Department of Computer Science
Series of Publications A
Report A-2013-6

Creativity-Supporting Learning Environments: Two Case Studies on Teaching Programming

Mikko-Ville Apiola

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV, University Main Building, on August 23rd, 2013, at noon.

University of Helsinki
Finland

Supervisor
Dr. Matti Lattu, Dr. Tomi Pasanen, University of Helsinki, Finland

Pre-examiners
Professor Lauri Malmi, Aalto University, Finland
Associate Professor Henrik Hansson, Stockholm University, Sweden

Opponent
Professor Erkki Sutinen, University of Eastern Finland

Custos
Professor Esko Ukkonen, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.Helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-10-9030-1 (paperback)
ISBN 978-952-10-9031-8 (PDF)
Computing Reviews (1998) Classification: K.3.2
Helsinki 2013
Unigrafia

Creativity-Supporting Learning Environments: Two Case Studies on Teaching Programming

Mikko-Ville Apiola

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
mikko.apiola@helsinki.fi

PhD Thesis, Series of Publications A, Report A-2013-6
Helsinki, August 2013, 62 + 83 pages
ISSN 1238-8645
ISBN 978-952-10-9030-1 (paperback)
ISBN 978-952-10-9031-8 (PDF)

Abstract

It is known that students' learning approaches, types of motivation, and types of self-regulation are connected with learning outcomes. It is also known, that deep learning approaches, self-regulated learning, and intrinsic types of motivation are connected with creativity. However, in computing pedagogy there is a lack in empirically grounded analyses in integration of the varying educational theories to build learning environments that support creativity. The literature of programming education proposes a variety of theoretical, as well as practical viewpoints in relation to the teaching and learning situation. However, little effort has been put on understanding cultural and contextual differences in pedagogy of programming. Literature shows that education is highly context dependent, and that educational design should account for contextual differences. In programming education, the nature and implications of those differences are hitherto unclear.

In this study, the paucity in research about creativity-supporting learning environments in computing education, and about contextual differences in the pedagogy of programming are addressed through two case studies. In the first context (C_{UH}) of this study (Department of Computer Science, University of Helsinki, Finland), a method of learning-by-inventing was designed and integrated into a robotics-based programming class, and its effects on students' learning were investigated through qualitative analysis of 144 interviews. In the second context (C_{TU}) of this study (IT Department, Tumaini University, Iringa University College, Iringa, Tanzania) a number of interventions for supporting intrinsic motivation and deep approaches to learning were designed, and their effects on students' learning were studied through qualitative and quantitative methods, and a controlled research setup. In addition, a mixed methods study about contextual factors, which affect the learning environment design was conducted.

In context C_{UH}, the results show that the provided environment supported the learning of creative processes through a number of mechanisms. In general, the provided environment was shown to facilitate changes in students' problem management approaches, and extended students' deep and surface learning approaches to computer science related problem solving and problem management. In context C_{TU}, the results reveal that students face many similar challenges than students in other educational contexts, and that the standard learning environment does not offer enough support for gaining the requisite development. Learning is also hindered by many contextually unique factors. Testing a model where students work on their homework under guidance, facilitated by active student-teacher collaboration did not result in significant advantage over the control group. However, the qualitative results about guided environments were exclusively positive.

In context C_{UH}, the analysis suggests that learning of creativity may be facilitated by supporting deep learning strategies, intrinsic motivation, and self-regulated learning through utilizing a combination of open learning environment configuration, learning-by-inventing, and robotics as the vehicle for learning. Secondly, the analysis suggests challenges in context C_{TU} to be addressed through increasing the number of practical exercises, by selecting the proper amount of guidance required in the learning environment, and by implementing educational action research as a standard component into the learning and teaching environment.

Computing Reviews (1998) Categories and Subject Descriptors:
K.3.2 Computer and Information Science Education

General Terms:
Design, Experimentation, Human factors

Additional Key Words and Phrases:
Creativity, Motivation, Learning approaches, CS1, Capstone Courses

Acknowledgements

Thank you to the following people who helped with this book and its eventual realization: Esko Ukkonen, Matti Tedre, Matti Lattu, Tomi Pasanen, Lauri Malmi, Erkki Sutinen, Nella Moisseinen, Markku Hannula, Jari Lavonen, and Henrik Hansson. I also want to thank my parents, friends, other family members, and especially my wife Saila.

List of Original Publications

This thesis consists of the present introduction and the following five original research papers denoted in this thesis as P1..P5, printed in their original form. The contributions of the present author are detailed in section 4.4, p. 44.

P1: Apiola, M., Lattu, M., and Pasanen, T.A. Creativity-Supporting Learning Environment—CSLE. *ACM Transactions on Computing Education*, 12(3):11:1–11:25, July, 2012.

P2: Apiola, M., Tedre, M. Deepening Learning through Learning-by-Inventing. *Journal of Information Technology Education: Innovations in Practice*, 12(01):185–202, July, 2013.

P3: Apiola, M., Tedre, M. Towards a Contextualized Pedagogy for Programming Education in Tanzania. *In Proceedings of 2011 IEEE Africon Conference.* Livingstone, Zambia, September, 2011.

P4: Apiola, M., Tedre, M. New Perspectives on the Pedagogy of Programming in a Developing Country Context. *Computer Science Education*, 22(03):285–313, September, 2012.

P5: Apiola, M., Moisseinen, N., Tedre, M. Results From an Action Research Approach for Designing CS1 Learning Environments in Tanzania. *In Proceedings of 2012 ASEE/IEEE Frontiers in Education Conference.* Seattle, WA, USA, October, 2012.

Contents

List of Original Publications **vii**

1 INTRODUCTION **1**

2 THEORETICAL BACKGROUND **7**
2.1 Approaches to Teaching Programming 7
2.2 A Framework for Supporting Creativity 13
2.3 Conceptualizing a Learning Environment 17
2.4 Students' Actions in a Learning Environment 23

3 METHODS AND DATA **29**

4 RESULTS **35**
4.1 Overview of the Articles 35
4.2 Results (Context C_{UH}) 37
4.3 Results (Context C_{TU}) 40
4.4 Contributions of the Present Author 44

5 DISCUSSION AND CONCLUSIONS **47**
5.1 Future Suggestions 49

References **53**

Chapter 1

INTRODUCTION

Creativity and Computing Education

A range of research has been conducted in the area of computer science education. There are studies from pedagogical viewpoints including motivation, cognition, and multiple learning theories, as well as a collection of experience reports making practical suggestions, and observations related to the learning and teaching situation. Some of the trends in the recent decades of computing education suggest the inclusion of student-centered practices such as problem-based-learning (PBL) or inquiry learning (IL), which are argued, among other things, to support intrinsic motivation and deep approaches to learning. Contrasting evidence on alternative approaches have spoken in the favor of more traditional, teacher-driven learning practices. Research about contextual or cultural factors of the learning environment is also inconclusive. Despite the large amount of research, there is no coherent understanding about how pedagogical approaches work in different courses, contexts, cultures, and continents in computer science education.

In universities, computer science is being taught in a myriad ways. Some curricula may focus for example in programming, algorithmic thinking, and problem-solving skills, while other curricula may, for example, emphasize contextually relevant practical skills. While some learning environments utilize student-centered pedagogies, others may prefer alternative pedagogical approaches. In some cases, educators prefer tangible instruments, such as robotics tools in the knowledge construction process, while others may prefer starting from abstract ideas, and utilize visualization platforms to support the learning process. Different learning environments are constructed from a combination of intended learning outcomes, pedagogical approaches, types of educational resources and educational technology, and grading criteria. Each type of curriculum, and each taught topic promotes a specific view of the learning environment, and defines the required skill set of the teacher.

In general, learning objectives of tertiary education have been categorized in multiple well-justified ways. Learning objectives may range from ones that are narrowly defined and typically easy to measure–such as learning of factual knowledge or specific skills–to those that are more broadly defined and difficult to measure–such as improvement of activating and self-reguled learning skills, and improvement of creative abilities (Entwistle, 2007). One categorization is based on intended levels of understanding (Biggs, 1979).

A survey study (Joy et al., 2009) reviewed major computer science education journals and conference series including over 3500 papers in all, and found that most articles focus on technical descriptions with often very little evaluation from an educational perspective (Joy et al., 2009). It has also been argued, that computing education papers generally lack in having a well justified educational perspective (Randolph et al., 2005). Many papers do not address educational issues, but are based on reports of tool development or of use of technology in the classroom (Joy et al., 2009). Some have argued, that:

> *"too much of the research in computing education ignores the hundreds of years of education, cognitive science, and learning sciences research that have gone before us. We know that student opinions are an unreliable measure of learning or teaching quality."(Almstrum et al., 2005)*

Computer science is a relatively new discipline, and the identification of appropriate strategies to teach the diverse topics it includes remains open to debate. Even though student-centered, problem-based pedagogical approaches are becoming more common in computing education, in general they are still rare in comparison to more traditional instruction (O'Grady, 2012). It is widely accepted, that understanding how to support deep approaches to learning, creativity and intrinsic types of motivation is a globally important challenge in all disciplines of education. The main motivation for this study comes from a lack of studies about understanding creativity, intrinsic motivation, and learning approaches in the context of computer science education.

In the broad scheme of things, the challenges and problem types of computing are constantly growing more complex. Computer science is increasingly penetrating into other fields of life, which will further expand the problem types posed to computing professionals. In the future, computing professionals will need to cope with problems of a wider variety, and problems, which will require innovative and multifaceted ways of problem solving. From this viewpoint, broadening students' skills from working with basic types of problem solving into focusing also in creativity and real-life complexity is a global challenge, too.

Framing the Research

The factors, which affect learning in different contexts, countries, cultures, and continents may differ a lot, but finding ways to support motivation, deep approaches to learning, and creativity can be considered as globally important. Whilst there is an impressive array of research (Trigwell et al., 1999) on deep approaches to learning (Marton and Säljö, 1976), creativity (Csikszentmihalyi, 1996, Mumford, 2003), and intrinsic motivation (Niemiec and Ryan, 2009)—their embodiments in modern creativity-supporting learning environments, as well as their implications to design of novel learning environments of computer science, are hitherto poorly understood. The abundance of competing theories on learning and motivation makes the design of new learning environments a daunting task. How does one combine ideas from educational theories–such as self-determination, problem discovery, creative problem solving, cognitive development, and threshold concepts–so that their combination supports creativity, intrinsic motivation and deep approaches to learning? The paucity in studies on exploring the support for creativity, intrinsic motivation and deep approaches in modern learning environments of computing education leads to the broader research aim for this study, which is set as follows.

How does one provide support for creativity, deep approaches to learning, and intrinsic motivation in teaching of computer programming and software development?

In this research, the strategy to address this aim is divided into several actions. Firstly, addressing the aim is seen to require a theoretical model about how to provide a learning environment supportive of creativity, intrinsic motivation, and deep approaches to learning. Second, the theoretical framework needs to be put to practice and researched. In this study, the second phase was conducted in two educational contexts. The first case study was conducted in the context of computer science undergraduate program of the University of Helsinki (denoted in this study as context

C_{UH}), and the second case study was conducted within a newly initiated information technology program: Tumaini University's BSc in IT Program in Iringa University College, Iringa, Tanzania (denoted in this study as context C_{TU}).

Roughly speaking, the research progressed as follows: based on consideration and comparison, a selection of central theories of creativity, motivation, and learning approaches was done, upon which a theoretical framework was built. In the first case of study, turning the theoretical framework to practical arrangements included the usage of LEGO®Mindstorms as a platform for students' learning. Preliminary visions for the experiment included providing an open learning environment, which would grant students with more freedom over their problem-discovery, problem-solving, and problem management processes. The concrete plans included arranging experimental courses in the University of Helsinki (context C_{UH}), and conducting educational research on the courses from the perspective of a number of central learning theories.

In the second case of this study (context C_{TU}), the theoretical framework was utilized to improve and analyze the learning environment of computer programming courses in Tumaini University, Tanzania. Because the educational and sociocultural context was foreign, and because programming had been considered as the most challenging topic both for the students and the former teachers in that educational context (Tedre et al., 2011, Tedre and Kamppuri, 2009), it was seen necessary to conduct an additional study (as compared to context C_{UH}) for gaining understanding of the contextual challenges in teaching and learning programming. The visions for modifying the learning environment in context C_{TU} included the increase of student-centered activating exercises, and the decrease of instructivist lectures.

Thesis Structure

This thesis consists of the present introduction, and the five original research papers (cited as Paper I, II, III, IV, V) that are refereed international journal and conference publications. In order to explore the research phenomenon outlined in this thesis, each paper contributes to the increased understanding on improvement of learning environments of computer science.

Paper I provides a general framework for a learning environment supportive of creativity, intrinsic motivation and deep approaches to learning in computer science. The framework is developed in context C_{UH} (University of Helsinki, Department of Computer Science, Finland.) Paper II deepens the understanding of students' behavior in a learning environment, which is based on the framework described in Paper I. Papers III and IV explore contextual elements, which affect the learning environment design in basic programming courses in context C_{TU} (Tumaini University's IT-programme in Iringa, Tanzania.) Paper V extends the understanding of the learning environment in context C_{TU} by studying the impact of guidance for homework practice, and by exploring the students' study approaches.

By following this theme, the structure of the present introduction has been modeled upon the research papers in the following way: Chapter 2 introduces background studies in relation to teaching computer programming and software development (section 2.1), this thesis' theoretical stance on creativity (section 2.2), a model for designing computer science learning environments from the teacher's perspective (section 2.3)[1], and theories to understand students' actions in learning environments (section 2.4). Chapter 3 introduces the research methods and data, followed by overview of the results in Chapter 4. Finally, Chapter 5 provides discussion of the results, and concludes the thesis.

