3/5).

One of the main issues in code camp is the social aspect. Things are learned together, and experienced programmers help less experienced students. This is supposed to create a good code camp spirit. Unfortunately only 45% of the students felt the code camp spirit like it was supposed to be. In the previous 24h code camps the code camp spirit was very strong, and it was surprising that this part was unsuccessfully transferred to the one week long code camp. One of the reasons was that due to large number of student the practical work was done in two separate locations thus breaking the groups into two parts. Only the last night gathered some compliments as that time the students really worked together to finish their works.

5. Conclusion

This paper presented the code camp approach used in teaching programming. This approach is very practical as it decreases the possible fear of programming. This paper evaluates the code camp approach based on an extensive questionnaire made for the students that participated in the last week long code camp.

According to the questionnaire the code camp approach was very successful in teaching programming to the students. Students learned both basic programming skills as well as special skills concerning game development for the mobile devices. Also the context awareness was introduced to the students in a meaningful way. 82% of the students felt that they would participate in the next code camp also. This possibility is closer than they could imagine. (http://www.it.lut.fi/ssotc/).

References

[1] A. Ghafarian, Teaching design effectively in the introductory programming courses, Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference, Roanoke College, Salem, Virginia, United States, Pages: 201 – 208, 2000

[2] R.S. Lemos, Teaching programming languages: A survey of approaches, Technical Symposium on Computer Science Education, Proceedings of the tenth SIGCSE technical symposium on Computer science education, Pages: 174 – 181, 1979

[3] P. Maheshwari, Teaching programming paradigms and languages for qualitative learning, ACM International Conference Proceeding Series, Proceedings of the 2nd Australasian conference on Computer science education, The Univ. of Melbourne, Australia, Pages: 32 – 39, 1997, ISBN:0-89791-958-0

[4] M. de Raadt et al., Introductory programming: what's happening today and will there be any students to teach tomorrow?, ACM International Conference Proceeding Series, Proceedings of the sixth conference on Australian computing education - Volume 30, Dunedin, New Zealand, Pages: 277 – 282, 2004

[5] C. Zilles, SPIMbot: an engaging, problem-based approach to teaching assembly language programming, Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, Pages: 106 – 110, 2005, ISSN:0097-8418

[6] J. Bennedsen and M. Caspersen, Revealing the programming process, Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE technical symposium on Computer science education , St. Louis, Missouri, USA , Pages: 186 – 190, 2005 , ISSN:0097-8418

[7] M.C. Carlisle at al., RAPTOR: a visual programming environment for teaching algorithmic problem solving, Technical Symposium on Computer Science Education , Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, Pages: 176 – 180, 2005, ISSN:0097-8418

[8] E. Vihtonen, S. Alaoutinen, A. Kaarna, ”Computer supported learning environment for C programming language”, Proceeding of the First Annual Finnish/Baltic Sea Conference on Computer Science Education, Kolin Kolistelut-Koli Calling Conference, 19th-21st October, 2001. pp. 27-32.

[9] MUPE Platform, htpp://www.mupe.net.