ENHANCE STUDENTS' MOTIVATION TO LEARN PROGRAMMING THE DEVELOPING PROCESS OF COURSE DESIGN

Lars Reng

Dept. of Architecture, Design and Media Technology, Aalborg University Copenhagen, lre@create.aau.dk

Lise Busk Kofoed

Dept. of Architecture, Design and Media Technology, Aalborg University Copenhagen, lk@create.aau.dk

ABSTRACT

The paper investigates and poses a number of potent solutions to the problem of teaching programming and other technical topics to students who are neither technically skilled nor aspired to learn these. The research has been done over a period of five years at the bachelor part of the relatively new multidisciplinary engineering education Medialogy at Aalborg University Copenhagen. The education was developed to meet new demands from the interactive media industry and have during the last seven years educated hundreds of bachelors and candidates to fill the void between the many creative fields of media, art, design, and the technical engineering disciplines. Since the dawn of Medialogy it has been the goal to attract young creative artistic students with an interest in technology and media. Through the research presented in this paper it became evident that the education attracted several types of students, including a large group passionate about the artistic/content part of media, art, and design; and with no or little motivation to change their aim and learn technical disciplines to meet the new demands of the industry. The challenge was to find a pedagogical approach for teaching technical topics.
The paper deals with programming, and will explain different steps of a new programming course in detail, and relate students test data to each of the initiatives causing the leaps of improvement. Furthermore the students' technical abilities and enhanced balance between the interdisciplinary disciplines of the study are analysed. The conclusion is that the technical courses have got a higher status for the students. The students now see it as a very important basis for their further study, and their learning results have improved to a more than satisfactory level seen from the study board's point of view.

KEYWORDS

Motivation, learning, programming, image processing, multi-disciplinary, artistic.

INTRODUCTION

During the last decade there has been an amazing development in new media and media-platforms. The industry has changed and so has the demand for the engineers working in the content related fields of the many fast growing areas. As a natural response to the new demands of the industry many universities are trying to establish new educations to close the void, by creating a new type of engineer that can meet the new challenges from industry [1]. The Medialogy master and bachelor programs were established at Aalborg University approximately eight years ago. Medialogy is a multidisciplinary education with a goal to create a new type of engineer with a strong skillset and understanding bridging from the latest media technologies in both software and hardware, to new and old artistic disciplines and to get a deep understanding of the human perceptual system [1]. Traditional technical engineers seem to lack an understanding of both the human cognitive system and the techniques and problems linked to creating high value media content. Equally problematic, is the artists and media directors lack of knowledge of how technically challenging their many ideas are for the engineers to develop. The candidates from Medialogy should be able to close this gap, being a valuable asset for any company as a mediator or part of any of the three groups.

The paper investigates the problems of teaching highly technical topics to students who are neither technically skilled nor keen on improving their skills in these topics. The relatively new interdisciplinary engineering program; Medialogy within the technical faculty at Aalborg University has in the last years been drawing a large amount of students into the void between the many creative fields of media, art, design and the technical engineering disciplines. Whereas the numerous engineering educations have a great appeal on those students who are both inclined and passionate about highly technical topics, Medialogy seem to have a great appeal for those students who are passionate about the creative side of media, art and design. Since the start of the new Medialogy study, it has been evident that the level of the more artistic minded part of student's technical side was far from the desired goal of the program. Studying Medialogy implies a certain level of mathematics and programming which has been a great problem for many students. The more artistic minded students seem not to be interested in those technical subjects and when they start the Medialogy study they cannot see the reason for learning technical subjects with the results that many students have given up and ended their study.

When the authors of this paper started teaching at the Medialogy program numerous attempts with different teachers and programming languages have already been conducted. With somewhat equally disappointing results the heads of study had more or less accepted students lower technical skills as an inevitable weakness. With this problem unsolved the education repeatedly produced strong artistic and humanistic minded students, but with a weak third leg on the technical side.

So the big challenge of finding a pedagogical approach, which could motivate an interest for the technical subjects and avoid students leaving the study were eminent. The authors of this paper took therefore the initiative to request the head of studies the permission to fundamentally change the way technical topics are being taught and this was granted.