[1]The learning environment model presented in section 2.3 partly offers such new contribution to the thesis, which is not presented in the articles.

Chapter 2

THEORETICAL BACKGROUND

This chapter introduces the theoretical background for this thesis. Firstly, a survey of recent approaches for teaching computer programming and software development is presented (section 2.1). Secondly, a theoretical basis for supporting creativity is presented (section 2.2). Thirdly, a conceptual framework for designing and analyzing learning environments is presented (section 2.3). Finally, a collection of theories for understanding students' behaviors in a learning environment is presented (section 2.4).

Computer science consists of three intertwined traditions (Denning et al., 1989). The theoretical tradition deals with verifiable theoretical structures, such as algorithms, data structures, and their properties. The engineering tradition aims at working implementations, products, and inventions. The scientific tradition aims at finding causalities and generalizations based on models, theories, and laws (which in turn derive from empirical observations and measurements). The traditions upon which different curricula and courses are rooted bring great variation to each curriculum's problem types and to suitable pedagogical approaches (Tedre and Sutinen, 2008).

2.1 Approaches to Teaching Programming

This section presents an overview about computer science education research focusing especially on introductory computer programming courses (generating novice programmers), and on software development courses (turning novices into experts).

Introductory programming (Generating novices)

Programming education is a widely researched and intensely discussed topic. A working group of McCracken et al. (2001) investigated the programming competency of students, which had just completed their CS1 and CS2 courses. Several universities participated in the study with a combined sample of 216 students from four universities. The disappointing results revealed, that many students did not know how to program at the end of their introductory courses with an average score of 22.89 out of 110 points on an evaluation criteria designed for the purposes of the study (McCracken et al., 2001). Many studies report, that learning programming poses great challenges to many students (see for example: Robins et al. (2003), McCracken et al. (2001), Lister et al. (2004)). One popular explanation for the learning challenges is related to lacks in abilities to problem-solve (Lister et al., 2004).

A number of studies consider the most important aspect of programming education to be related to problem-solving skills (Pears et al., 2007, Palumbo, 1990). Those studies propose that addressing the development of problem-solving skills should be a major goal of the pedagogical design in introductory programming. One extensive literature review drew a strong connection between the learning of programming and the learning of problem-solving skills (Palumbo, 1990). The authors of that review argued that in a typical introductory course in programming, too little time is spent on practice in order to develop the necessary problem-solving skills (Palumbo, 1990). Another popular view of programming education emphasizes the learning of syntax and structure of a programming language; most introductory programming books follow this view (Pears et al., 2007).

Lister et al. (2004) studied a hypothesis that students' challenges in programming might be related to their "fragile grasps of basic programming principles and the ability to systematically carry out routine programming tasks", in a study involving students from seven countries. The results revealed, that students' abilities to carry out code tracing (or "desk checking"), and their abilities to predict outcomes of codes, and values of variables at given points of program execution were generally weak. Lister et al. (2004) suggest that *"this is because students have a fragile grasp of skills that are a prerequisite for problem-solving."* A great number of other experiments and research also exists (see for example: Pears et al., 2007). One ongoing debate centers around the appropriateness of different programming languages for teaching programming (Pears et al., 2007).

Results from studies by Soloway and Ehrlich (1984) show, that expert programmers use two types of programming knowledge: 1) generic program fragments known as programming plans, and 2) such rules of the programming discourse, which capture the conventions in programming and govern the composition of the plans into programs. It is suggested, that syntax and semantics is not enough, but instead students must be given instruction about “vocabulary terms”, such as program mechanisms, explanations, goals, plans, rules of programming discourse, and plan composition methods, which are scaffolds that expert programmers have learned to know and use (Soloway, 1986).

It is generally accepted that it might take a long time to turn a novice programmer into an expert programmer (Robins et al., 2003). The ability to understand written program code is a good starting point. However, research studies have shown there to be little correspondence between the ability to read a program and the ability to write one (Winslow, 1996, 21).

A typical way to teach CS1 courses in universities is to utilize a traditional pattern of instructional lectures, seasoned with a collection of take-home exercises, followed by a pen-and-paper exam. This is the approach adopted in most introductory programming courses and textbooks, although problem-solving, program design, and constructing an executable program have been suggested to comprise the underlying issues in learning programming (Robins et al., 2003). A number of alternative approaches exist, based on, for example, student-centered approaches (O’Grady, 2012).

There is a range of attempts of implementing a problem solving approach to CS1 courses by bringing student centered and problem based learning into programming courses (Ambrosio and Costa, 2010, Bakar and Shaikh Ab Rahman, 2005, Beaumont and Fox, 2003, Duke et al., 2000, Nuutila et al., 2005, Peng, 2010). Common in adding problem based learning (PBL)-style activities to the learning environment is emphasis put on 1) open-ended “real-world” learning tasks, 2) the changed role of the teacher from an instructor to a coach, and 3) studying in collaborative groups, 4) granting the students more control in terms of planning their studies and setting their own personal learning objectives, and 5) the extension of the learning objectives from basic programming to “life-long learning”, self-regulatory, and group work skills.

How the problem-based activities are implemented in practice and how they are researched differs on multiple dimensions. Some approaches utilize open ended projects (Ambrosio and Costa, 2010, Bakar and Shaikh Ab Rahman, 2005), while some utilize a combination of open-ended and traditional programming tasks (Nuutila et al., 2005), and some report the utilization and development of an extensive ”problem-oriented” learning material, which is intended to guide and lead the students’ thinking to the right solutions (and to learning) (Kurhila and Vihavainen, 2011, Duke et al., 2000). For example, a study (Duke et al., 2000) reports an extensive HTML-material consisting of 160 programming problems, and several hundred webpages consisting of tips and guides of varying difficulty level, while another study (Nuutila et al., 2005) introduces an approach where a combination of group work case-based discussion sessions guided by a tutor are utilized in combination with individual programming assignments, a programming project, and essay tasks. One study (Kurhila and Vihavainen, 2011) was partly based on a modification to the learning environment by adding one-on-one guidance to the homework practice-environment.

In many papers, the changed role of the teacher means the utilization of tutors and assistants in helping group work, or extensive practical sessions where students are able to drop in and out, and where assistants are available to provide support and continuous feedback in students' individual work (Duke et al., 2000, Kurhila and Vihavainen, 2011). The changed role of the teacher might also mean that there is no lecture sessions at all, or it might mean that the lectures utilize different kinds of learning practices. An example of such learning practice is the utilization of a think-aloud method, where programs are written together with students *on-the-fly* (Duke et al., 2000). In one approach (Nuutila et al., 2005) students are presented with cases, which the students examine, after which they identify the problems related to the task, brainstorm together, sketch an explanatory model, and establish their own learning goals, after which a period of individual study follows. Finally, closing sessions to discuss and combine each student's work is held (Nuutila et al., 2005). In one approach (Duke et al., 2000) practical assessment tasks are utilized instead of pen-and-paper exams.

Many of the approaches utilize only learning tasks that students must complete in groups of students (Ambrosio and Costa, 2010, Bakar and Shaikh Ab Rahman, 2005, Beaumont and Fox, 2003, Peng, 2010). Some problem-based approaches report a combination of both group and individual work (Nuutila et al., 2005), while some approaches emphasize only individual work (Duke et al., 2000, Kurhila and Vihavainen, 2011). Many approaches report granting more control to students in terms of arranging their own studies, setting their own learning objectives, and finding their own learning materials. Granting control means also posing additional learning objectives related to self-regulation, group work, and individual study. The amount of guidance and control has triggered debates, as some argue for example that too open learning environments are not suitable for novice learners (Kirschner et al., 2006).

The role of educational technology is one track of research in relation to learning introductory programming. There exists a variety of software tools designed to support learning of programming (see for example Kelleher and Pausch (2005)). One popular and well-studied tool especially aimed at program visualization in introductory programming courses is Jeliot (Ben-Ari et al., 2011). Recently, the utilization of MOOC (Massive Open Online Course) softwares have also become common as a platform for teaching CS1 courses.

Advanced programming (Turning novices into experts)

A common approach to teach software development in universities is to take the software engineering perspective (see for example: Dugan (2011)). Software engineering is the discipline concerned with the application of theory, knowledge, and practice for effectively and efficiently building software systems that satisfy the requirements of users and customers (ACM Information Technology Curriculum Committee, 2005). In software engineering, the concept of life cycle model is used to define phases, which occur during software development (Abran et al., 2004). The common set of phases include requirements analysis, design, implementation, verification, and maintenance. Examples of common life cycle models include the waterfall-model, evolutionary development, the spiral model and iterative or incremental development. Popular iterative models include for example the XP (eXtreme Programming) and Agile models.

Capstone courses are courses, which are targeted towards university students who are nearing the completion of their studies, and who have acquired the basic skills from their previous courses. The idea of capstone courses are to teach how to apply the content learned in previous courses to practice. This is often achieved through a final year, group-based software-engineering project. Alternate capstone course models found in the extensive survey study of Dugan (2011) included a research experience course (see Schneider (2002)), but research experience courses were found to be rare in comparison to a mainstream of software-engineering projects, and were often considered by educators as *"lacking the authentic experience needed by industry-oriented students".*

Teaching of software development through software engineering principles is often done by utilizing a teacher-driven lecture session followed by a practical project, which often aims at simulating a real-world engineering project (Dugan, 2011, Baker et al., 2003). One pedagogical justification is based on the argument that a good way to learn is by practicing in an environment as much similar to “real world” (a job in software industry, for example) as possible. In a number of cases (see for example: dos Santos et al., 2009, Brodie et al., 2008, Qiu and Chen, 2010) such an environment is seen as a favourable way to teach software development. On the other hand, software engineering courses have been critizised for their ignorance of learning theories, and for poor constructive alignment (Biggs and Tang, 2011) between the learning outcomes, and the learning environment (Baker et al., 2003, Chimalakonda and Nori, 2011, Armarego, 2008, Navarro and van der Hoek, 2008).

It has been argued that while learning theories have been leveraged in software engineering only in a minimal way, they actually could play a significant role in this domain (Baker et al., 2003). One hypothesis is, that following industry standard recipes and defined processes already in the academia may restrict students' possibilities to come up with ideas, explore, dwell on subjects, problems and matters of the students' own learning needs and interests. Thus, it is not well understood, how tuning for efficiency already in academia will affect the students' efficiency and creativity later on, when entering the "real world".

Common Pedagogical Trends

Since the shift from behaviorist to constructivist thinking on teaching and learning in the recent decades, student-centered, project-based, and problem-based pedagogical approaches have become increasingly common, also in the context of computer science education. Common examples of pedagogical theories, which follow the constructivist learning paradigm are Problem Based Learning (PBL), Project Based Learning, and Progressive Inquiry (see for example: (Hmelo-Silver, 2004, Jonassen, 2000, Hakkarainen, 2003, Barron et al., 1998)). One important aspect in all these pedagogical theories is that the teacher's role is switched from a behaviorist model of giving direct instruction towards acting as a coach, or a facilitator of the learning process. In addition, realistic, open-ended projects, and cases are utilized as learning tasks in contrast to fixed, closed-ended tasks.

Wide range of experiments have been reported, which aim at implementing student-centered practices into computing education (O'Grady, 2012). Currently, most studies on problem-based principles in computer science education cluster around describing pedagogical interventions, and students' reactions (opinions) about the interventions. While a lot of reports on utilizing problem-based principles in CS education exist, only a minority of studies has a more thorough educational perspective, evaluating the approaches beyond student feedback. Even though attempts of implementing problem-basedness exists in courses ranging from introductory programming to software development, it is argued, that currently the penetration and research of problem-based principles in computing education is shallow (O'Grady, 2012).

Many open debates around problem-based, and student-centered instruction are ongoing. One of the debates is that about the amount of control between the teacher and the student. Problem-based and similar open environments are typically considered to be more open, granting students more control over their learning. However, problem-based learning, for example, has been criticized for being "minimally-guided", and as such improper for certain learner groups (Kirschner et al., 2006). Other arguments have responded, that while minimally guided instruction indeed does not fit all learner groups, it is a misinterpretation to conclude problem-based learning as equivalent to minimally guided learning (Schmidt et al., 2007). Other ongoing debates center, for example, around the problem types, which should be utilized in problem-based learning approaches.

The more modern approaches vary quite much in terms of, for example, the tasks and problem types utilized (ranging for example from closed problems to open problems), which kind of classroom interaction is utilized (ranging for example from student-centered to teacher-driven), how much control, and guidance students are granted, and whether group work is utilized and how is it utilized, what is the teacher per student ratio, which kind of classroom tools are utilized. The learning objectives vary from short-term related (programming) objectives, to long-term (problem-solving, self-regulation and active learning styles) learning objectives. The definition of a learning environment may be thought to consist of sets of variables, which properties and values vary over multiple dimensions.

2.2 A Framework for Supporting Creativity

Creativity

Creativity is a widely researched, and intensely discussed concept, defining of which is, however, complex (see for example: Mumford (2003), Sternberg and Lubart (1999)). Sternberg and Lubart (1999) define creativity as *"the ability to produce work that is both novel (i.e. original, unexpected) and appropriate (i.e., useful, adaptive concerning task constraints)"*, and note that "Creativity is a topic of wide scope that is important at both the individual and societal levels for a wide range of task domains."

