Enhancing Students' Motivation to Learn Programming by Using Direct Visual Feedback

Reng, Lars

Published in:
Innovations 2012

Publication date:
2012

Document Version
Early version, also known as pre-print

Chapter XX

Enhancing Students' Motivation to Learn Programming by Using Direct Visual Feedback

LARS RENG[1]
[1]*Department of Architecture, Design & Media Technology, Aalborg University, Denmark. E-mail: lre@create.aau.dk*

In several new approaches to interdisciplinary engineering education, students often see the technical subjects such as mathematics and programming as problematic to learn. These technical subjects of concern here are within programming, and image-processing algorithms are used as a means to provide direct visual feedback for learning basic C/C++. The pedagogical approach is within a Problem Based Learning (PBL) framework and is based on dialogue and collaborative learning. At the same time, the intention was to establish a community of practice among the students and the teachers. A direct visual feedback and a higher level of merging between the artistic, creative and technical lectures have been the focus of motivation, as well as a complete restructuring of elements of the technical lectures. The paper will present test results based on over 400 students, gathered over a period of five years. The paper will explain different steps of the new programming courses in detail, and relate students' test data to each of the initiatives causing the leaps of improvement. Furthermore, the students' technical abilities and the enhanced balance between the interdisciplinary disciplines of the study are analysed. The conclusion is that the technical courses have attained a higher status for the students. Students now see them as a very important basis for their further study, and their learning results have improved to a satisfactory level from the study board's point of view.

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 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 programmes 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 programmes were established at Aalborg University approximately nine years ago. Medialogy is a multidisciplinary programme aimed first at creating a new type of engineer with a strong skill set and understanding, ranging from the latest media technologies in both software and hardware to new and old artistic disciplines, and also at gaining a deep understanding of the human perceptual system [1, 2]. 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 realization of how technically challenging their many ideas are for the engineers to develop. The candidates from Medialogy should be able to close this gap and be a valuable asset for any company, as a mediator or as part of any of the three groups mentioned above.

In this article, the author 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 programme Medialogy, within the engineering and science faculty at Aalborg University, has, in resent years, been drawing a large number of students into the void between the many creative fields of media, art, design and the technical engineering disciplines. While the numerous engineering programmes hold a great appeal for those students who are both inclined towards and passionate about highly technical topics, Medialogy seems to appeal to those students who are passionate about the creative side of media, art and design. Since the start of the new Medialogy study programme eight to nine years ago, it has been evident that the technical level of the more artistic minded part of students was below that required by the programme.. 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 programme they cannot see the reason for learning them, with the result that many students have withdrawn from the course. So a big challenge for the programme has been to find a pedagogical approach which could stir an interest in the technical subjects and avoid students pulling out.. The author of this article has, with full support from the head of studies, taken the initiative to try to fundamentally change the way technical topics are being taught.

All programmes at Aalborg University follow the Aalborg University pedagogical model based on PBL principles [2, 3]. According to the Aalborg PBL model, students use approximately half of the study time every semester to work in their group on 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 is used on courses related to the project topic or as a prerequisite for courses for an upcoming semester. So each semester students have to analyse the semester theme, find a relevant problem they want to solve, use or develop suitable theories, find useful methods, design and develop a product, and after thorough testing evaluate whether the problem has been solved.

After the first three years it became evident to the Medialogy teachers that the type of engineer students that were attracted to Medialogy were less technically inclined and prone to learning hard technical topics. So the consequences were that the teachers' expectations of the students' technical level were lowered, and in some areas, where a technical skill set was the desired goal, only a superficial understanding was expected. It became clear that the technical parts of the programme needed to be critically evaluated.,

and the results showed an urgent need for improvement of the technical courses, especially programming.

Four years ago the author of this article was hired to teach programming at the bachelor level of Medialogy. The paper will describe the stepwise transformation of the course in the attempt to raise the students' technical level to the original desired goal. The paper will focus in particular on the merger between programming, image processing and the third-semester PBL project, in an attempt to provide direct visual feedback for the programming course.

METHODS

The author has used the case method combined with action research [4, 5]. All four programming courses have been followed and analysed over the last five years by the author. The courses have been developed during the process according to previous experiences and results. The prosess has included characteristics from continuous improvement (CI) processes, and the evaluation processes have been inspired by CI evaluation methods [6]. The data are course assignments, exams, and interviews with students and project supervisors.