Problem Based Learning

All educations under Aalborg University are following the Aalborg University pedagogical model based on Problem Based Learning (PBL) principles [2,3,4]. According to the Aalborg PBL model students would every semester use approximately half of the study time by working in their group with a project where they have to choose and solve a problem within a field selected from the overall theme of their semester. The other half of the time would be used on courses related to the project topic or as a prerequisite for courses for an upcoming semester. So each semester students would have to analyse the semester theme, find a relevant problem they want to solve, use or develop suited theories, find useful methods, design and develop a product, and after thorough testing evaluate if the problem had been solved. The PBL pedagogical approach traditionally is a very motivational pedagogical learning method, but this was not enough for teaching technical subjects at Medialogy. After the first four years it became evident for Medialogy that the type of engineer students that were attracted to Medialogy, were in general less technical inclined and prone to learn hard technical topics. The consequences were that the teachers' expectations of the students' technical level were lowered, and some areas where a technical skillset was the desired goal only a superficial understanding was expected. The students also avoided problem areas that required strong technical skills and thereby also solutions that should have been

achieved by students of Medialogy. It became clear that the education needed to critically evaluate the technical parts of the program, and the result showed an urgent need for improvement of the technical courses, especially programming that is part of the foundation for many of the other courses. It was also clear that a new pedagogical focus on the technical courses was needed.
The PBL approach was still the basic pedagogical approach, but we had to emphasis the experiment and the feed-back especially in connection to reflection[5,6]. Furthermore we had to bring several motivational factors from both extrinsic and intrinsic theory into the pedagogical approach.
Four years ago the first author of this paper was hired to teach programming on the bachelor level of Medialogy. The paper will describe the iterative experimental process to stepwise transform the course in the attempt to raise the students' technical level to the original desired goal.

METHOD

The challenge was to establish the pedagogical approach which could be useful for a new programming course design and which at the same time considered the specific user group – namely students who were not interested in programming.

The method used is based on a case study carried out during five years [7]. The focus each year is to improve the programming course according to the goals in the study regulation. Each year the course is evaluated and the evaluation is the basis for design of an improved new version of the course design. This means that the content and the form of the programming courses have changed, but the overall goal is the same: to learn the students programming according to the level stated by the study regulation. The case study has elements of action research [8] as the teacher is the one who has developed and implemented the different course design, but using reflective step backwards the 'going native' aspect is avoided. The data has been collected during the process, and data are results from observations, course exercises and exam results.

Programming at Medialogy

The bachelor program of Medialogy was originally only a two year education since the students could apply after taking a two year long multimedia college education, which would give them the needed merit to skip the first year of a three year bachelor education. The programming part of the education was therefore split evenly between the third to sixth semester. The third semester would introduce the basic concepts of programming, the fourth would add the object oriented programming (OOP) concepts, the fifth adds 3D graphics programming with OpenGL, and the sixth finally introduces artificial intelligence programming (AIP).

This paper will focus on the changes made to the third semester C/C++ programming course, and the fourth semester Object Oriented Programming (OOP) course. The students are on the third semester introduced to the basic concepts of procedural programming. The curriculum includes elements such as data-types, variables, loops, branching, arrays, structs, functions, pointers, searching & sorting, linked lists, trees, etc. The object oriented programming part of C++ is taught on the fourth semester. The fourth semester OOP course also includes an introduction to the Unified Modelling Language (UML), which is a standardized analysis and design tool that through numerous tables and diagrams can provide an unambiguous description of a software solution.

Detecting the problem (spring 2007)

The first author of this paper had been hired only two weeks before the start of the semester. The task was to take over the teaching of the fourth semester OOP course. The previous lecturer had left suddenly and with no intends to support his successor. The course was therefore run as a standard university course, with two times 45 minutes of lecturing, followed by two times 45 minutes of exercise time in the students' group rooms. In order to ascertain more accurate knowledge of what the course should prepare the students for; a teaching assistant position for the sixth semester artificial intelligence programming course was requested. After only a couple of weeks it became evident that the students' skill level was significantly lower than the requirement for following the courses. Especially on the sixth semester, the level of the students was such that the lecturer, even after lowering the level, had less than a handful of students capable of completing most of the exercises. On the fourth semester OOP course, the students had a very hard time solving even simple problems by use of programming. Many of the students openly admitted that those that had passed the third semester programming course had done so by, to a large degree, memorizing material from the book or slides. The first assumption was therefore that if the students were given more exercises and better opportunities to also use it in other relations, they should be able to catch up.