Creativity may also be defined as the *ability* to challenge assumptions, to recognize patterns, to see in new ways, to make connections, take risks and to seize upon change (Herrmann, 1996). The ability to be creative has been connected to a certain *working style* or *problem-solving process*, which involves a persistent process of idea generation, idea evaluation, and the ability to transfer the selected ideas to action (Jackson and Shaw, 2006, 89-108). Csikszentmihalyi (1996) argues, that *"creativity occurs when a* *person, using the symbols of a given domain such as music, engineering, business, or mathematics, has a new idea or sees a new pattern, and when this novelty is selected by the appropriate field for inclusion into the relevant domain."*

There are a multitude of viewpoints to creativity (see for example Mumford (2003)). However, for the purposes of this thesis, creativity is understood from the viewpoint of several popular studies (Amabile, 1983, Csikszentmihalyi, 1996), according to which, creativity requires the simultaneous presence of three components: intrinsic motivation, certain cognitive processes and working styles, and domain-relevant skills. Highest levels of creativity may be found from where these three components overlap the most (Amabile, 1983). Those three components of creativity are briefly introduced in the forthcoming subsections.

Intrinsic Motivation

Intrinsic motivation is defined as the motivation to engage in an activity primarily for its own sake, because the activity is perceived as interesting, involving, satisfying, or challenging (Amabile, 1987, Ryan and Deci, 2001, 2000b). In contrast, extrinsic motivation is defined as the motivation to engage in an activity primarily in order to meet a goal extrinsic to the work itself, such as attaining a reward, winning a competition, or meeting some external reward such as recognition. One study (Amabile, 1983) proposed a hypothesis that *"the intrinsically motivated state is conductive to creativity, whereas the extrinsically motivated state is detrimental to creativity."* The concept of intrinsic motivation has gained a lot of interest in educational psychology, and in addition to being beneficial for creativity, it has been argued to be a favorable condition in itself, for example for learning (Niemiec and Ryan, 2009).

According to several studies, intrinsic type of motivation is connected to deep approaches to learning, while extrinsic types of motivation connect with surface learning approaches (Marton, 2005, Fransson, 1977). There again, intrinsic motivation has been identified as one crucial component in creativity (Amabile, 1987). One study (Amabile, 1987) introduced the phenomena by using a "maze metaphor", in which the creative problem-solving process is represented using the metaphor of a maze with various exits representing different kinds of solutions to a problem. Extrinsically motivated straightforward, algorithmic, or step-by-step solutions are represented by a straight path from the entrance to the exit. More unusual or creative solutions require intrinsic motivation and thus can be reached only by taking a more heuristic approach and exploration of the problem space (the maze) (Amabile, 1987)).

According to some studies, extrinsic rewards, such as positive evaluations or other awards prior to performance, seem to create extrinsic motivation (Amabile, 1987, Amabile and Collins, 1999). On the other hand, if a task is constrained or controlled, it has been argued to result in reduced autonomy, and thus, reduced intrinsic motivation (Amabile and Collins, 1999). The perceived level of autonomy and freedom are related to higher levels of intrinsic motivation, where for example competing for prizes to be offered for best products may restrict intrinsic motivation, and also creativity (Amabile and Collins, 1999). The self-determination theory (SDT) (Ryan and Deci, 2000a) argues that intrinsic motivation can be supported by supporting its three forming factors: autonomy, competence, and relatedness.

In the context of higher education, it has been argued that intrinsic motivation can be supported by promoting a feeling of autonomy (in contrast to a feeling of being controlled), promoting the feeling of relatedness (in contrast to the feeling of isolation), and supporting the feelings of competence (in contrast to the feelings of incompetence) (Ryan and Deci, 2001, 2000b).

Cognitive Processes

The required cognitive process of creativity may be defined as a process, which involves the generation of multiple ideas, and the ability to select the good ideas from the pool of available ones. Thus, the process involves persistence in idea generation and idea evaluation. Finally, the good ideas need to be transferred to action (Jackson and Shaw, 2006, pp. 89-108).

The cognitive process of creativity is sensitive to both internal and external barriers. It seems that the *type of problem* is related to the required cognitive process: well-defined problems may not require a creative cognitive process to be solved, but *open-ended* problems should be used instead. The problem should also pose enough, but not too much *challenge*. Other generally acknowledged enhancing factors for the creative process are *time for incubation* (Sio and Ormerod, 2009), and a *positive mood* (Davis, 2009). It is argued, that the environment should be psychologically safe.

Thus, previous research about the cognitive processes of creativity has identified a number of factors, which are linked with the process. Those factors include the problem types, challenge level, incubation time, mood, and psychological safety. Connecting with other learning theories, the cognitive process of creativity is linked with deep approaches to learning (Marton and Säljö, 1976), which have further been researched to connect with epistemological positions (Perry, 1970), conceptions of learning (Marton et al., 1993), and other properties of the learner such as attitudes, or orientations.

There exists many pragmatic methods proposed to support creative work. Most of the methods are based on the thought that creativity requires an environment that encourages risk-taking (it does not for example reward for simple but working text-book solutions), and self-initiated projects and provides help and time for developing ideas and individual effort. Some of the methods introduced in the literature include brainstorming (Osborn, 1963), verbal check-lists (Eberle, 2008, Osborn, 1963), picture stimulation and mind mapping (Buzan, 1991), and 3+ (Lavonen and Meisalo, 2009)[1]. A general idea in these methods is the purpose of supporting idea generation by suppressing the common tendency to criticize or reject ideas, delete old ways of thinking and encourage new kinds of mental associations using different types of games or tasks.

Domain Relevant Skills

A person must be exposed into the domain in question, and must posses the domain relevant knowledge and skills to be able to add to that specific domain. *"No matter how enormous mathematical gifts a child may have, he or she will not be able to contribute to mathematics without learning its rules"* (Csikszentmihalyi, 1996). Further on, even if the rules are learned, the domain must recognize and legitimate the novel contributions (Csikszentmihalyi, 1996).

Domain-relevant skills are seen as one essential requirement for creativity (Amabile, 1987, Amabile and Collins, 1999). For example, to be able to compose creative music, one has to hold preliminary skills in music. Or, if one is to publish creative results in the domain of science, it is necessary to master things such as scientific research methods, and domain-relevant previous research. In the domain of software development, programming skills are one domain-relevant prerequisite. In learning introductory programming, part of the domain relevant skills are related to skills, which are a prerequisite for the programming-related problem-solving abilities (see for example: Lister et al. (2004)).

Synthesis: a Framework for Promoting Creativity

It is now possible to combine a framework for supporting creativity in computer science higher education (Table 2.1). The framework is combined together from five components. The three first components (competence,

[1]For many others see (Smith, 1998, Higgins, 1994).

Table 2.1: Conceptual Framework for Supporting Creativity

Component		Method of support
Intrinsic motivation	*Autonomy*	Provide choice and opportunity for self-direction
	Competence	Use creativity-enhancing methods, provide effectance promoting feedback
	Relatedness	Promote interaction with creative working methods (games and plays)
Domain-relevant skills		Support learning to recognize one's own skills, and learning needs
Cognitive Processes		Use creativity-enhancing methods: brainstorming, 3+, and open-space workshops Support deep approaches to learning Encourage risk-taking and exploration

autonomy, relatedness) derive from intrinsic motivation research. The other two required components are domain-relevant skills (DRS) and cognitive processes and working styles (CP). The table lists these main components, and general guidelines for supporting each component in the learning environment.

2.3 Conceptualizing a Learning Environment

This section defines a model through which a learning environment may be defined and analyzed. A learning environment is a combination of teaching practices, physical surroundings, learning tasks, and assessment practices. A learning environment provides scaffolds for a student's learning trajectory—a path that a learner takes to accomplish learning goals (Dron, 2007, pp. 61-70). The learning environment may generate destructive friction in cases, where the environment is too strictly or too loosely structured in relation to a student's self-regulation skills. Constructive friction emerges from a proper amount of shared control between the teacher and the student (Vermunt and Verloop, 1999). The learning environment should activate the student's zone of proximal development (ZPD) (Vygotsky, 1978).

Emotions and motivation affect each other, which together have an effect on performance (Pekrun, 2006). Task involvement is fostered by many emotions, and solving a challenging task often requires a range of emotions. The learning environment should promote a balance between feelings of competence and feelings of challenge (Moneta and Csíkszentmihályi, 1999). Imbalance leads to a decrease in concentration and involvement. As a rule of thumb, a too high challenge is better for concentration than a too low challenge (Moneta and Csíkszentmihályi, 1999). The organismic integration theory (OIT) argues, that intrinsic motivation is a favorable state for learning, and the learning environment can support it by utilizing a proper combination of autonomy, relatedness, and competence (Niemiec and Ryan, 2009).

The following subsections present a combination of variables in aim to understand learning environments of computing education.

2.3.1 Intended learning outcomes and assessment

An intended learning outcome defines what a student is expected to be able to do after exposure to teaching (Biggs and Tang, 2011). Intended learning outcomes may be categorized between easily definable and easily assessable "short-term" learning outcomes, such as memorizing of factual information or understanding how a certain algorithm works, to more ill-defined, difficultly assessable "long-term" learning outcomes, such as acquisition of new learning skills, self-regulation skills, problem-solving skills, deep approaches to learning, or active learning skills, for example. In a study, researchers found out that it took years for change in learning styles to show up in test scores (Lonka and Ahola, 1995). Short-term learning outcomes are typically more straightforward to measure.

The assessment tasks should be in a proper *constructive alignment* (Biggs and Tang, 2011) with the intended learning outcomes, and with the teaching activities. If not, students can "escape" by engaging in inappropriate learning activities such as surface approaches to learning (Biggs and Tang, 2011, pp. 99).

One property of intended learning outcomes and assessment is *control*, which means how much control the students and the teacher have in defining the intended learning outcomes and the assessment procedures. In a very open environment, students may participate in setting their own intended learning outcomes (and assessment tasks), while in a typical university course the teacher is in full control of the intended learning outcomes and the assessment tasks. From the teacher's perspective, there is variation in how much the learning outcomes and assessment tasks are determined by institutional demands, and other background factors.

2.3.2 Learning tasks

The learning environment provides the student with certain learning tasks, i.e. problems, which the student solves in order to learn. A variety of problem types can be found in education, which may include for instance logical problems, algorithmic problems, story-problems, rule-using problems, decision-making problems, troubleshooting problems, diagnosis-solution problems, strategic performance problems, case-analysis problems, and design problems (Jonassen, 2000). Problems share a number of properties, for example they are subject-relative and context-dependent (Mills, 1956, pp. 76). Problems may also be classified according to their openness. In closed (well-structured) problems the starting point, solving technique, and goal state are known (Sutinen and Tarhio, 2001). In open (ill-structured) problems the starting point, technique, and goal can all vary from closed to open. Other problem classifications include the dimension between pseudo-problems, authentic problems, and ethical problems. The selection of problem types in computing education is usually highly related to the computing tradition, and to the intended learning outcomes.

2.3.3 Tradition of computing

Computer science consists of three intertwined traditions (Denning et al., 1989) (see section 2). Although the traditions are deeply interwoven, most problems (learning tasks) typically emphasize one of the traditions over the others. The traditions may be tacit within a department's ethos, and thus invisible to the teacher and the learners. Each tradition of computing determines techniques, theories, and working modalities in computing practice.

2.3.4 Problem control

Solving a problem is often only one stage in a process of solving multiple problems. Many problems raise more new problems than they solve, and thus the process of managing the solving of multiple problems is important in a learning environment. In computing education, *problem control* can be closed/controlled by the teacher (teacher gives certain tasks to solve), or it can be controlled for example with an industry-standard software engineering model (in software development courses), or it can be more open (a science-like research project, or a design-oriented software course) (Apiola et al., 2012). Other restrictions or limitations include the selection between individual work and groupwork, and the particular platform or environment where the problem is to be solved. Another factor is the width versus depth of problem coverage: the environment may provide a wide range of material to be touched only on the surface, or vice versa: a narrow range of topics with increased depth.

2.3.5 Subenvironments

The learning environment typically contains a number of subenvironments. The subenvironments may include for example a classroom environment (lectures), exercise meetings, the homework environment, and exam environment. A subenvironment may also be digital, for example a program visualization environment such as Jeliot (Ben-Ari et al., 2011). Each of the environments have their own unique characteristics, and may contain multiple factors, which enhance or restrict students' learning.

The *classroom environment* is a typical subenvironment known in higher education. Usually the classroom environment means lectures or excercise sessions where a teacher or several teachers interact with a group of students. A classic distinction between *interaction styles* is that between teacher-driven (instructivist), and student-centered (constructivistic) interaction styles. Although many opinions exist, the general trend is against instructivist teaching, which has been argued to connect with surface learning (Entwistle, 2007, Biggs and Tang, 2011), extrinsic motivation (Hoskins and Newstead, 2009), bad learning outcomes, insufficiency of stimulating higher order thinking, and low attention (Biggs and Tang, 2011). Examples of constructivist classroom interaction may include for example *peer instruction* (Mazur, 1998), think-aloud modelling, work-along excercises, concept maps, and many others (Biggs and Tang, 2011).

The amount of *control*, and *guidance* in different subenvironments may vary. Examples of teacher-controlled subenvironments include lecturing and tutorials, where the teacher is highly in control. Environments, which contain more of student participation may include peer-assisted studying, various types of group work, and various types of interactive excercise sessions. Subenvironments outside of university premises may include home-environments, libraries, and other places, which all have their own unique factors, which influence the learning situation. The home environment is, for example, typically highly individually managed.

2.3.6 Learning materials and available resources

Learning materials can be of a variety of types. Other resources available in the learning environment may include for example availability of facilities, electricity, light, computers (measured by for example guaranteed access hours), books, libraries, digital libraries, tables and other furniture, as well as availability of support and guidance from peers, teachers, and assistants.