The pedagogical approach was based on the PBL principles [7] and one of the main focuses of the courses was to relate the course content to understandable problems for the students so they could see the purpose of the learning [8]. During the process it became evident that an important part of the pedagogical strategy should be based on motivation. Students had lots of assignments as well as exams during the semester, so the extrinsic motivation was established. Test results and interviews showed that one group of students was very motivated and was almost enjoying programming. The challenge was to find a pedagogical method so all students could explore and see the benefit of learning programming. The goal was to find "the code" so this group of students could find their intrinsic motivation [9, 10]. Another teaching strategy was to emphasize and support reflection as a means for understanding [11, 12] and to use students' previous experience either from programming courses or from related areas to support their learning process and to maintain motivation. Experiments were used with inspiration from Donald Schôn [12] when developing different aspects of the courses. But one of the most important factors was to develop course material and course content "just in time".

PROGRAMMING IN MEDIALOGY

The bachelor Medialogy programme was originally only a two-year education since the students could apply after taking a two-year multimedia college education, which would give them the needed accreditation to skip the first year of the three-year bachelor education. The programming part of the education was therefore split evenly from the third to the sixth semester. The third semester introduced the basic concepts of programming, the fourth added the object-oriented programming (OOP) concepts, the fifth added 3D graphics programming with OpenGL (GFX), and the sixth finally introduced artificial intelligence programming (AIP). All four programming courses are graded using the seven-point scale system. The grades -3 and 00 are failing grades. The grades 02, 4, 7, 10 and 12 are passing grades (12 = A).

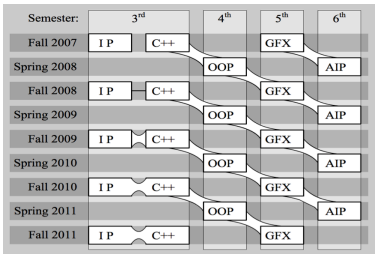

FIGURE 1

COURSE OVERVIEW. THE FOUR COLOURED ["SHADED" AFTER PRINTING?] COLUMNS DEPICT THE FOUR SEMESTERS. THE NINE ROWS DEPICT THE SEMESTERS OVER THE LAST FOUR AND A HALF YEARS.

This paper will focus on the changes made to the third-semester C/C++ programming course. The students are introduced to the basic concepts of procedural programming in the third semester. The curriculum includes elements such as data types, variables, loops, branching, arrays, structs, functions, pointers, searching and sorting, linked lists, trees, etc. Object-oriented programming (OPP) is not taught until the fourth semester.

FIGURE 2

SIX LOW-RESOLUTION VERSIONS OF THE LENNA IMAGE. (1) ORIGINAL, (2) INVERTED, (3) WITH 'SALT'& 'PEPPER' NOISE, (4) BLURRED, (5) EDGE DETECTION, (6) MEDIAN-FILTERED RESULT OF IMAGE (3)[3?]

Unknown

Formatted: Font:(Default) Arial, 10 pt

Image Processing

The third-semester image-processing (IP) course is a general introduction course to basic image processing. The course does, however, have a strong focus on basic computer vision since this is needed for most third-semester project artefacts. This short section will describe some of the image-processing filter algorithms that enable the students to get direct visual feedback when programming. The author of this article has a master's degree in computer vision and graphics, and has often used images to depict debugging information and a range of algorithms from physics to artificial intelligence. It was therefore a most welcome opportunity to have the programming course running in parallel with a course on image processing. The basic algorithms used for noise reduction, colour and marker detection and 2D visualization can cover up to 80% of the material in the programming course if used right. In the figure below, five of the simplest most used operations are depicted on the world-famous Lenna image.

Inverting (Figure 2 (1)) an image or changing its brightness and contrast requires very little programming. This makes it a very good exercise when introducing arrays, since most students have performed these operations in Adobe Photoshop or a similar graphic program. Operations such as blurring (Figure 2 (4)) or edge detection (Figure 2 (5)) require a little more, and are therefore perfect continuations. The median filter, on the other hand, requires a sorting method and can therefore be a challenge to most new programmers.