Apply the knowledge (fall 2007)

Before the start of the new third semester, several meetings were held with the coordinator of studies in order to request approval for changes to the structure and content of not only the programming course, but also the main semester project and minor changes to other courses on the semester. Ever since the beginning of the education the students have been asked to build a product using some kind of image processing on their third semester. This have always been done with the help of software programs like EyesWeb or Jitter, that allows even users with no real programming skills to setup a small program that can do simple image processing tasks, including basic computer vision operations. When the first author of this paper requested that the students in the future should be forced to actually program their whole project application using C++ in order to motivate and improve their skills in programming, most co-lectures and semester supervisors were very sceptical. The students programming skills have previously been too low to attempt this. The lectures were again held in the classical two times 45 minutes of lecture in a large auditorium, and two times 45 minutes of exercise in students' group rooms. Too many students came unprepared and dozed off in the large auditorium, no matter how enthusiastic the lectures were given. So even with 2-3 teaching assistants it was almost impossible to give the amount of help needed. In the project period following the ends of the programming course, a few extra hours were used in order to support the students attempt to implement image processing in their semester projects. This had a remarkable positive effect on the one or two students in each group who ended up doing this. Not only improved their programming skills, but also their detailed understanding of the basic image processing operations. Unfortunately, it was possible for the weakest students to avoid this task, and so they did. The last major change made here was also the one given the greatest concern to the heads of the study. Even if it meant failing more than half, we had to stop sending students on to the next programming course unless they mastered the basics of the first one. The bar was raised and more than half failed. See figure 1.

Leaving the auditorium (spring 2008)

When starting the fourth semester OOP course, more than half the students were still waiting for the re-exam of the third semester programming course. In order to give all students a better chance to complete the fourth semester course it was decided to begin the course with the UML part. All would when have a more even chance of following the material. At the same time a small spare-time after hours lecturing club were established for those that needed help before the re-exam. The students would get free extra teaching by the course teacher, and the authors would get some chance to speak with the students and test their current level as the date approached for the re-exam. Most of this group seemed to have no real interest in learning or having skills in programming. More then half of them did, however, passed the re-exam. The OOP course was moved away from the auditorium and into a large classroom, so it was possible to walk behind the students and help them then they would attempt to implement a program on their laptops. This was done in order to enable a change in the lecture structure. Instead of the old 2+2 structure, lectured material and exercises were now being mixed, so a small exercise of 5-10 minutes were given to conclude each sub topic. The coherent lecturing time was therefore only 15-20 minutes between the exercises, and the students were in the same seminar room during lecturing as well as during exercises. This seemed to have a great positive effect on not only the attention level of the lecturing bursts. Also, the amount of related questions had gone up in the slot between the short lecture and the related exercises. Another huge benefit was that instead of walking around to visit the students in their groups, and repeat the same answer to a question maybe ten times, the blackboard could help answer it once and in more depth, and no teaching assistants were needed. Attempts were made in order to force programming into the semester project as done for the third semester. The coordinator and supervisors did however not want to change their routine here, and the students were therefore left with a two-month break in programming between the last lecture and their exam. The passing requirements were again held high, and many failed. See figure 2.