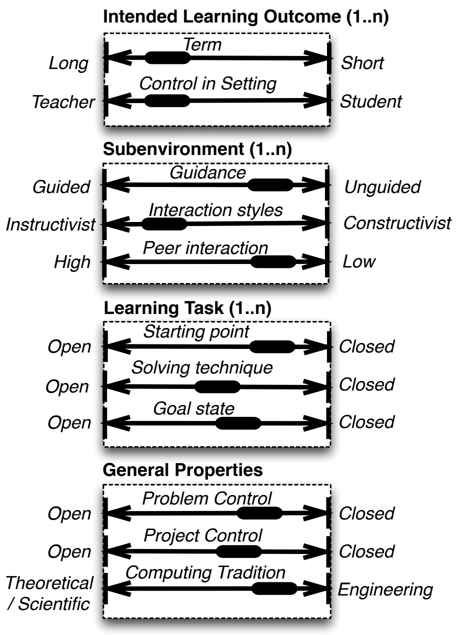

Figure 2.1: Central Variables of a CS Learning Environment

2.3.7 Control

An *open* environment grants full control to a student, while a *closed* environment gives the teacher full control over the learning situation. In an open environment, the learning is similar as visiting a marketplace: the learners interact with those market stalls (subenvironments), which fulfill their learning needs (Meisalo and Lavonen, 2000). The learner may also be in control of the intended learning outcomes, assessment, learning tasks, and even the amount of control in different environments (the student may be granted the control to choose the amount of control). If the learners are left with more control, they should be able to understand the consequences of their choices (Dron, 2007).

Synthesis: CS Learning Environments: Teacher's Perspective

The most relevant variables introduced in the above sections are summarized in Figure 2.1. In the upper part of Figure 2.1 are the intended learning outcomes, which can be set on a continuum from long-term learning outcomes to short-term learning outcomes, and which can be either controlled by the teacher or the student (as discussed in section 2.3.1). A learning environment typically contains at least one intended learning outcome.

A learning environment consists of a number of subenvironments (second section in Figure 2.1). As discussed in section 2.3.5, the amount of guidance may vary in different subenvironments from minimally guided to maximally guided. Also, the interaction styles may vary on a continuum between instructivist and constructivist, as well as the amount of peer-interaction, which may vary on a range from low to high.

Learning tasks (third section in Figure 2.1) are given to students in aim for them to achieve the intended learning outcomes. Learning tasks in computer science can be categorized on three continuums based on the openness of the starting point, solving technique, and goal state of each task (see section 2.3.2). A learning environment may contain an unlimited amount of learning tasks. Learning tasks are related to the intended learning outcomes.

Other central properties of the learning environment (fourth section in Figure 2.1) include the problem control (section 2.3.4), which can vary from open to closed, the overall control in the learning environment (section 2.3.7), and the tradition of computer science, which can vary between the theoretical, scientific and engineering traditions (section 2.3.3).

Theoretically speaking, any selection for the values of the variables can be made by the teacher. In practice, the configuration options are influenced by forces such as the characteristics of the surrounding context (sometimes denoted as the "design milieu" (Duveskog et al., 2013)), the skills of the teacher, and other factors, part of which can be hard to operationalize. The presented model can hardly be conclusive or exact: it is a well recognized issue, that in educational settings it can be considerably difficult to treat classroom settings, combined with social and psychological issues, motivation, and conceptions as independent or dependent variables (for example: Juuti and Lavonen (2006)).

In this study, the present model is utilized in several ways. Firstly, to contrast a typical learning environment for software development (see section 2.1) a new kind of learning environment is provided and researched in context C_{UH} of this study. This is accomplished by "switching" the student more control over setting the intended learning outcomes (as compared to typical learning environments in context C_{UH}), by promoting open-ended learning tasks, and by granting the students freedom in problem control. The classroom environment is also "switched" from instructivist to constructivist with high peer-interaction. Secondly, in context C_{TU} of this study, several modifications for a typical learning environment of introductory programming are inspected. Generally speaking, the classroom environment is "switched" from instructivist to constructivist by developing a variety of contextually relevant classroom pedagogies. In addition, a new configuration to the homework environment is inspected by studying the impact of increased guidance.

2.4 Students' Actions in a Learning Environment

The learning environment provides the student with problems to learn how to solve, and the student then takes an approach for solving the required problems, utilizing the scaffolds and support structures offered by the learning environment. The types of students' approaches to solving the problems may differ on a large scale.

A student's learning process is proposed to be affected by a number of factors. The student's learning process has been researched for example from the viewpoint of approaches to learning, self-regulation, cognitive processing, metacognitions, learning orientations, conceptions of learning, motivation, affect, social interaction, context, and meta-affect (see for example: Marton and Säljö, 1976, Pintrich, 2004, Lonka et al., 2004, Hannula, 2004). The process of problem solving (through which learning may partly happen) is affected with motivation, emotions, knowledge transfer, memory processes, language parsing, intellectual ability, and expertise (see for example: Kotovsky, 2003).

Two mainstream research tracks on student learning are the SAL (Students' Approaches to Learning) track, and the SRL (Self-Regulated Learning) track (Lonka et al., 2004). SAL is based on European research, while SRL is based on North-American research. SRL learning models have been criticized for being overcomplicated to be valuable for educators or educational researchers. In contrast, the SAL models have been criticized for oversimplifying learning (Biggs, 2001).

Approaches to Learning (SAL)

Research on deep and surface approaches started in the mid-1970s (Marton and Säljö, 1976), and have since been followed with a broad range of research. The surface approach to learning is described as an information-reproducing approach, while the deep approach is described as the knowledge transforming approach. Deep approaches to learning have been shown to produce better learning outcomes in comparison to surface learning approaches (Marton and Säljö, 1976). It has been shown, that a student may switch between learning approaches in different learning tasks (Richardson, 2005), and also within one study task. Switching of learning approaches has been confirmed for example among engineering students (Marton and Säljö, 1976, Laurillard, 2005).

A student's learning approach is not seen as a stable tendency of a student, but it is seen to be formed as a result of the interaction between the student and the learning environment (Marton and Säljö, 1976). It has been shown, that the choice of a student's learning approach is also affected by the student's general conception of learning (Marton, 2005, Marton et al., 1993, Van Rossum and Schenk, 1984), the student's conception of the specific learning task, and the student's conception of what is required of her (Marton and Säljö, 1976). In addition, it has been shown that intrinsic motivation generated by a non-demanding and supportive learning environment is related to deep approach to learning, and extrinsic motivation resulting from threat to self-esteem and ego-involvement is connected with surface learning approach (Fransson, 1977).

Deep and surface learning approaches have been found by a number of studies, and their existence has been confirmed among a number of study topics, for example in the domains of problem solving and engineering (Marton, 2005, Laurillard, 2005). In this study, approaches to learning were analyzed "directly" by investigating students' problem-solving approaches in relation to programming and other learning tasks, as well as through students' conceptions of learning in general, the specific learning tasks they were given, what is required of them, and openness of their learning environment. This was done because approaches to learning have been confirmed to influence students' learning outcomes.

Self Regulation (SRL)

Learning related self-regulatory behavior may be described from four dimensions: motivation/affect, behavior, cognition, and context (Pintrich, 2004). According to Pintrich (2004), self-regulatory activities follow a time-ordered sequence consisting of making plans, setting goals, monitoring, controlling, reacting and reflecting. However, there is no strong evidence about the time-order, and thus different phases may also operate in parallel, and dynamically for example in cases, where plans and goal setting activities update themselves on the basis of information received from control threads (Pintrich, 2004).

The cognition dimension represents activities and strategies for planning, monitoring and regulation of cognition. It includes activities for activating prior cognitive and metacognitive knowledge, and includes regulation of cognitive functions such as memory, reasoning, learning, problem-solving, and thinking strategies. *The motivation/affect dimension* consists of regulation of self-efficacy, beliefs, perceptions of task challenge, and task value beliefs, and the activation and utilization of different coping strategies for example in relation to dealing with negative affect such as fear and anxiety.

The behavior dimension includes management and planning of time and effort. This includes allocating time, making schedules, capability to control effort and persistence, and help-seeking behaviors. *The context dimension* includes activities for modifying the context. The context is often restricted by the learning environment, but for example in some student-centered classrooms students are encouraged to gain more control for example by designing their own learning tasks (Pintrich, 2004).

In this study the SRL framework is operationalized by looking at students' reports on coping strategies, metacognitive knowledge, positive and negative affects, and time and effort regulation. This was done because self-regulation behaviors have been studied to have an impact on learning outcomes.

2.4.1 Properties of the Learner

A certain kind of behavior (deep or surface learning approach, coping strategy, self-regulation mechanism, motivational state) is a combined result from the interaction process between the student and the learning environment, and a more general tendency to behave in a certain manner. For example, deep and surface learning approaches are seen as resulting from the interaction between the learner and the environment (Marton and Säljö, 1976). However, more stable tendencies of behavior will have their own influence for the learning activities performed by the student. Those properties may include *learning orientations, conceptions, previous skills, personal interests*, and *personality variables.*

The direction and domain of the orientation may vary, for example students may have certain orientations towards their studies in full, but may have different kind of orientations towards specific study topics, courses, study methods, or learning situations. The orientations may develop and change during studies, and they can be seen acting as mediators between contextual background factors (see section 2.4.2), and the actual study approaches (see previous section 2.4).

Learning Orientations

Learning orientations are more stable tendencies to act in certain ways, and may provide explanations to, for example, which kind of course or topic preferences the students have. Study orientations can indicate how students attribute the basic meaning of their studies, which will influence essentially setting of goals, planning, organizing, and approaches to learning in different learning tasks (Lonka et al., 2004).

Students' general study orientations have been categorized for example into three dimensions: the utilizing, internalizing, and achieving orientations (Biggs, 1979). The utilizing orientation is well resonated with surface approaches to learning, and it is characterized by extrinsic motivation in terms of avoiding failure, minimum effort, and syllabus-boundedness. The internalizing orientation is well resonated with deep approaches to learning, and it is intrinsically motivated and syllabus-free; student studies beyond the requirements and beyond the topic. The achieving orientation revolves around winning, and it utilizes a systematic approach for gaining highest possible grades using both deep and surface approaches, whenever appropriate (Biggs, 1979).

Other studies, which were based on inventories to look at university students' more general approaches to learning are for example the Approaches to Studying Inventory (API), Revised Approaches to Study Inventory (RASI), Approaches and Study Skills Inventory for Students (ASSIST), the Inventory of Learning Strategies (ILS), Inventory of General Study Orientations (IGSO), and the Reflections on Learning Inventory (RoLI) (Lonka et al., 2004). All the inventories of student's approaches to learning are more or less based on Marton and Saljo's (Marton and Säljö, 1976) original distinction between surface and deep approaches. For example the IGSO (Inventory of General Study Orientations) (Mäkinen et al., 2004) has repeatedly produced the following scales representing students different orientations towards their studies: the deep-, anxious surface-, achievement-, systematic-, work-life-, practical-, social-, and lack of interest-, orientations (Lonka et al., 2004). Another study categorized study orientations to academic, work-life, and non-committed (Mäkinen et al., 2004).

Ylijoki (2000) identified one main disciplinary "tribe" within computer science students of a certain department. That "tribe" was described as professionally or industrially oriented, emphasizing hard expertise and respect for pragmatic skills. That orientation was seen to be influenced by an institutional moral order and culture (Ylijoki, 2000). In this thesis, students' learning orientations were looked "directly" by looking at students' views on their larger goals in relation to their studies, as well as through students' reports on their failure avoidance, adherence to syllabus, amount of effort, and emphasis of achievement. This was done, because learning orientations have been studied to have an impact on learning outcomes.

Conceptions of Learning and Intellectual Development

Conceptions of learning started as a topic of study based on an assumption, according to which the students' perceptions of learning tasks reflect their past experiences of similar situations, and in that way mirror the differences in students' preconceived ideas about what it takes to learn (Marton, 2005). Marton et al. (1993) found the following six qualitatively different conceptions of learning: *increasing the quantity of information, memorizing, acquisition of facts and methods, abstraction of meaning, interpretive process aimed at understanding reality, and changing as a person.* Similar conceptions of learning have been confirmed to exist by a number of research studies worldwide (Richardson, 1999).

The students' general conceptions of learning have been suggested to represent a time-ordered developmental hierarchy, although there is not yet strong evidence to support this claim (Marton, 2005). However, this hypothesis has received support from comments of informants who did mention a process of transition between the conceptions (Marton, 2005), as well as observed similarities between the conceptions of learning with Perry (1970)'s longitudinal study of students' intellectual development. Students conceptions of learning have been found to correlate strongly with students' deep or surface approaches to learning (Van Rossum and Schenk, 1984).

In addition to general conceptions of learning, research studies have confirmed that situation-specific conception about learning affects students' approaches to learning (Marton and Säljö, 1976). Those studies have resulted in quite strong evidence, which indicates that the type of testing affects students' approach to learning. This can also be called the "backwash effect", which means that students adopt an approach, which is determined by their expectations of how their learning will be tested (Marton and Säljö, 1976). For example, in the context of problem solving, students who concentrated more on what is expected of them were more likely to adopt surface approaches to learning (Laurillard, 2005).

This study analyzed students' conceptions of intellectual development through the six conceptions of learning described by Marton et al. (1993), as well as through their notions of the backwash effect (students learn what they think they will be tested on). This was done, because conceptions of learning were, based on earlier research, assumed to have an impact on students' motivation, which in turn has been studied to impact learning outcomes.