Detecting the Problem (Spring 2007)

The first course to teach was the Object-Oriented Programming (OOP) course. The previous teacher had left rather suddenly a month before with no desire to support the successor; therefore, no exchange of knowledge and experience from previous years was given. As a result, the course was designed and run in a very traditional engineering education style. The course, which was the students' second programming course, was split into 15 lectures of 2x45 minutes of auditorium lecturing and 2x45 minutes of assisted exercises, which took place in the students' own group rooms (each group has a room for project work). It became evident rather quickly that only a fraction of the students had mastered the basics of the first programming course which had ended only a month before. As a result, it was very hard to focus the teaching on the desired curriculum. To investigate whether this problem had occurred before, and also to see how well the students performed at the end of their bachelor programme, the entire sixth-semester Artificial Intelligence Programming (AIP) course was monitored with a focus on the students' abilities to apply basic programming to solve the exercises. The results were shocking. From a whole semester of sixth-semester students, only one student dared to attempt to solve the final free competition exercise. So even though his simple artificial intelligence did not work perfectly, he still won. Numerous meetings with the coordinator of studies concluded that the last four years of changing teachers and choice of programming language had proven that more extensive changes would be required in order to break the reoccurring problem of the weak technical level of the majority of the students. High levels of freedom to initiate new ideas were therefore granted. Also, it was agreed that the requirements to pass the exam should be raised significantly in the first year, and then by another 10% the following three to four years, as the new improvements to the course would hopefully gain full effect.

Raising the Bar (Fall 2007)

Interviews with the students in their fourth and sixth semesters had revealed that many had been able to pass the third-semester programming exam using a high level of memorization of blackboard examples. In order to prepare the students for the challenging programming courses of later semesters, the severity of the third-semester course had to be raised. This would clearly result in a much higher number of failed exams and students dropping out, unless a better way of teaching the curriculum was found. The students had, for the last four years, since the beginning of the programme, undertaken a course in image processing in the third semester, and they had built an interactive installation using this as part of their semester project. This had, however, been done with tools such as EyesWeb, where the details of the different operations were hidden and handled by the tool.

In order to bring more focus to the programming part of the semester it was decided to remove the tools, and instead the students were required to program the entire project artefact in C/C++. The open computer vision library (OpenCV) was all the external software they were allowed to use. The programming course was run in parallel with the early stages of their semester project and the image-processing course. The programming course was run in a classical style, using two hours of lecturing and another two for exercises. The students came poorly prepared to the programming lectures and almost half of them seemed to lack any motivation to learn programming, or do the exercises required. More than half of the students were unable to apply what had just been taught in the lecture only minutes before the exercises. There was almost no flow in the exercises, and most students seemed to be stuck as soon as the teaching assistants had left them. The semester project artefacts were programmed by the project groups in C/C++ (each project group had four to six students). Project exams unfortunately revealed that only the strongest programmers in the group were able to explain the source code, indicating that not all students had benefitted from the extended programming practice possibility which their work on the semester projects should have offered. On the other hand, those that had programmed major parts of the projects could explain in a very high level of detail how the different image-processing algorithms affect an image. The level of difficulty of the programming exam questions was greatly increased, as previously agreed. This unfortunately resulted in more than 50% failing the course (see Figure 3).

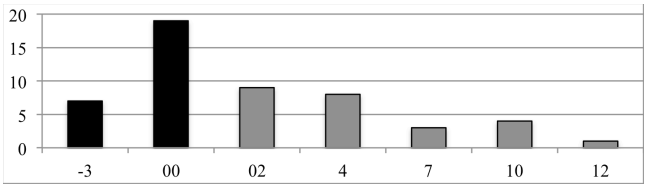

FIGURE 3
RAISING THE BAR (FALL 2007). THE GRADES -3 AND 00 ARE FAILING GRADES. THE GRADES 02, 4, 7, 10 AND 12 ARE PASSING GRADES. (12 = A). THE X AND Y AXES REPRESENT THE GRADE AND FREQUENCY FOR THE FALL COURSE 2007

Merging the Courses (Fall 2008)

Interviews with supervisors and skilled students the previous year had indicated that programming all the algorithms from the image-processing course would greatly improve the students' understanding of how and why they worked the way they do. Another interesting observation made during the breaks of the programming courses was that many of the students who lacked the motivation to learn programming were using their breaks to continue to work on the more artistic courses. It was therefore decided to attempt to merge the image-processing and programming courses, and to use all the filters used in the more artistic classes as programming exercises. If the more artistic-minded students were asked to recreate some of the effects they liked from software such as Adobe Photoshop, it might increase their motivation to understand how these filters were programmed, and thereby learn more programming. Another important benefit was that changes in images often seemed to make more sense for the students than just numbers from a program.

