The Design of Sweden's First 5-year Computer Science and Software Engineering Program

Fredrik Heintz
Dept. of Computer and Information Science
Linköping University
fredrik.heintz@liu.se

Inger Erlander Klein
Dept. of Electrical Engineering
Linköping University
inger.klein@liu.se

ABSTRACT

In 2013 Linköping University started the first 5-year engineering program in Computer Science and Software Engineering in Sweden. The goals of the program are to provide a holistic perspective on modern large scale software development, to provide a deep and broad understanding of computer science and computational thinking, and encourage innovation and entrepreneurship. The student response has been very good with more than 600 applicants to the 30 slots, of which more than 130 had this program as their first choice among all programs in Sweden. In this paper we present the goals, the design principles, and the resulting program. The ACM/IEEE CS Curricula has been used to make sure that the program provides a solid foundation in Computer Science. Three pedagogical ideas that we have used are (1) project courses to integrate theory and practice as well as provide experience with the most common form of working in industry; (2) courses that cover multiple programming paradigms and languages as well as multiple software development methodologies so that the students are prepared to take on the continual changes we know will come; and (3) a special course in engineering professionalism with groups of students from the first three years together reflecting on topics related to being a professional engineer. The paper concludes with a discussion about some important aspects such as computational thinking and the relation to the ACM/IEEE CS Curricula.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education

1. INTRODUCTION

We live in a changing world. Our society becomes more and more dependent on computers and software. The pace of change is steadily increasing, with technology as one important driving factor. To manage the pace and the complexity when developing sophisticated large-scale software intensive systems we need software engineers with the appropriate knowledge, skills and attitudes. To meet this challenge Linköping University has developed Sweden's first 5-year Master of Science in Engineering program in Computer Science and Software Engineering (in Swedish Civilingenjör Mjukvaruteknik).

In this paper we describe the vision, goals, and design of the program. Effort has been put in to formulate goals and principles that can be used by other similar programs as well, even though the circumstances and realization may significantly differ. This should also make the presentation more interesting to a larger audience. We then discuss some important aspects such as computational thinking and the relation to the ACM/IEEE CS Curricula, which is also used to compare related programs in Sweden.

One reason for writing this paper is to invite discussion and constructive criticism around the question on how to design a modern computer science and software engineering program for the changing world we live in. We therefore encourage you to contact us if you have questions or comments.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCSE'14, March 3–8, 2014, Atlanta, GA, USA.
Copyright 2014 ACM 978-1-4503-2605-6/14/03 ...$15.00.
http://dx.doi.org/10.1145/2538862.2538925.

2. ENGINEERING EDUCATION IN SWEDEN

2.1 5-year Master of Science in Engineering programs

The Swedish Master of Science in Engineering (Civilingenjör) programs are professional degrees consisting of 5 years of studies. Normally the students are awarded both a Civilingenjör degree, a Bachelor of Science in Engineering degree, and a Master of Science in Engineering degree. There are a number of national requirements these programs must fulfil including engineering skills such as the ability to conceive, design, implement and operate systems, but also other aspects for example communication skills, team work, and aspects regarding social sciences such as the role technology has in society considering both ethical and cultural aspects. The entrance requirement is a high school degree with math level E, physics level B, and chemistry level A. In Austria, Finland, and Greece, as well as previously in Germany, the corresponding degree is called Diploma Engineer. All engineering programs share a common core of problem solving, mathematics and natural sciences.

2.2 Computer Related Programs at Linköping University

Linköping Institute of Technology, which is part of Linköping University, currently has five programs related to computing besides the new program. There are two 5-year Master of Science in Engineering programs, Computer Science and Engineering (Civilingenjör Datateknik) and Information Technology (Civilingenjör Informationsteknologi), one 3-year Bachelor of Science in Engineering program in Computer Engineering (Högskoleingenjör Datateknik), one 3-year Bachelor of Science program called Innovative Programming (Innovativ programmering), and one 2-year Master of

Science program in Computer Science. Here we focus on the 5-year engineering programs. The 5-year engineering program in Computer Science and Engineering covers hardware, software and the interplay, while Information Technology focuses on infrastructure needed to transmitt, structure and present information.

2.3 Related Programs in Sweden

The other institutes of technology in Sweden such as The Royal Institute of Technology (KTH) and Chalmers also have related 5-year Master of Science in Engineering programs. Most of them are closer to computer engineering than to computer science. The programs that are the closest are the 5-year Master of Science in Engineering programs in Computer Science and Engineering at KTH and in Information Technology at Chalmers. Even though the programs have similarities there are important differences. For example, the IT program at Chalmers focuses more on Software Engineering than on Computer Science and mainly learn a single programming language (Java) rather than give a broad foundation of many different languages.