2.4.2 Background Factors

Background factors may include complex factors related to a country's educational system, sociocultural variables, and a wide range of other factors, which may indirectly or directly have an influence on students' learning. The curricula sets quite much demands for a student. In addition, there are a whole lot of informal and implicit demands regarding studies, the so called hidden curricula of a university or a faculty, an ethos of the department which is often not visible (for example: Bergenhenegouwen, 1987). A student's study goals and the hidden or visible institutional demands may collide, which may result in the generation of attitudes. There is a line of research on study cultures in different academic disciplines (for one example, see: (Ylijoki, 2000)).

Social forces may influence the situational learning approaches through mediating learning orientations, or according to some studies, also directly: social forces form a very important aspect of fixation and negative transfer (Pretz et al., 2003). Peers, culture, and language play a major role in all phases of the problem-solving process (Pretz et al., 2003). In addition, group's standard practices may have become entrenched so deeply that they impede changes in the group members' ways of thinking (Pretz et al., 2003). For example the familiarity or unfamiliarity to solve a certain type of problem, or to work in a certain kind of learning environment is affected by previous education, culture, language, and group work issues. These, in turn, may influence the learning process through mechanisms such as fixation and negative transfer.

Complex socialization processes include the generation of social norms, attitudes and values, which may later be internalized to one's personality system. They may further form the basis for interests, preferences, motivational orientations, conceptions, learning approaches, and other properties of the learner (Lonka et al., 2004). Other background variables, which have at least been shown to indirectly influence students' study orientations include parents' educational background, cultural capital, and family interaction in childhood (Lonka et al., 2004). Background variables may also include "rules" for interaction, such as the *power distance* between teachers and students, cultural norms for classroom interaction, and individualistic versus collectivistic study habits, and culturally acceptable dress-codes.

In this study, background factors were inspected through looking at group work issues, students' background education, and variables of the sociocultural context. These variables were selected, because they are hypothesized to have an impact to students' learning outcomes.

Chapter 3

METHODS AND DATA

Educational research can roughly be divided into two categories: research about education, and research for education (Juuti and Lavonen, 2006). While the former (sometimes denoted as basic education research) has an intellectual objective, the latter (sometimes denoted as applied education research) has a more pragmatic objective to improve learning and teaching praxis. This study utilizes a combination of basic and applied education research.

The aims of this study were of two kind: to design new learning environments, and to explore how they work: i.e. to gain insight and understanding about students' learning in two new (as compared to typical learning environments in the corresponding contexts) types of learning environments. Similar to educational design-research, the project started from a situation where one recognizes that there is something problematic in the learning environment, and neither researchers nor teachers know exactly how to act in the prevalent circumstances (Juuti and Lavonen, 2006).

Two different educational contexts were selected, because teaching is context-dependent by nature (see for example (Wiliam, 2008)), and the inclusion of two different kind of contexts was presumed to richen and widen the scope and the results of the study. There exists a low amount of studies in developing country contexts related to computer science and information technology education, which led to the selection of the developing country context as the second context for the study.

In this study, one of the challenges was that the contexts of research and the phenomena under study (student learning, and the learning environment) are very much intertwined. Those characteristics led this study to be designed as an action research (Noffke and Somekh, 2009) study. More specifically, the study utilized a mixture of research approaches typical in educational research including case study procedures, qualitative content analysis, mixed-method studies, as well as a controlled experimental research setup.

Case study principles were selected, because they are seen to fit situations where the boundaries between the phenomenon and context are not clearly evident (Yin, 2003, pp. 13), where the aim is in-depth exploration of, e.g. an educational program (Creswell, 2009, pp. 13), and where the meta-question is "what can be learned from a single case?" (cf. Randolph, 2008, pp. 53). The case study strategy is commonly seen to suite education research well (Stake, 1995, ch.10). Qualitative content analysis, mixed-method studies, and controlled experimental setups are widely used tools in multiple situations of educational research.

This thesis contributes to the following research tasks:

1. To increase the understanding of supporting the learning of creativity (including intrinsic motivation and deep approaches to learning) in tertiary level computer programming and software development education

   > "Creativity is a topic of wide scope that is important at both the individual and societal levels for a wide range of task domains (Sternberg and Lubart, 1999)". This applies also to higher computing education, where there is a need for research about creativity and innovation friendly instructional approaches. In many contexts of higher computing education, there is a lack of research about interventions to support students' learning of creative processes.

2. To increase the understanding of contextual factors, which influence the design of learning environments of computer programming

   > Pedagogical literature shows that educational design should account for differences in the ways of learning and teaching between industrialized and developing countries, and between different educational contexts in general. In educational research, both contextual understanding and knowledge in educational theories are necessary preconditions (Wiliam, 2008). However, little effort has been put on understanding cultural and contextual differences in teaching programming and software development.

The activities included in this research consisted of working as a teacher and educational researcher in two one-semester experimental courses at the Uni-

Table 3.1: Collected Data

Study (Paper)	**I**	**II**	**III**	**IV**	**V**	*Total*
Individual interviews	144					144
Group interviews			2*5			2
Classroom observation	yes	yes	yes		yes	yes
Homework observation					yes	yes
Surveys				30	25	55
Controlled research setup					yes	yes
Study transcripts		yes				yes

versity of Helsinki, Finland (context C_{UH}), where a data consisting of 144 hours of transcribed interviews, combined together with observation notes was collected. The second case included a 1,5 year period working as an assistant lecturer at Tumaini University's Iringa College in Iringa, Tanzania (context C_{TU}), being in charge of a number of programming courses including introductory programming, advanced programming, programming projects, artificial intelligence, and programming in C. The second case included collecting of a variety of data through surveys, interviews, observations, and an experimental research setup.

Data

The bulk of collected data included in this research is listed in Table 3. For the purposes of Papers I and II, a qualitative data containing 144 student interviews (1 hour each), combined with classroom observations was collected. Additionally, for the purposes of Paper II, students' study transcripts were inspected. For the purposes of Paper III and IV, a data consisting of two group interviews (5 students each) was collected, amplified with classroom observations, and a survey data with 30 participants. For the purposes of Paper V, classroom observations, homework observations, a survey data with 25 participants, and data from a controlled experimental research setup were collected.

To be more specific, Papers I and II included three different data collection methods to collect three sets of data: semi-structured interviews, observation notes made by the researchers during and after each learning session, and students' study transcripts. Sampling for interview data was a comprehensive sample, and it consisted of 72 initial interviews and 72 concluding interviews. All dropped-out students were reached for the concluding interviews, too. The interviews were semi-structured, and in the initial interviews students were asked to broadly describe their studies, including their personal interests. In the concluding interviews students were asked to describe, in detail, their learning process throughout the course. The interview protocol was tested with two randomly selected computer science students before the actual interviews. Interviews were tape-recorded and transcribed, and for the purposes of this study, quotes were translated from Finnish to English.

Data saturation point was met roughly halfway the interview data after which no new theoretical constructs were encountered. However, as some theoretical constructs described in Papers I and II that were looked for were not encountered at all, the full data set was analyzed. A number of phenomena outside the theoretical constructs were also identified: those are described in more detail in Papers I and II. The same procedure was done on the observation notes. Study transcripts were analyzed by looking at students' phase of studies, grade point averages (GPA), and course preferences (previous courses taken).

Papers III and IV included three different data collection methods to collect three sets of data: semi-structured interviews for teachers, semi-structured group interviews for students, and a survey for students. Six previous teachers of programming were interviewed either face-to-face or electronically. The students' data were collected through conducting two group interviews consisting of the best five students from the programming class from the semester, which started in autumn 2010, and five other randomly chosen students from the same course. The first group of students consisted of four males and one female, and the second group consisted of three males and two females. The interviews were conducted at an office at the university campus. Each interview took approximately 2 hours of time. All interviews were built around the themes "Biggest obstacles for learning programming", and "Ways to improve teaching of programming".

For the purposes of Paper IV, the central themes arising from the interviews and research literature were combined, and from those themes a structured survey was generated. The survey contained 59 items on a seven-point Likert scale (*1=not at all true, 2=very little true, 3=slightly true, 4=moderately true, 5=quite true, 6=very true, 7=completely true*). The survey was administered to $N = 50$ second year IT students, with a response rate of 60% ($n = 30$). As for demographic data, the gender of the sample distributed over 19 males (63.3%) and 11 females (36.7%), with an average age of 29.9 years (min 20, max 45). As previous education, 63.3% (19 students) had received secondary school education, 23.3% (seven students) had received a diploma in education, while 6.7% (two students) held a diploma in some other field than education.

The interviews and the following survey were designed to be applicable to all computer programming courses, although the answers reflected especially the experiences from the Programming II course, which the interviewed students had just completed. The interview data were analyzed by two researchers (the authors of Paper IV) independently by identifying the central themes raised by the students during the interviews.

Paper V included three types of data: observation data, which was collected by the participating teacher-researchers by writing detailed notes during guided practices sessions. Secondly, a survey study was conducted among the students. The survey asked 12 questions, which measured the students' perceived utility of the different components of the provided learning environment (course materials, classroom interaction, unguided practice, and guided practice). The survey also collected demographic data. The impact of guidance in the homework environment was inspected by utilizing a controlled pretest-posttest research setting.

Chapter 4

RESULTS

4.1 Overview of the Articles

Paper I (context C_{UH}) answers the research questions "How creativity can be supported by opening the learning environment", "How does the students' experience of the provided learning environment reflect the provided theoretical base", and "How does a robotics kit work as a vehicle for students' learning in the provided learning environment". The paper studies a contextually unique configuration of the learning environment (see Figure 2.1), which is based on the theoretical framework for creativity (see Table 2.1), and where the learning environment is generally set as very open. In that learning environment the classroom interaction is mostly constructivistic and inclusive of peer-interaction, and the overall problem control is open, and aimed towards inventions. The paper looks at the collected data through it's theoretical base of creativity including aspects of intrinsic motivation, cognitive processes and working styles, and domain relevant skills. The paper also looks at the data for implications of the effects of the robotics kit to motivation and learning approaches.

Paper II (context C_{UH}) broadens the research conducted in the learning environment described in Paper I by answering the research questions "Which kind of learning approaches do students undertake in an open learning environment", and "Which factors of the learning environment support, and which undermine different choices between students' learning approaches". The paper looks at the collected qualitative data through a number of central learning theories.

Firstly, the study analyzes students' approaches to learning through students' conceptions of four things: learning in general, the specific learning tasks they were given, what is required of them, and openness of their learning environment. Secondly, the study analyzes intrinsic motivation (as defined by the self-determination theory SDT) through looking at students' reports on their conceived level of autonomy, their feelings of competence, and their reports on collaborational (relatedness) aspects of their study work. The study analyzes students' conceptions of intellectual development through their conceptions of learning, as well as their notions of the backwash effect (students learn what they think they will be tested on). Students' learning orientations are looked through students' reports on their failure avoidance, adherence to syllabus, amount of effort, and emphasis on achievement.

Paper III (context C_{TU}) studies practical approaches based on the theoretical framework of creativity (see Table 2.1) for improving the learning environment of programming courses. Firstly, the learning environment was opened (see Figure 2.1) by reducing the amount of instructivist lecturing, and by adding class time dedicated to practice, in which the role of the teacher was changed to that of a coach or a facilitator. Secondly, affective support for students was promoted. Thirdly, *coding-while-lecturing*, a contextual adaptation of a pedagogical method was developed, where the process of writing and desk-testing programs is constantly demonstrated by programming on-the-fly, while at the same time reflecting on the process and encouraging the students to reflect also. While desk-testing the program, students were repeatedly queried about the state of the program and about the values of the variables. Fourthly, the role of exercises and practice was pressed.

The collected qualitative data is analyzed through looking at students' reactions and opinions towards the positive aspects as well as the downsides of each of the interventions in relation to learning.

Paper IV (context C_{TU}) utilizes group interviews and quantitative surveys, and studies several crucial elements, which may affect the learning and teaching of computer programming in context C_{TU}. The paper answers the following research questions: "What are the students' and teachers' perceptions about improving programming education" and "How do students view themselves as learners." The paper analyzes the data through issues related to intrinsic motivation, deep versus surface approaches to learning, self-regulation, study-orientations, and background factors.

Paper V (context C_{TU}) centers around the amount of guidance given by the learning environment. The focus of study in this paper is in the effect of granting more guidance to the students' homework environment (see Figure 2.1). The study reports on developing and testing of a model where students work on their homework under guidance, facilitated by active student-teacher collaboration, continuous feedback, and student support. The research methods included observations, student feedback, and a controlled pretest-posttest experimental research setup. The research questions for the study are "What are the main factors that influence a student's problem-solving process when working with practical exercises" and "What is the impact of guided exercise sessions to learning outcomes." The results revealed issues related to intrinsic motivation, approaches to learning, self-regulation, study-orientations, and background factors.

4.2 Results (Context C_{UH})

This section provides an overview of the main results in Papers I and II.

In general, the studied learning environment was found to be different compared to learning environments typically provided for learning software development in context C_{UH}. The data shows, that the learning environment directed students towards an experimental learning approach, and that the learning environment was in many cases supportive of intrinsic motivation. The learning approaches were found to connect with students' orientations towards their studies, and to students' conceptions of learning as well as students' previously learned problem-solving as well as problem management approaches. For example, some students were not completely ready to assume the increased amount of control offered by the learning environment, but would have wanted to give the control back to the teacher. In other words, they were unable to cope with the openness and criticized the lack of definitive objectives and technical guidelines. The students' social interaction and reflection patterns were new compared to other courses.