It was therefore decided to completely merge the two courses: programming and image processing. Instead of starting with a two-hour lecture and then two hours of exercises, the students were taken out of the auditorium and into a large seminar room. This allowed the lecture and exercise time to be interleaved in 15–20 minute intervals, thereby allowing students to implement each new method directly after it had been presented and discussed. The teacher was close to the students during lecture time and could move around between them. This made it easier for students to ask questions, but also for the teacher to askwhether the students had understood what was going on. According to interviews with students and supervisors, the effect of this merger not only had a positive effect on the students' knowledge on image processing but it also resulted in better semester project artefacts. The difficulty level of the programming exam was raised by approximately 10%. Even under these conditions the results of the exam indicated that the changes from the previous year had had a positive effect on the level of the students' programming skills (see Figure 4).

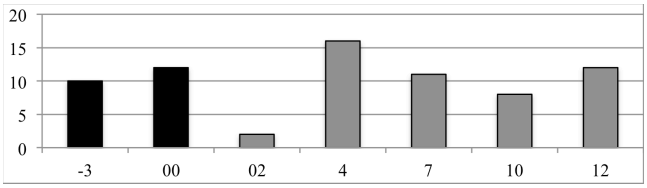

FIGURE 4
MERGING THE COURSES (FALL 2008). THE GRADES -3 AND 00 ARE FAILING GRADES. THE GRADES 02, 4, 7, 10 AND 12 ARE PASSING GRADES. (12 = A). THE X AND Y AXES REPRESENT THE GRADE AND FREQUENCY FOR THE FALL COURSE 2008

The Semester Spirit (Fall 2009)

A phenomenon that is discussed daily among teachers but is still a bit of a mystery is the semester spirit. Even though we know there is a great difference between the best and

weakest students and a great difference between the most and least dedicated students, everybody knows that semesters can be very different. Something can make students at one semester accept lazy behaviour, or they can work harder than those the year before them. Even though only minor changes were added to the image-processing and programming course, the students in the fall semester seemed to be less motivated and more reluctant to do all the exercises. This had an overall effect on both semester projects and the programming exam. Again the difficulty level of the programming exam had again been raised by approximately 10% (see Figure 5).

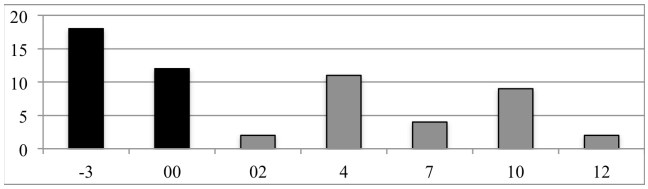

FIGURE 5
THE SEMESTER SPIRIT (FALL 2009). THE GRADES -3 AND 00 ARE FAILING GRADES. THE GRADES 02, 4, 7, 10 AND 12 ARE PASSING GRADES. (12 = A). THE X AND Y AXES REPRESENT THE GRADE AND FREQUENCY FOR THE FALL COURSE 2009

Focus on the Artists (Fall 2010)

In order to do everything possible to avoid another year with a collective lack of motivation, several of the master's students that had passed the exam two years earlier were invited to talk to the new third-semester students at the very beginning of the course. It was not only the strong programmers that were invited, as a few very artistic-minded master's students explained how they used their programming skills to improve their work daily.

The aim of this special effort was to try and motivate "the hard to reach" and more artistic-minded new students. Another initiative added this year was that the last part of the image-processing course was delayed and delivered at the time when the students

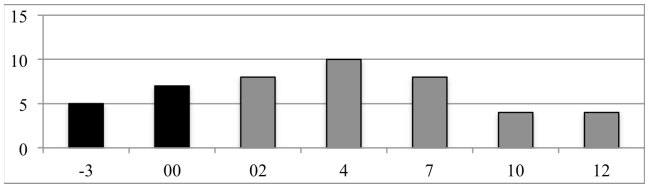

FIGURE 6
FOCUS ON THE ARTISTS (FALL 2010). THE GRADES -3 AND 00 ARE FAILING GRADES. THE GRADES 02, 4, 7, 10 AND 12 ARE PASSING GRADES. (12 = A). THE X AND Y AXES REPRESENT THE GRADE AND FREQUENCY FOR THE FALL COURSE 2010

were implementing their semester project artefact. Instead of traditional course teaching, these hours were used as an open support for any image-processing or programming problem related to the semester project. The programming exam had the desired level of difficulty at approximately another 10% harder than the year before. The results were better than expected (see Figure 6).