Teaching by direct visual feedback (fall 2008)

The first author of this paper was, before semester start, offered to also teach the image-processing course. Combined with the role as semester coordinator, and therefore full control of the students' schedule, this gave a much-desired chance to combine the two courses in an attempt to create a synergy effect. With a personal background in image processing, the powerful effect of direct visual feedback on any chance to a program that generates a well-chosen image, is highly valued. The chance to use this on a group of strongly graphical/artistic minded students in daily teaching was therefore a welcomed offer. The image-processing lectures were placed in the auditorium with all exercises done with pen and paper. The following programming lecture, held in a seminar room, would then include implementation and experiments with the filters recently presented. This showed to have a positive effect on both courses, however, surprisingly highest for the image-processing part. Another new element to the lectures was the introduction of a 30-minute test in the beginning of every lecture. The Pop-quiz, as it was called, was introduced as an attempt to get more students to study before the lecture, instead of thinking that showing up was enough to learn it all. In interviews with the students, they responded that they really liked the tests, because it showed them how much they were missing. But still some just decided to show up 30 minutes later if they have not prepared them self for the test. The students were again forced to use the programming in their projects, but the attempts to motivate all to take part in the implementation of the group's project failed, because the project deadline pressure forced the groups to put the strongest programmers on finishing the product on time. Seeing the positive effect on those failing the programming exam the year before, the passing requirements were raised by another approximately ten percent. See figure 3.

Less hours same requirements (spring 2009)

As a result of a change in the study-plan, the hours for the OOP course were cut back by 33%. The curriculum were however unchanged. To resolve this problem, half of the UML were made into a self-study and more time was used on exercises in the remaining lectures. A number of optional hand-ins was scheduled throughout the semester, including an add-on

to the software that produced the product for their project. Unfortunately, this seemed to only make the strong students stronger. The most artistic minded third of the students still seemed to lack the motivation to work through the most challenging exercises, and thereby truly understand the logic behind the language. Different measures had to be taken to reach this group. Through numerous after hours discussions with the students it became evident that the semester did not consists of only one type of student. It seemed that the students could be categorized in three overlapping groups: One type, the 'humanistic-engineers', were very motivated to learn the technical topics of the education, including programming. The second type, the 'medialogiest', seemed to accept that they needed to master both technical, humanistic, and artistic skills to complete the study. The last type, the 'artistic minded', seemed to focus purely on creating contend for media, and therefore completely lacked any intrinsic motivation [9] to learn programming.
See exam results in figure 4.

Sugar and whip (fall 2009)

During the summer month between last semester and the new, we searched for the missing piece to the puzzle. How could the artistic minded third of the students be intrinsic motivated to aspire to learn programming? Sparking up this topic in conversations with leading members of the Danish Film-academy or leading programmers and designers from the local gaming industry all lead to some very interesting statements. So very early in the course, time were used on presenting numerous online job descriptions from gaming and media companies where both designers and artists with some programming knowledge were higher valued than those without. The merging between the programming and image-processing course was increased. Lectures in both courses were now given in a classroom. Some implementations of the image-processing exercises were done in C++ and the students were continuously given image-processing exercises in the programming class wherever it would fit the topic of the lecture. To prevent lazy students to skip the first half hour of the lectures to avoid the Pop-quiz, the questions were spread out over the whole lecture. The lecture and exercise part were switched so no student would attempt to solve the exercises only based on the short presentation given in class. By giving the exercises before the explanations, it was also possible to adjust the focus based on where the students had the most difficulties. The supervisors of the students' projects on the third semester had the feeling that the students level in image-processing had gone up. Equally important were the feedback from the teachers on later semesters. The level of the students programming skills seemed to have gone up. And since failing the lazy students seemed to 'awaken' more than half each year, the passing requirements were raised by another approximately ten percent. See exam results in figure 5.

Earn the right to learn (spring 2010)

Before the beginning of the semester all teacher were informed that starting from 2011, hours for teaching would be cut back. Doing an experiment with the new nonteaching style would be a perfect match with another long desired experiment. Students are pleased to come and go for all lectures, and we cannot demand any hand in of assignments in order to qualify for a lecture. But as the previous semester had shown, the key to lifting the artistic minded students was to find something to motivate them. The regular lectures were cut back by half, as the new 2011 rules would require. Instead the students were given group assignment in the UML curriculum. At the same time a new after hours free study activity was created, inspired by the extra lessons given to failed students the year before. "The Dedicated Programmers Society", also known as DPS were made for those that felt the need for extra help. Since this 'club' was a free study activity we could require each student to hand in an assignment before each 'meeting' in order to qualify for the free help. Getting the 'weaker' students in a smaller forum where all were there because they needed help to understand the curriculum, opened up for questions from students who would normally be to afraid to ask a ‘stupid’ question. But it also gave us unique one-on-one interaction with exactly the students we needed to motivate the most. The DPS was not run perfectly on its first year, but the potential was obvious. See figure 6.