The research data shows, that the toolkit utilized as the platform for students' work (LEGO®Mindstorms) affected students' motivation in a couple of ways. Firstly, students' initial motivations to enter the course were attributed to the novelty value of the robotics kit, as well as to a mental association with playfulness and freedom connected to the robotics kit. In addition, students attributed the motivating effects of the robotics kit to it's tangible aspects: the effect of a program code can be perceived with one's own senses, the robot actualizing in "real life", communicating and moving within it's environment. When working with the robotics kit, motivation was found to be affected by the robotic kit's good suitability for an explorative working approach, and it's provision of multiple kinds of computing problems with varying challenge levels. Overall, the projects as well as their subproblems were often more challenging than students had first anticipated. Students' sense of ownership was increased by the fact that they were lent their own robotics kits, and they were given freedom in all phases of their work.

In relation to learning orientations, the results show that students have topic orientations among the various computing topics. In relation to their present studies, the data indicated topic orientations of students' towards three main classes: *theoretical and scientific* topics, such as algorithms, mathematics, physics, and theory of computing; *pragmatic* topics, such as software engineering and interaction design; and *applied* topics with interest towards arts, humanities, and social sciences. Variation was found in how fixed the orientation was, and whether the orientation was distinctly in one class, or spread among the classes. In other words, some students had a clear view of their topic orientation, while some showed interest towards topics across the three classes.

The data also revealed traces of motivational components as properties of students' orientations. For example, most students reported an extrinsic component in studying, involving motives such as growing as a professional and obtaining qualifications for the labor market. The intrinsic component involved interest in the actual computer science-related study topics, and interest in working on own extra-curricular projects. Characteristics of both deep and surface approaches to learning arose from the research data, which confirmed that both approaches are adopted in the provided learning environment. Deep approaches to learning were found to connect with intrinsic sources of motivation, while surface learning approaches did connect with extrinsic motivation.

This learning environment required the students to search and select the problems, which they wanted to work with. The required problem discovery process was found to include a number of challenges. The challenges included, for example, how to circumscribe the problem, how to update the task's problem space, and how to continuously evaluate the appropriateness of one's problem solving strategy to the problem at hand. Many times, these activities were found to require additional self-regulation activities in comparison to those required in other courses. Another issue was connected with the perceived difficulty of problems. In many cases, students found that problems, which they had selected turned out to be more complex than they had prima facie looked like.

Table 4.1: Two Identified Problem Management Approaches (Paper II)

Serialistic Approach	**Exploratory Approach**
Linear and iterative	Cyclic and free-moving
Serialistic	Holistic
Risk-averse	Open to risks
Industrial by nature	Hobbyist by nature
Closed-ended	Open-ended
Outcome-oriented	Experiment-oriented

On one hand, these aspects of students' actions in the learning environment seemed to facilitate deep approaches to learning, and they seemed to push learners towards the zone of proximal development (ZPD) (Vygotsky, 1978). On the other hand, cases where students were not able to update their problem spaces accordingly, or were in general unable to cope with the openness, often resulted in decreased intrinsic motivation. This in turn resulted in increased surface learning approaches.

The data exposed a continuum of approaches related to students' general problem management processes; the way the students treat the management of a set of problems. The students' problem management approaches were found to form a continuum from *serialistic* to *exploratory*. The serialistic problem management approach treats the set of problems with a fixed set of linear steps to be followed, with similarities for example with the waterfall software engineering model. To contrast the serialistic approach, the exploratory problem management approach was found to be a more organic, flexible, exploratory, and iterative process. In practice, students made prototypes, they flexibly jumped back and forth between designs, constantly switching prototypes, and models, making comparisons of different solutions, and performing scientific experiments.

In Table 4.1, central characteristics of the identified two problem management approaches are presented. The problem management approaches were found to be attached to students' problem discovery, problem selection, as well as their problem solving approaches. Since the learning environment is built upon open problem discovery, problem selection, and problem solving, in this environment the exploratory approach arguably worked better compared to the serialistic approach. The two problem management approaches have clear parallels with deep and surface approaches to learning, and it seems that although the serialistic approach is extremely well suited for the industry, it does not seem to be well suited for learning. This is because the serialistic approach does not promote deep approaches to computer science related problem solving, problem management, and problem discovery, and it does not invite the student to break deep into the ZPD (Vygotsky, 1978). In contrast, in the exploratory problem management approach the learner drifts away from pre-determined models and solutions into the development of one's own problems, approaches, models, prototypes, and solutions.

The factors, which influenced students' selections of problem management approaches were, based on the data, unclear. One hypothesis is, that the problem management approach is related to the backwash effect, which has been studied to influence students' learning approaches (see section 2.4). The backwash effect means that students choose such approaches, which they think they will be rewarded for, often based on their experiences from other learning environments. However, conclusive traces of the backwash effect were not found in the data.

It was also found, that students' conceptions of the learning environment were found together with a student's topic orientation to influence motivation. In this case, the learning environment did not restrict the types of problems, but welcomed all problems if they were related to computer science. Even though, some learners perceived the learning environment for example to be exclusively a hands-on, time-intensive engineering workshop. Indeed, such topics were found to be popular. However, in the case of some students, their conceptions of the learning environment restricted them from choosing a topic of their own preference. For example, a theoretically oriented student might have dropped out of the course, caused by a perception that the course could not offer a platform for problems of her preference.

Finally, the results confirmed that the theory, which states that creativity requires the simultaneous presence of three broad components (intrinsic motivation, domain relevant skills, and certain cognitive working styles), is useful in supporting creativity also in this educational context. The results also confirmed, that creativity theory can be successfully turned into practical teaching arrangements by opening the learning environment and by utilizing creativity supporting games and plays in the learning sessions (see Figure 2.1). The resulting configuration of the learning environment is given the name *learning-by-inventing.*

4.3 Results (Context C_{TU})

This section provides an overview of main results in Papers III, IV and V.

The pedagogical approaches described in Paper III were built upon a number of central theoretical notions of intrinsic motivation and cognitive processes (see Table 2.1.) In concert with the central components for intrinsic motivation (autonomy, competence, and relatedness) the pedagogical approach was designed to provide social interaction, introduce optimal challenges, cut down evaluation, and move students away from beliefs that good or bad performance is caused by students' internal properties, and by providing opportunities for self-directed learning.

Conjointly with central notions of cognitive processes (deep learning, experimental learning, reflection, and contextual relativist views of knowledge), the pedagogical approaches were designed to employ constructivist models and experiments, support of reflection, and demonstration of persistent problem solving. The approach utilized ideas of open learning environments in the classroom, where students were free to work on a number of learning tasks. Affective support was designed in line with the self-determination theory by promoting competence and empowerment, and marketing the relevance of programming to multiple applications and career opportunities, drawing connection with IT-work, and emphasizing careers, which students presumably find interesting, such as web-development.

As the cognitive development levels, as well as multiple other characteristics of students in a group may vary greatly in context C_{TU}, it was found to be important to have multiple kinds of support structures for differing students. Therefore, to address the challenge of supporting the requisite cognitive development of students, a large set of practical exercises was designed. The exercises were aimed at providing intellectually challenging tasks for students at varying stages of development. The exercises ranged from extremely easy to slightly more difficult to very challenging. The data shows, that at the beginning many students reported that they had to work and struggle a lot doing the exercises, which is not a negative thing at all. The data shows, that while working with the exercises, there was a specific point when some of the students started to learn programming.

Based on theories of cognitive development, a contextual adaptation of a pedagogical tool was developed (coding while lecturing), where the process of writing programs is constantly demonstrated by programming on-the-fly, while at the same time reflecting on the process with the students. The results show, that the approach discouraged the students from taking surface approaches to learning. An additional benefit of coding while lecturing was that the teacher was able to keep aware of students' cognitive development. The results show, that students reacted to the coding-while-lecturing approach in various ways. On the one hand, the data shows, that students did enjoy the approach and were interested, often active, and intensively observant of the class. On the other hand, this kind of approach was new students, which led to some confusion for example in relation to note-taking habits. Some students considered the coding-while-lecturing approach good but the difficulty too high.

The results of Paper IV reveal, that a lack of support for feelings of competence and students' perceived relevance of programming affects motivation, which leads to generation of negative emotions. Repeatedly failed attempts to solve programming exercises were reported to decrease one's feelings of competence and increase the perception of programming as extremely difficult. On the other hand, feelings of success were reported to quickly increase perceived feelings of competence, resulting in increased intrinsic motivation, inclusive of flow experiences. Lack in prerequisite knowledge and background skills were considered as important challenges.

Experimenting (deep) and memorizing (surface) learning approaches were both found, but they were not found to be associated. Instead, students were switching between their old and new learning styles when conducting their studies in introductory programming. Also, a connection between a perceived poorness of one's programming skills, and a surface learning strategy was found, as well as a connection between a surface learning strategy and positive attitudes towards plagiarizing. The data shows, that the students as well as previous teachers were well aware of a need to supplement deep learning and problem-solving skills.

In addition, language problems were identified as an important issue by students and teachers. Also, habits of working in groups were found to cause challenges in cases, where the group work is manifested in unproductive ways. Students were found to be well aware of issues regarding group work dynamics, yet students were found to hold a protective attitude towards their own group and themselves. The results also revealed issues related to differing patterns of classroom interaction between teachers from varying educational contexts.

The results show, that many of the underlying issues (motivation, group work dynamics, need for deep learning approach, need for interaction in the classroom) are partly of the same kind in other contexts, continents, and cultures. However, the means for supporting each component may vary a lot in ways, which will in context C_{TU} partly remain to be discovered. The data confirms, that like many other things, pedagogy cannot be imported from foreign practices and ideas alone, but it is crucial to understand the local practices of teaching and learning.

The data confirmed, in this particular context C_{TU}, the argument that problem-solving skills and learning of programming are very much connected with each other. In this context, the acquisition of well-functioning learning strategies and skills seems to be affected by group work strategies, motivational profile, and issues related to study approaches. It was found, that one important aspect in learning programming seems to be the work that students are required to do on their own time, outside the instructed learning sessions. For one reason or another, in too many cases students are not able to find effective ways of working, they lose motivation, and they resort to rote-memorizing, or even plagiarism.

The results show, that a major part of the challenge can be attributed to lack of effective learning skills required in learning programming, and thus, to unfamiliarity with applicable problem-solving skills. Those students who lack applicable problem-solving skills are in risk of resorting to unproductive learning strategies, such as rote memorizing. When that method fails, some students utilize extrinsically motivated strategies, such as free riding in group work, copying from other students, and plagiarizing. It is important to note, that none of these issues are particular to this context, but can probably be found all over the world in one form or another.

Paper V reports results from an experiment, where guidance was added to students' "homework" environment. Survey data, observation data, and quantitative data from a controlled pretest-posttest experimental research setup was collected. A class of students was randomly split to half. The first group conducted their homework exercises under guidance, while the second group worked on their own. The experimental and control groups were switched half-way the study, so that everyone received equal amount of guidance during the course. The learning outcomes were frequently measured by giving both groups quizzes after the completion of each set of exercises.

The results show, that students' problem-solving processes were affected by the multiple problem types, which students faced during their work with the exercises. Examples of common problem types revealed by the study were difficulties with understanding the task description, challenges in formulating a plan for solving the problem, and syntax problems such as missing commas in the source code. The data shows, that the continuous feedback loop provided by the learning environment was effective and a necessary scaffold to overcome the above-mentioned problems that the students faced. The results also show, that the guided exercise sessions were motivating and were appreciated by the students. The results also show, that the students considered programming to be a difficult, but also an interesting topic.

The survey results show, that while students considered the guided learning sessions as very valuable, still they perceived unguided exercises to be slightly more useful for learning compared to guided exercises. Thus, it seems, that there is a role for both activities. Time management and self-regulation issues were brought out by the study. Students reported issues related to task-switching, time management and prioritizing when they work with homework from multiple courses, which affects their choices between deep and surface approaches to programming tasks. The results show, that the guided environment could directly address some of the learning barriers related to self-regulation issues, and also issues of group work dynamics by restricting the study topic to programming and by setting the group dynamics rules. However, externally given rules might have also inhibited some positive aspects of students' group work and work dynamics, which makes it unclear whether the outcomes were solely positive.

The results of the controlled research setup were found to contradict the positive observations during the guided exercise sessions. All learning tasks in the pretest condition and both posttest conditions resulted in no statistically significant difference between the performance of the two randomly assigned groups. The results show, that in the workshop situation many students started to understand certain learning tasks and perform well in them. Despite their apparent mastery of the concept and skill, in the exam, the same students failed to solve similar tasks that they had successfully completed in the workshop sessions. This might result from an imbalance in *constructive alignment* between the learning environment and the assessment tasks (Biggs and Tang, 2011, pp. 95). One explanation is that students indeed do learn equally effectively under guidance and outside of guided environments, and that the qualitative results from participant observation and student feedback were misleading. Further experiments are however needed to confirm either of the alternative hypotheses.

4.4 Contributions of the Present Author

For Paper I, the present author designed the research setting including the educational intervention together with the two other authors of Paper I. The present author conducted all data collection required for the study, as well as approximately 90% of surveying the literature, data-analysis as well as the writing.

Paper II utilized mostly the data collected for Paper I. The research design was done completely by the present author. The data analysis and writing was done in a co-operation between the authors of Paper II, in a way where the present author was responsible for approximately 90% of the work.

In Paper III, the present author contributed on designing the educational interventions and designing the research. In data collection, co-operation was received from the second author of Paper III. Data-analysis, literature survey, as well as writing the paper was conducted by the present author approximately by 75%.

In Paper IV, the second author provided valuable co-operation in most phases of conducting the research including research design, data collection, data analysis, and writing. Most part of the work, with an overall rate beyond 80%, was conducted by the present author.