The Will to Succeed (Fall 2011)

Observing the students in the seminar room as well as in their group work, it was evident that while some students could not stop programming, others could not wait for programming to stop. Informal interviews with this group revealed that they did not see programming as an important part of their future education, nor their future job. The author therefore decided to use a large part of the first couple of lectures to help the students find an intrinsic motivation to follow the courses. Guest lecturers from both the master programme and media industry were invited to explain the importance of programming for programmers, designers, managers and also artists. After trying to establish the need for programming skills or a knowledge of programming principles, a new experimental addition was added to the course. Principles from motivational coaches were tried out in the form of neuro-linguistic programming (NLP)-style goal setting [13]. Finally, a small Facebook group was created for the students so they could cry for help when most fellow students would notice it, after school hours. To further explore the possibilities of direct visual feedback, a few examples and exercises were given in the 3D graphics environments that most students hope to work with in their future jobs. The feedback from these were very positive, and the motivation among the more artistic-minded students seemed to be increasing. The programming exam difficulty was raised to the final level of approximately 10% harder than the year before. The level was now fully in line with the study plan and equal to purely technical engineering programmes. The results were better than expected (see Figure 7).

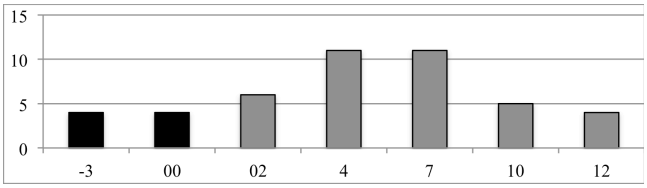

FIGURE 7
THE WILL TO SUCCEED (FALL 2011). THE GRADES -3 AND 00 ARE FAILING GRADES. THE GRADES 02, 4, 7, 10 AND 12 ARE PASSING GRADES. (12 = A). THE X AND Y AXES REPRESENT THE GRADE AND FREQUENCY FOR THE FALL COURSE 2011

RESULTS

Students are graded using the seven-point scale system. The grades -3 and 00 are failing grades. The grades 02, 4, 7, 10 and 12 are passing grades (12 = A). The figure below (Figure 8) depicts the total sum of grades given according to the seven-point scale. The x-

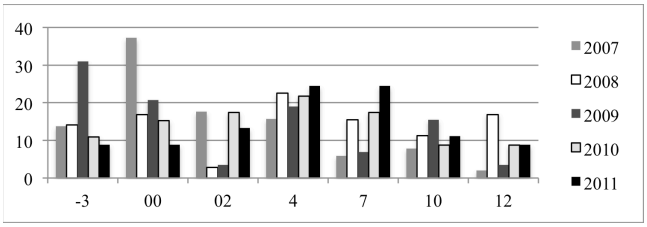

FIGURE 8
GRADE DISTRIBUTION IN PERCENTAGE BETWEEN 2007 AND 2011. THE X-AXIS REPRESENTS THE GRADE AND THE Y-AXIS THE FREQUENCY IN PERCENTAGE FOR THE FALL COURSES 2007–2011

axis represents the grade and the y-axis the frequency in percentage for the fall courses 2007–2011. As mentioned earlier, the 2009 results bring doubt to the results and conclusion. It is therefore important to keep in mind that the exam tests have been gradually made more difficult, so the students would have to score approximately 40% better in 2011 to earn the same grade as another student from 2007. The final year shows that the pedagogical approach was good for the students' learning. The combination of motivation and the untraditional teaching methods, especially the opportunity students had to work with programming problems in their projects, were proven to enhance the general level of programming.

CONCLUSION AND PERSPECTIVES

The merging of the programming and image-processing courses was carried out as a natural support to the students' new requirement of having to implement an image-processing-based interactive artefact through the use of C/C++ in their semester projects. When this initiative was approved five years ago there was no or little actual programming in the projects in the bachelor semester (sixth). The artificial intelligence programming course was rarely used in any semester projects. Today this has changed dramatically. Several bachelor groups are today implementing advanced algorithms for artificial intelligence in their semester projects. Numerous projects are being programmed in industry-standard game engines, and students are starting to program full-scale commercial productions in their early master's semesters. The ideas for improving the course were good, and the use of programming in the students' semester project supported motivation as well as learning.