Passed and future students (fall 2010)

The year before we had used small parts of the first couple of lectures to present statements and online job descriptions from some of the possible future employers of the artistic focused students. All with the same message: “programming skills is an asset”. A few older students had even been invited to show what they could do with the programming skills they had learned here. Having strong programmers form the master program present their work had been a great experience. It seemed, however, to have a greater impact on the technically inclined students than the artistic minded. The solution was obvious. This year a few very gifted artistic focused master students were asked to present some of their impressive artistic creations. But then turning the focus on how they used the skills they had learned in the programming course to improve their efficiency when modelling their 3D creations. This finally seem to be exactly the kind of motivational inspiration the artistic minded students could use to get a little intrinsic drive. The course was run with more focus on the exercises and less on the lecturing. This clearly divided the students in those that study, and those that did not. The passing requirements were again raised the ten percent. The goal is now to increase the number of students passing the course in the first attempt. See the grades in figure 7.

Reading for the goal (spring 2011)

This course was run almost as the one the year before. The students were asked to do a few extra assignments and hand them in for approval. Two lectures were converted into a large group assignment on UML. The exam results seem to confirm that the high level on the third semester course do in fact have a positive impact on programming courses on later semesters. See figure 8.

Sharpen the saw (fall 2011)

Having used almost five years improving the courses, the semester composition, and also the study plan; it was great to see that the average skills of the students had reached the desired level. Following and observing some the students all the way to their master, to see them invent and implement the hybrid artistic, creative, and technical products they were meant to do. It was therefore saddening that the university had decided to cut back the amount of teaching hours for both the programming and image-processing course, and also the hours students had to implement their semester project. Knowing that many of the students would have a hard time learning the curriculum with less help and more time alone, it was crucial to get them more focus and well prepared to each lecture. Having use almost five years fine tuning the courses, it was decided to attempt to improve the discipline of the students. Inspired by Stephen R. Covey’s famous phrase, we would try to “sharpen the saw”[9]. By taking elements from neuro-linguguistic programming and self motivation experts such as Anthoney Robbins[10], half the time of the first two lectures where used on discussing how the students could improve their motivation and self discipline. A few exercises, including visualizing of personal written goals were attempted. Resent research by Dan Pink[11] and others have challenged the way intelligent creative people are motivated. A large potion of the exercises was therefore replaced with a number of small personal projects designed to also motivate the artistic minded students. The students were autonomous developing small games or image processing related applications. Even though the students were given less teaching the exam were again made approximately ten percent harder, and reaching the desired level set five years earlier. The results were surprisingly positive. See figure 9.

RESULTS

The follow eight graphs picture the exam results of the third semester programming course (figure 1-4), and the fourth semester OOP course (figure 5-8). The grades -3 and 00 are failing grades, whereas the grades 02, 4, 7, 10, and 12 are passing grades.