In Paper V, research design, and data collection was conducted approximately 60% by the present author. Data collection included co-validation of course learning outcomes between the authors of Paper V, and collecting survey results as well as observation notes, and arrangements related to the controlled research setup. Data analysis, literature survey, and writing was done by the present author with an overall workload of 80%, leaving approximately 20% for the second and third authors.

Chapter 5

DISCUSSION AND CONCLUSIONS

In context C_{UH}, the research and design of a learning environment, based on a theoretical framework of creativity, intrinsic motivation and deep approaches to learning, combined with robotics as a learning tool, yielded a number of interesting insights into students' learning in an opened learning environment. A couple of findings, however, rise above the others.

Firstly, the learning environment together with the robotics toolkit did offer students a rich assortment of computer science related problems, which did attract a number of students of varying personal interests. Partly based on it's novelty value, the robotics toolkit also offered a powerful trigger for motivation to enter the course. Secondly, the learning environment made it possible for students to choose such project management approaches, and to focus their efforts on their preferred specific sub-problems of a project in ways, which are not always supported in typical university courses in context C_{UH}.

Several phenomena were connected to the free selection of project management, problem management, problem discovery, and problem solving approaches. From the learning environment's part, the above factors were found to have an impact on students' intrinsic motivation. On the other hand, intrinsic motivation was affected by students' conceptions of learning in computing, which confirmed results from previous research. In the best cases a student learned to deal with the extended autonomy by self-regulating the problem parameters, or by switching into an alternative problem altogether, when faced with a challenge of too high level. Thus, in this learning environment, students' were required to take deep and surface approaches to learning and problem solving, and also to problem management.

Finally, it was found that the theoretical framework of creativity (see Table 2.1) could be successfully applied into a university course by making modifications to the learning environment (see Figure 2.1), and according to the results, the model suits such purpose well. This configuration of the learning environment, which is called learning-by-inventing, stood in direct contrast to the usual practices where the teacher gives either a list of homework problems to solve, or a larger assignment at the beginning of the semester.

In context C_{TU} of this study, turning creativity theory into practice by supporting intrinsic motivation and deep approaches to learning was approached from a number of perspectives. First, a shift to an open environment was promoted by ridding a traditional model of instructivist lecturing and adding of student-centered classroom practices. Since students of context C_{TU} were used to more closed environments where the teacher is in full control, the changes in mindsets did not happen overnight. Second, changing the group work dynamics without destroying well-functioning group work spirit proved not to be an easy task. The results showed that cultural ways of working in groups offer benefits when the help, which students are willing to give each other is manifested in productive ways. However, group work strategies may also hinder learning of some group members, if a group focuses on utilizing the skills of the most competent group members to complete assignments.

Thirdly, classroom practices such as development of coding-while-lecturing practices worked well in forcing students out of rote memorizing learning approaches. However, again it was found that students need time and space to become more active and reflective in the classroom, since they are used to different modes of teaching and learning. Fourthly, an adaptable model of exercises ensured students' experiences of feelings of success. Each student was able to go hands-on and get the sense of competence and grounds for cognitive development. Fifthly, selecting the proper amount of guidance to students' homework environment proved to be an important variable, which affects the students' learning. However, although the observation notes, as well as the opinions of researchers, teachers, and students all spoke exclusively in the favor of guided learning, the quantitative results showed no statistically significant difference in the learning outcomes between guided and unguided groups.

In addition to results from practical teaching approaches, a number of contextually relevant variables, which affect students' learning were revealed by this study. Firstly, the results show, that need for supplementary teaching in deep learning and problem solving skills is an important additional learning objective in this learning environment. The issue is partly caused by background factors, such as a lack of proper science and math teaching in primary and secondary education in Tanzania. During their programming studies, students were found to be switching between their new (deep processing) and old (surface processing) approaches to learning.

Secondly, the need to increase the support on intrinsic motivation was recognized by the study. It was also revealed that the underlying factors impacting the motivation may be more complicated than a proposed dual-component model consisting of the lack of feelings of competence and a low perceived value of programming. Thirdly, group work dynamics issues were recognized as important both by the students and by previous teachers. However, students were found to hold protective attitudes towards their own group and themselves. Fourthly, language problems were identified as an important issue. Fifthly, cultural differences regarding classroom pedagogy and other teaching arrangements especially concerning foreign teachers was identified as an important issue. Sixthly, additional factors include a serious lack in availability of computers, other facilities, and resources in general.

5.1 Future Suggestions

In context C_{UH}**,** one important question for further exploration is related to learning objectives. This research showed, that students' skills in coping in a more open environment varied. All students are professionals in passing computer science courses, where problems are given by someone, they are clearly articulated, there are supportive materials available, and there is often at least one correct solution for each problem. Some students were able to cope well in a more open environment and learned to set and revise their own problems, while some students considered something to be wrong. In the future, a number of new configurations for a learning environment could be designed and researched. Those learning environments would—in the best case—encourage students to invent, to follow their own interests, and teach them to set their own learning objectives, and assessment tasks.

Teaching software development by utilizing the software engineering model is optimized to minimize risk. However, teaching to avoid risk-taking may prevent the development of creativity, since inventing involves taking risks. In addition, students often possess their own ideas and approaches to solving certain computing problems, which the learning environment should update. If the learning environment does not promote and require risk-taking with new ideas, conceptual change is not likely to happen.

In programming, this can become highly tangible. If a programmer has learned to avoid risks and knows that one's existing habits can be used to solve a certain problem, one may not be open for learning more powerful approaches. While mastering the industry standard models is a must for information technology professionals, additional learning environments could also be provided, which would give more room for the creation of novelties. That would also require that students learn more capabilities of handling uncertain situations, and that they learn more abilities to deal with the related negative affects, such as frustration.

This study has challenged the utilization of an industry-oriented software engineering model (for example: Dugan (2011)) for learning software development. Creativity does not take place only by accident but it can be supported with existing theoretical knowledge and practical models. Following a software engineering model teaches one way of managing a set of problems, which the student is supposed to solve when designing and writing computer software. In a very open learning environment, a student has the freedom to either choose a problem management approach, which he is already familiar with, or then to invent a completely new way of working. Assuming the students to find new working strategies for themselves is a lot to ask. Thus, a more rigorous introduction and hands-on training of a creative working style should be given in the future.

The approach presented in this study is related to a number of commonly known approaches such as Problem Based Learning (PBL), Project Based Learning, and Progressive Inquiry (see for example: (Hmelo-Silver, 2004, Jonassen, 2000, Hakkarainen, 2003, Barron et al., 1998)). The common denominator of these approaches is the upgraded role of the teacher. Instead of giving direct instruction and knowledge the teacher facilitates, and coaches the learning process. Realistic, open-ended cases and project goals are emphasized in the abovementioned approaches. In the context of computing education, the learning-by-inventing approach provided in this study has extended other common student-centered approaches by granting students freedom in setting their own learning objectives, and by focusing explicitly on inventing and creativity.

In context C_{UH}, this thesis proposes several options for further studies. First, better support structures for teaching of a creative working process should be explored. Second, the reasons why some students performed better and felt more comfortable in the open learning environment compared to other students poses many important research questions. The interplay between creative problem solving, a student's success in other studies (GPA), and psychological characteristics, combined with phenomena of social interaction together form a complex construction with many possible hypotheses and research questions for further investigations.

In context C_{TU}, this thesis has focused on researching the development of introductory programming education.

During the short history of developing teaching of computer programming in context C_{TU}, teaching of programming has predominately been the responsibility of foreign visiting experts. The foreigners have brought in their own pedagogical approaches and teaching techniques, as well as their pedagogical biases. While many of their teaching efforts have shown positive signs, still the present approaches have not worked very well for students of context C_{TU}. Some intentional revisions to the learning environment have been made. However, generally speaking, the current approaches have not resulted in satisfactory learning outcomes in programming courses.

Several suggestions for facing the challenges in teaching and learning programming can be made. Firstly, research efforts should be put on designing learning materials and exercise tasks, which would be aligned properly with the needs of the students. Secondly, emphasis should be put on design and research of other possible support structures and scaffolds in the learning environment. In this track of research and development acts, one important direction is the search for efficient guided environments, such as instructed exercise sessions. Such exercise sessions should utilize the top performing programming students from previous classes as teaching and research assistants, which is a widely utilized and well approved practice in universities worldwide. Other directions include studying of visualization platforms, automatic assessment, and MOOC-based interventions.

Thirdly, it is important that all development, teaching, and research acts are not isolated to programming courses only, but they must be systematically aligned with other courses of the BSc in IT graduate program. All development and research acts must be planned in co-operation with the university administration, which for example needs to accredit the steep increment of allocated practice hours in certain courses, as well as the utilization of grading methods and other arrangements, which may differ from the standard regulations of the university.

Finally, maybe the most important thing is, that the teaching acts in relation to programming courses are not sufficient alone, but must be extended with increased amount of research acts, also. This means, that basic education research is required for better understanding the educational context, as well as the students. Equally important is to add applied education research, which is essential for gaining better understanding of best practices in programming education by systematically testing the impact of different educational interventions with scientific methods.

References

Abran, A., Bourque, P., Dupuis, R., Moore, J., and Tripp, L. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK).* IEEE Press.

ACM Information Technology Curriculum Committee (2005). Computing Curricula: Information Technology Volume.

Almstrum, V. L., Hazzan, O., Guzdial, M., and Petre, M. (2005). Challenges to Computer Science Education Research. *SIGCSE Bull.*, 37(1):191–192.

Amabile, T. M. (1983). Social Psychology of Creativity: A Componential Conceptualization. *Journal of Personality and Social Psychology*, 45(2):357–376.

Amabile, T. M. (1987). The Motivation to Be Creative. In Isaksen, S., editor, *Frontiers of Creativity Research: Beyond the Basics.* Bearly Limited, Buffalo, N.Y.

Amabile, T. M. and Collins, M. (1999). Motivation and Creativity. In Sternberg, R. R. and Lubart, R., editors, *Handbook of Creativity.* Cambridge University Press.

Ambrosio, A. P. and Costa, F. M. (2010). Evaluating the Impact of PBL and Tablet PCs in an Algorithms and Computer Programming Course. In Lewandowski, G., Wolfman, S. A., Cortina, T. J., and Walker, E. L., editors, *SIGCSE*, pages 495–499. ACM.

Apiola, M., Tedre, M., Pasanen, T. A., and Lattu, M. (2012). Towards a Framework for Designing and Analyzing CS Learning Environments. In *Proceedings of FIE'12 Frontiers in Education Conference*, Seattle, WA, USA.

Armarego, J. (2008). Constructive Alignment in SE Education: Aligning to What? In Ellis, H., Demurjian, S., and Naveda, J., editors, *Software Engineering: Effective Teaching and Learning Approaches and Practices*, pages 15–37. IGI Global.

Bakar, M. S. and Shaikh Ab Rahman, S. N. (2005). A Kick Start in Implementation of PBL in Computer Programming. In *Regional Conference on Engineering Education RCEE 2005*, Johor, Malaysia.

Baker, A., Navarro, E. O., and van der Hoek, A. (2003). An Experimental Card Game for Teaching Software Engineering Processes. *Journal of Systems and Software*, 75(1–2):3–16.

Barron, B. J. S., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., Bransford, J. D., Cognition, T., and at Vanderbilt, T. G. (1998). Doing With Understanding: Lessons From Research on Problem- and Project-Based Learning. *The Journal of the Learning Sciences*, 7(3&4):271–311.

Beaumont, C. and Fox, C. (2003). Learning Programming: Enhancing Quality Through Problem-Based Learning. In *Proceedings of 4th Annual LTSN-ICS Conference (LTSN-ICS'03).*

Ben-Ari, M., Bednarik, R., Levy, R. B.-B., Ebel, G., Moreno, A., Myller, N., and Sutinen, E. (2011). A Decade of Research and Development on Program Animation: The Jeliot Experience. *Journal of Visual Languages & Computing*, 22(5):375 – 384.

Bergenhenegouwen, G. (1987). Hidden Curriculum in the University. *Higher Education*, 16:535–543.

Biggs, J. (2001). Enhancing learning: A matter of style or approach? In Sternberg, R. and Zhang, L., editors, *Perspectives on Thinking, Learning, and Cognitive Styles*, pages 77–102. Elrbaum.

Biggs, J. and Tang, C. (2011). *Teaching for Quality Learning at University: What the Student Does.* McGraw-Hill Education, 4th edition edition.

Biggs, J. B. (1979). Individual Differences in Study Processes and the Quality of Learning Outcomes. *Higher Education*, 8(4):381–394.

Brodie, L., Zhou, H., and Gibbons, A. (2008). Steps in Developing an Advanced Software Engineering Course Using Problem Based Learning. *Engineering Education*, 3(1):2–12.

Buzan, T. (1991). *Use both sides of your brain: New Mind Mapping Techniques.* Plume Books, New York.

Chimalakonda, S. and Nori, K. V. (2011). Can we Make Software Engineering Education Better by Applying Learning Theories? In *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, page 561.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* Sage Publications, Thousand Oaks, CA, USA, 3rd edition.

Csikszentmihalyi, M. (1996). *Creativity: Flow and the Psychology of Discovery and Invention.* Harper Perennial, New York.

Davis, M. (2009). Understanding the Relationship Between Mood and Creativity: A meta-analysis. *Organizational Behavior and Human Decision Processes*, 108(1):25–38.

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., and Young, P. R. (1989). Computing as a Discipline. *Communications of the ACM*, 32(1):9–23.

dos Santos, S. C., da Conceicao Moraes Batista, M., Cavalcanti, A. P. C., Albuquerque, J. O., and Meira, S. R. (2009). Applying PBL in Software Engineering Education. *Software Engineering Education and Training, Conference on*, 0:182–189.