The direct visual feedback achieved by using images as an output for most of the exercises in the programming course has been observed as a strong motivator and potent

debugger for the very graphically minded students of Medialogy. The direct visual feedback is not the only initiative applied to the third-semester programming course. For the moment it is therefore not possible to conclude the effect based on the exam results depicted in the results section.

The course will soon be running for its sixth year, and once again the positive trends from the previous years are being used to full effect. The specially targeted group of artistic-minded students are still not being fully motivated. To achieve this goal, several new initiatives have been initiated and will be explored in more depth. The students will be presented with members from the media industry to inspire them and inform them of the demand for programming-educated employees. The students have been asked to write down their goals for the course, as well as how they intend to reach these. Finally, a few small assignments will try to expand the use of direct visual feedback by moving the students into 3D programming using OpenGL or a 3D game engine such as Unity, Ogre or Unreal. There is no doubt that the journey started five years ago will continue to improve the technical teaching and success in general for all students, and especially for the students not motivated by technical subjects. It is my hope and belief that this method can be used in numerous fields of teaching. The concept of bringing intrinsically motivating elements from other fields into a course, and trying to explore what can trigger a drive and will to learn, should be applicable to almost any field.

ACKNOWLEDGEMENTS

The development of this article would not have been what it is today without the support and counselling of supervisor and mentor Lise Busk Kofoed. A great deal of respect and gratitude is therefore given as well as deserved.

REFERENCES

1. R. Nordahl and L. B. Kofoed, "Medialogy – Design of a Transdisciplinary Education using Problem-based Learning," *Proceedings from SEFI 36th Annual Conference*, 2008, Aalborg.
2. L. B. Kofoed and R. Nordahl, "Learning Lab – Teaching Experienced Students PBL," *Proceedings of the 18th Conference of the Australasian Association for Engineering Education,* Melbourne, Department of Computer Science and Software Engineering, University of Melbourne, 2007.
3. L. B. Kofoed, S. Hansen, and A. Kolmos, "Teaching Process Competencies in a PBL Curriculum," A. Kolmos, Flemming K. Fink, and L. Krogh (eds.), *The Aalborg Model: Progress, Diversity and Challenges*, 2004, Aalborg University Press, Aalborg.
4. K. E. Weick, K. M. Sutcheliffe, and D. Obstfeld, "Organizing and the Process of Sensemaking," *Organization Science,* 16(4), 409-421, 2005.
5. D. Schön, *The Reflective Practitioner. How Professionals Think in Action,* Ashgate Publishing Limited, 2009, England.
6. F. Jørgensen and L. B. Kofoed, "Integrating the Development of Continuous Improvement and Innovation Capabilities into Engineering Education," *European Journal of Engineering Education,* 32(2), 181-191, 2007.

7. E. de. Graaff, and A. Kolmos, "Characteristics of Problem-based Learning," *International Journal of Engineering Education,* 5(19), 657-662, 2003.

8. D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development,* Prentice Hall, 1984, Englewood Cliffs.

9. M. Csikszentmihalyi and K. Rathunde, "The Measurement of Flow in Everyday Life: Towards a Theory of Emergent Motivation," in J. E. Jacobs, *Developmental Perspectives on Motivation, 60. Nebraska Symposium on Motivation,* University of Nebraska Press, 1993, Lincoln.

10. M. Csikszentmihalyi, S. Abuhamdeh, and J. Nakamura, "Flow," in A. Elliot, *Handbook of Competence and Motivation,* 598-698. The Guilford Press, 2005, New York.

11. J. Cowan, *On Becoming an Innovative University Teacher: Reflection in Action,* Open University Press, McGraw-Hill Education, 2006.

12. D. Schön, *Educating the Reflective Practitioner – Toward a New Design for Teaching and Learning in the Professions,* Jossey-Bass Publishers, 1990, San Francisco.

13. A. Robbins, *Unlimited Power: The New Science of Personal Achievement,* Free Press, 1997.

Lars Reng received an MSE in Computer Vision and Graphics form Aalborg University in 2003. Since 2007, he has been a Teaching Assistant Professor of Medialogy at the Department of Architecture, Design & Media Technology, Aalborg University, Copenhagen.