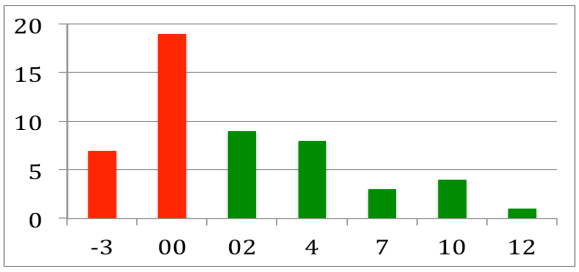
Figure 1. Exam results fall 2007

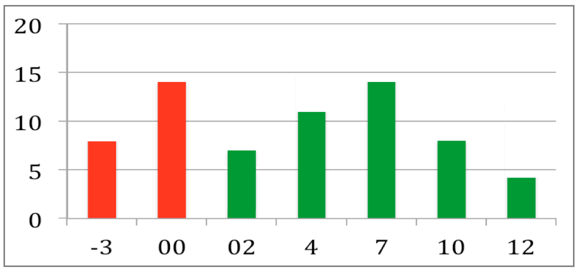
Figure 2. Exam results spring 2008

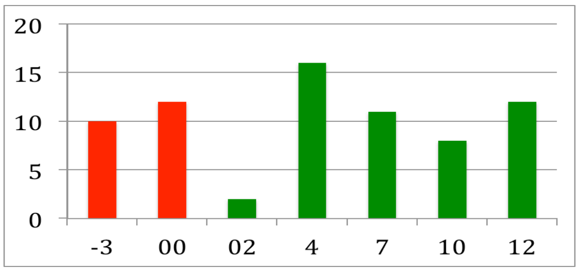
Figure 3. Exam results fall 2008

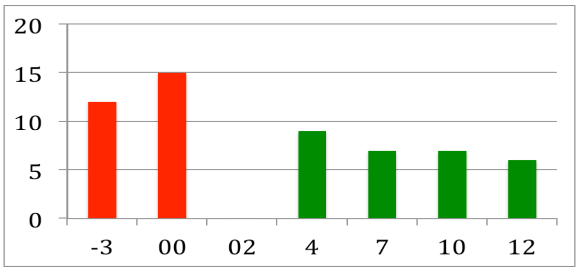
Figure 4. Exam results spring 2009

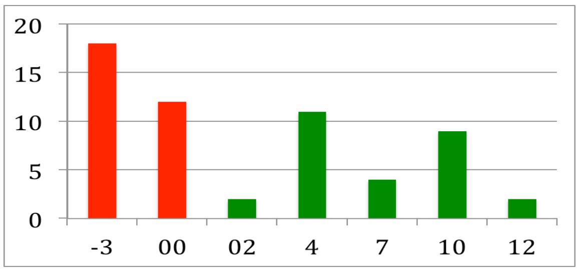
Figure 5. Exam results fall 2009

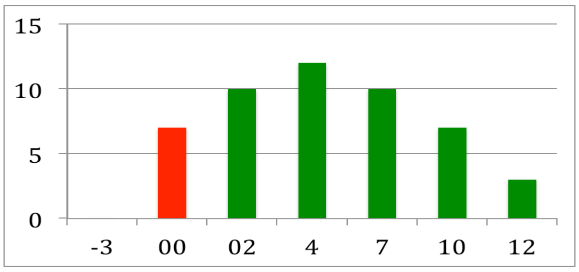
Figure 6. Exam results spring 2010

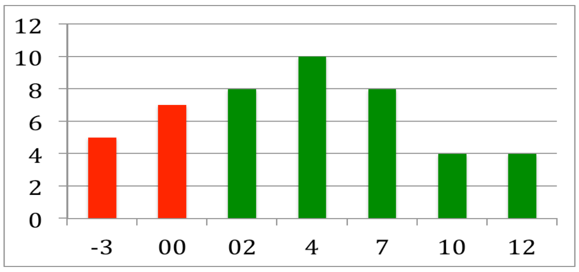
Figure 7. Exam results fall 2010

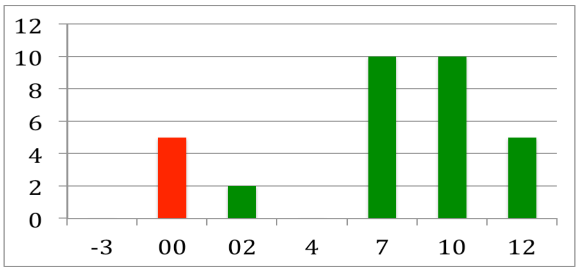
Figure 8. Exam results spring 2011

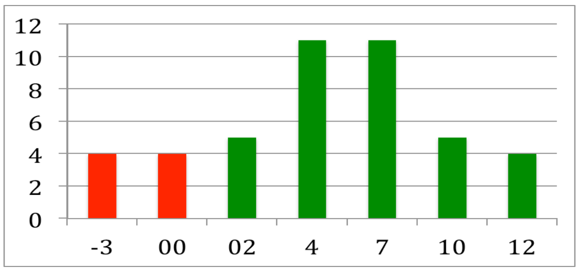
Figure 9. Exam results fall 2011

CONCLUSION

A promise made by the first author of the paper four years ago: “Give me the freedom to change the way they learn and use programming, and I will find a way to teach all of them”, has unfortunately still not been meet. The last five years of constant recursive learning for the students in the lectures and dialogue with the teacher between them, have however fundamentally changed the way programming is taught at Medialogy. The merging of courses and strengthened link to the problem based project work has worked very well. Most of the changes and new ideas have proven them selves useful, and will continuously be refined in hope to find the way to motivate artistic minded students to learn programming.

Each year have added several changes to the way we teach the students today. No technical teacher at Medialogy is today using standard auditorium lectures. The complete merge of lecture and exercise is adopted by most other courses. A few other teachers have also removed lectures with course material in favour of quest lectures, field trips, etc. to enhance the intrinsic motivation. The strong blend between programming, image processing, and the PBL project of the third semester seems today so natural its hard to imagine the old ways. The two new initiatives of lectures to sharpen the students, and increasing creativity and intrinsic motivation by more longer autonomous mini projects will without doubt be attempted again, and also on other courses.

Most important, it is great to see that the results so clearly show that the students have improved. Not only have a large number of the last group of students passed an exam that would have failed almost every student five years ago. The products the students create on later semesters demonstrate that Medialogy students are getting a strong third technical leg.

REFERENCES

[1] Cowan, John (2006). *On becoming an Innovative University Teacher – reflection in Action.* Open University Press, McGraw-Hill Education.