Dron, J. (2007). *Control and Constraint in E-Learning: Choosing When to Choose.* Idea Group, Hershey, PA, USA.

Dugan, R. F. (2011). A Survey of Computer Science Capstone Course Literature. *Computer Science Education*, 21(3):201–267.

Duke, R., Salzman, E., Burmeister, J., Poon, J., and Murray, L. (2000). Teaching Programming to Beginners – Choosing the Language is just the first step. In *Proceedings of the Australasian conference on Computing education*, ACSE '00, pages 79–86, New York, NY, USA. ACM.

Duveskog, M., Sutinen, E., and Cronje, J. (2013). Design Milieux for Learning Environments in African Contexts. *British Journal of Educational Technology*, page n/a. In press.

Eberle, R. (2008). *Scamper: Creative games and activities for imagination development.* Prufrock Press Inc., Waco, TX.

Entwistle, N. (2007). Research into Student Learning and University Teaching. In Entwistle, N. and Tomlinson, P., editors, *Student Learning and University Teaching*, volume Psychological Aspects of Education–Current Trends of *Monograph Series II*, pages 1–18. The British Psychological Society.

Fransson, A. (1977). On Qualitative Differences in Learning: IV — Effects of Intrinsic Motivation and Extrinsic Test Anxiety on Process and Outcome. *British Journal of Educational Psychology*, 47(3):244–257.

Hakkarainen, K. (2003). Emergence of Progressive-Inquiry Culture in Computer-Supported Collaborative Learning. *Learning Environments Research*, 6(2):199–220.

Hannula, M. (2004). *Affect in Mathematical Thinking and Learning.* PhD thesis, University of Turku.

Herrmann, N. (1996). *The Whole Brain Business Book.* Mc Graw-Hill, New York.

Higgins, J. M. (1994). *101 creative problem solving techniques: The Handbook of New Ideas for Business.* New Management Publishing Co, Winter Park, FL.

Hmelo-Silver, C. E. (2004). Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review*, 16(3):235–266.

Hoskins, S. L. and Newstead, S. E. (2009). Encouraging Student Motivation. In Fry, H., Ketteridge, S., and Marshall, S., editors, *A Handbook for Teaching and Learning in Higher Education*, pages 27–39. Routledge, New York, NY, USA, 3rd edition.

Jackson, N. and Shaw, M. (2006). Developing Subject Perspectives on Creativity in Higher Education. In Jackson, N., Oliver, M., Shaw, M., and Wisdom, J., editors, *Developing Creativity in Higher Education. An imaginative curriculum.* Routledge.

Jonassen, D. H. (2000). Toward a Design Theory of Problem Solving. *Educational Technology Research and Development*, 48(4):63–85.

Joy, M., Sinclair, J., Sun, S., Sitthiworachart, J., and López-González, J. (2009). Categorising Computer Science Education Research. *Education and Information Technologies*, 14:105–126.

Juuti, K. and Lavonen, J. (2006). Design-based research in science education: One step towards methodology. *Nordic Studies in Science Education (NorDiNa)*, 4:54–68.

Kelleher, C. and Pausch, R. (2005). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Comput. Surv.*, 37(2):83–137.

Kirschner, P. A., Sweller, J., and Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, 41(2):75–86.

Kotovsky, K. (2003). Problem Solving—Large/Small, Hard/Easy, Conscious/Nonconscious, Problem-Space/Problem-Solver: The Issue of Dichotomization. In Davidson, J. E. and Sternberg, R. J., editors, *The Psychology of Problem Solving*, pages 373–384. Cambridge University Press, Cambridge, UK.

Kurhila, J. and Vihavainen, A. (2011). Management, Structures and Tools to Scale Up Personal Advising in Large Programming Courses. In *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, pages 3–8, New York, NY, USA. ACM.

Laurillard, D. (2005). Styles and Approaches in Problem-solving. In Marton, F., Hounsell, D., and Entwistle, N., editors, *The Experience of Learning: Implications for teaching and studying in higher education. 3rd ed.*, pages 126–144. University of Edinburgh, Centre for Teaching, Learning and Assessment, Edinburgh, UK.

Lavonen, J. and Meisalo, V. (2009). Creative Problem Solving — University of Helsinki Teacher Training Materials.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. (2004). A Multi-national Study of Reading and Tracing Skills in Novice Programmers. *SIGCSE Bull.*, 36(4):119–150.

Lonka, K. and Ahola, K. (1995). Activating Instruction: How to Foster Study and Thinking Skills in Higher Education. *European Journal of Psychology of Education*, 10:351–368. 10.1007/BF03172926.

Lonka, K., Olkinuora, E., and Mäkinen, J. (2004). Aspects and Prospects of Measuring Studying and Learning in Higher Education. *Educational Psychology Review*, 16:301–323.

Mäkinen, J., Olkinuora, E., and Lonka, K. (2004). Students at Risk: Students' General Study Orientations and Abandoning/prolonging the Course of Studies. *Higher Education*, 48:173–188.

Marton, F. (2005). Approaches to learning. In Marton, F., Hounsell, D., and Entwistle, N., editors, *The Experience of Learning: Implications for teaching and studying in higher education. 3rd ed.*, pages 39–58. University of Edinburgh, Centre for Teaching, Learning and Assessment, Edinburgh, UK.

Marton, F., Beaty, E., and Dall'Alba, G. (1993). Conceptions of Learning. *International Journal of Educational Research*, 19(1):277–300.

Marton, F. and Säljö, R. (1976). On Qualitative Differences in Learning – 2: Outcome as a Function of the Learner's Conception of the Task. *British Journal of Educational Psychology*, 46(2):115–127.

Mazur, E. (1998). *Peer Instruction: A User's Manual.* Prentice Hall, Englewood Cliffs, NJ, USA.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, ITiCSE-WGR '01, pages 125–180, New York, NY, USA. ACM.

Meisalo, V. and Lavonen, J. (2000). Bits and processes on markets and webs: An analysis of virtuality, reality and metaphors in a modern learning environment. *Tietoa ja toimintaa: Journal of Teacher Researcher*, 6(2):10–27.

Mills, C. W. (1956). *The Sociological Imagination.* Oxford University Press, New York, USA.

Moneta, G. B. and Csíkszentmihályi, M. (1999). Models of Concentration in Natural Environments: A Comparative Approach Based on Streams of Experiential Data. *Social Behavior and Personality: An International Journal*, 27(6):603–637.

Mumford, M. D. (2003). Where Have We Been, Where Are We Going? Taking Stock in Creativity Research. *Creativity Research Journal*, 15(2/3):107.

Navarro, E. and van der Hoek, A. (2008). On the Role of Learning Theories in Furthering Software Engineering Education. In Ellis, H., Demurjian, S., and Naveda, J., editors, *Software Engineering: Effective Teaching and Learning Approaches and Practices*, pages 38–59. IGI Global.

Niemiec, C. P. and Ryan, R. M. (2009). Autonomy, Competence, and Relatedness in the Classroom: Applying Self-Determination Theory to Educational Practice. *Theory and Research in Education*, 7(2):133–144.

Noffke, S. E. and Somekh, B. (2009). Introduction. In Noffke, S. E. and Somekh, B., editors, *The SAGE Handbook of Educational Action Research*. SAGE Publications.

Nuutila, E., Törmä, S., and Malmi, L. (2005). PBL and Computer Programming — The Seven Steps Method with Adaptations. *Computer Science Education*, 15(2):123–142.

O'Grady, M. J. (2012). Practical Problem-Based Learning in Computing Education. *ACM Transactions on Computing Education*, 12(3):10:1–10:16.

Osborn, A. (1963). *Applied Imagination; Principles and Procedures of Creative Problem-solving.* Scribner, New York.

Palumbo, D. B. (1990). Programming Language/Problem-Solving Research: A Review of Relevant Issues. *Review of Educational Research*, 60(1):65–89.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007). A Survey of Literature on the Teaching of Introductory Programming. *SIGCSE Bulletin*, 39:204–223.

Pekrun, R. (2006). The Control-Value Theory of Achievement Emotions: Assumptions, Corollaries, and Implications for Educational Research and Practice. *Educational Psychology Review*, 18(4):315–341.

Peng, W. (2010). Practice and Experience in the Application of Problem-based Learning in Computer Programming Course. In *Educational and Information Technology (ICEIT), 2010 International Conference on*, volume 1, pages V1–170 –V1–172.

Perry, Jr., W. G. (1970). *Forms of Intellectual and Ethical Development in the College Years: A Scheme.* Holt, Rinehart and Winston, New York, NY, USA.

Pintrich, P. (2004). A Conceptual Framework for Assessing Motivation and Self-Regulated Learning in College Students. *Educational Psychology Review*, 16:385–407.

Pretz, J. E., Adams, N. J., and Sternberg, R. (2003). Recognizing, Defining, and Representing Problems. In Davidson, J. E. and Sternberg, R. J., editors, *The Psychology of Problem Solving*, pages 3–30. Cambridge University Press, Cambridge, UK.

Qiu, M. and Chen, L. (2010). A Problem-Based Learning Approach to Teaching an Advanced Software Engineering Course. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*.

Randolph, J., Bednarik, R., Silander, P., Gonzalez, J., Myller, N., and Sutinen, E. (2005). A Critical Analysis of the Research Methodologies Reported in the Full Papers of the Proceedings of ICALT 2004. *Advanced Learning Technologies, IEEE International Conference on*, 0:10–14.

Randolph, J. J. (2008). *Multidisciplinary Methods in Educational Technology Research and Development.* HAMK University of Applied Sciences, Hämeenlinna, Finland.

Richardson, J. T. E. (1999). The Concepts and Methods of Phenomenographic Research. *Review of Educational Research*, 69(1):53–82.

Richardson, J. T. E. (2005). Students' Approaches to Learning and Teachers' Approaches to Teaching in Higher Education. *Educational Psychology*, 25(6):673–680.

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2):137–172.

Ryan, R. M. and Deci, E. L. (2000a). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1):54–67.

Ryan, R. M. and Deci, E. L. (2000b). Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist*, 55(1):68–78.

Ryan, R. M. and Deci, E. L. (2001). On Happiness and Human Potentials: A Review of Research on Hedonic and Eudaimonic Well-Being. *Annual Review of Psychology*, 52:141–166.

Schmidt, H. G., Loyens, S. M. M., van Gog, T., and Paas, F. (2007). Problem-Based Learning is Compatible with Human Cognitive Architecture: Commentary on Kirschner, Sweller, and Clark (2006). *Educational Psychologist*, 42(2):91–97.

Schneider, G. M. (2002). A New Model for a Required Senior Research Experience. *SIGCSE Bull.*, 34(4):48–51.

Sio, U. and Ormerod, T. (2009). Does Incubation Enhance Problem Solving? A Meta-analytic Review. *Psychological Bulletin*, 135(1):94–120.

Smith, G. F. (1998). Idea-Generation Techniques: A Formulary of Active Ingredients. *Journal of Creative Behavior*, 32(2):107–133.

Soloway, E. (1986). Learning to Program = Learning to Construct Mechanisms and Explanations. *Communications of the ACM*, 29(9):850–858.

Soloway, E. and Ehrlich, K. (1984). Empirical Studies of Programming Knowledge. *IEEE Transactions on Software Engineering*, SE-10(5):595–609.

Stake, R. E. (1995). *The Art of Case Study Research.* Sage Publications, Thousand Oaks, CA, USA.

Sternberg, R. and Lubart, T. (1999). The Concept of Creativity: Prospects and Paradigms. In Sternberg, R. and Lubart, T., editors, *Handbook of Creativity.* Cambridge University Press.

Sutinen, E. and Tarhio, J. (2001). Teaching to Identify Problems in a Creative Way. In *Proceedings of the FIE'01 Frontiers in Education Conference*, volume T1D, pages 8–13, Reno, NV, USA.

Tedre, M., Apiola, M., and Oroma, J. (2011). Developing IT education in Tanzania: Empowering students. In *Proceedings of the FIE'11 Frontiers in Education Conference*, Rapid City, SD, USA.

Tedre, M. and Kamppuri, M. (2009). Students' Perspectives on Challenges of IT Education in Rural Tanzania. In Cunningham, P. and Cunningham, M., editors, *Proceedings of IST-Africa 2009 Conference*, Kampala, Uganda.

Tedre, M. and Sutinen, E. (2008). Three Traditions of Computing: What Educators Should Know. *Computer Science Education*, 18(3):153–170.

Trigwell, K., Prosser, M., and Waterhouse, F. (1999). Relations Between Teachers' Approaches to Teaching and Students' Approaches to Learning. *Higher Education*, 37(1):57–70.

Van Rossum, E. J. and Schenk, S. M. (1984). The Relationship between Learning Conception, Study Strategy and Learning Outcome. *British Journal of Educational Psychology*, 54(1):73–83.

Vermunt, J. D. and Verloop, N. (1999). Congruence and Friction Between Learning and Teaching. *Learning and Instruction*, 9(3):257–280.

Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, Mass., USA.

Wiliam, D. (2008). Comments on Bulterman-Bos: What Should Education Research Do, and How Should It Do It? *Educational Researcher*, 37(7):432–438.

Winslow, L. E. (1996). Programming Pedagogy–A Psychological Overview. *SIGCSE Bull.*, 28:17–22.

Yin, R. K. (2003). *Case Study Research: Design and Methods*. Sage Publications, Thousand Oaks, CA, USA, 3rd edition.

Ylijoki, O.-H. (2000). Disciplinary Cultures and the Moral Order of Studying - A Case-study of Four Finnish University Departments. *Higher Education*, 39(3):339–362.