[2] Busk Kofoed, L. Nordahl, R. (2007) *Learning Lab – teaching experienced students PBL.* In proceedings of the 18th Conference of the Australasian Association for Engineering Education, Melbourne; Department of Computer Science and Software Engineering. University of Melbourne.

[3] Kofoed, Lise; Hansen, Søren ; Kolmos, Anette.(2004) Teaching Prosess competencies in a PBL curriculum. In:*The Aalborg model : progress, diversity and challenges*. (Eds. Kolmos, Anette. Fink, Flemming K., Krogh, Lone. Aalborg: Aalborg University Press.

[4] Nordahl, Rolf and Kofoed, Lise B. (2008). *Medialogy – design of a transdisciplinary education using problem based learning.* In Proceedings form SEFI 36th Annual Conference., Aalborg.

[5] Schön, Donald, D. (2009*) The Reflective Practitioner. How Professionals Think In Action.* Ashgate Publishing Limited, England.

[6] Schön, Donald, D. (1990) *Educating the Reflective Practitioner – Toward a New Desogn for Teaching and Learning in the Professions*.Jossey – Bass Publishers, San Francisco.

[7] Yin, R. (1994). *Case study research: Design and methods* (2nd ed.). Beverly Hills, CA: Sage Publishing.

[8] Argyris, Putnam and Smith (1985), *Action Science*. USA. Jossey-Bass Inc. Publishers.

[9] Covey, Stephen R. (1990) The 7 habits of Highly Effective People (1st ed.). Free Press.

[10] Robbins, Anthoney. (1997) Unlimited power: The New Science of Personal Achievement. Free Press.

[11] Pink, Daniel. (2011) Drive, Riverhead Trade.

Biographical Information

Lars Reng is a Teaching Assistant Professor at the Department of Architecture, Design & Media Technology, Aalborg University, Copenhagen. He is teaching programming, image processing, artificial intelligence, computer graphics, and PBL project work. He has a background in gesture recognition, and is currently doing research in teaching of technical topics to multidisciplinary engineering students.

Lise Busk Kofoed is Professor and Head of Section in the Department of Architecture, Design & Media Technology, Aalborg University, Copenhagen. She earned her PhD in Organizational Development, Working Environment and Learning. She teaches undergraduate and graduate subjects within organization and management development in the engineering curriculum as well as pedagogical issues for the university teaching staff. A part of the pedagogical work is courses for PhD students and supervisors. Her research areas are development of the Problem Based Learning concept within engineering educations with special focus on cross disciplinary programs.