In 2011 a study was conducted that compared 10 of the Swedish 5-year Master of Science in Engineering programs in Computer Science relative to the ACM CS Curricula 2008 [4]. One conclusion from the study is that there is a very wide variate of programs, ranging from hardware oriented to software oriented, from theoretical to more applied, and from general engineering programs to more focused computer science programs. The new Computer Science and Software Engineering program is software oriented, both theoretical and applied, and more focused on computer science than on general engineering even though it has significant computer and electrical engineering content. In Section 5 the new program is compared to these 10 programs.

3. VISION AND GOALS

The vision for the program is to educate the best software engineers in the world. The program should provide the best possible qualification for an exciting, successful and meaningful career in the very broad field of computer science and software engineering. Graduates from the program will take on leading roles in the software industry, be driving forces in the further development of computer science and software engineering, and start new companies based on innovative ideas and an entrepreneurial spirit.

The program should provide the necessary knowledge, skills and attitudes to be able to develop every type of software intensive system, be able to program in every programming language, be able to use every programming methodology, and more importantly be able to choose the appropriate language and methodology for a particular situation. The program should prepare the students for a professional career as software engineers where soft skills such as communication, personal leadership and working together with others in cross-functional teams are essential.

The concrete goals for the program are to:

1. provide a solid foundation in computer science and mathematics;
2. provide an understanding of and experience with computational thinking;
3. provide a holistic understanding of modern software development;
4. provide an understanding of and experience with several different programming languages and paradigms as well as industrial software development methodologies;
5. provide an understanding of and experience with important application areas such as modern large scale distributed and embedded systems, mobile and social applications, data driven decision-making, and AI and robotics;
6. leverage project courses with relevant subject content that integrates and applies theory;
7. encourage entrepreneurship and innovation; and
8. emphasize global challenges, sustainable development and a global world.

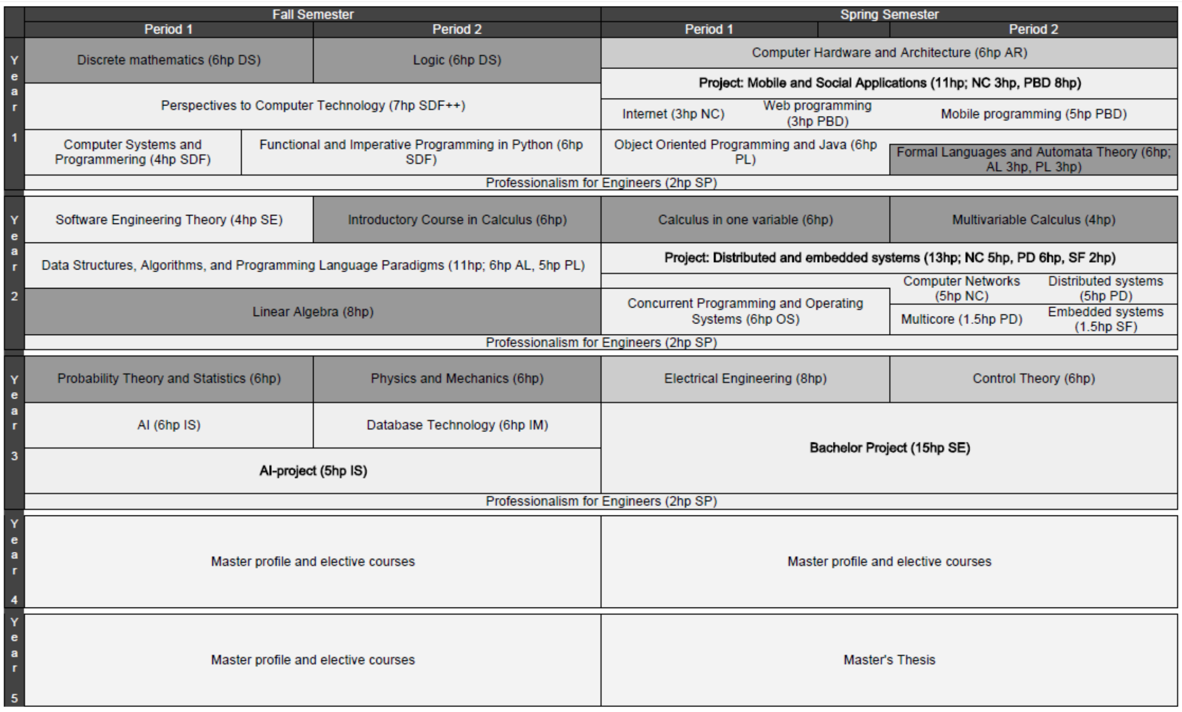
By achieving these goals the program will live up to its vision.

4. DESIGN

From the first idea of starting a new program to the first students arrived it has only been 2 years. The real design process started in January 2011 and it has taken approximately 18 months to reach a stable version of the first three years of the program. It has involved a group of teachers from Computer Science, Electrical Engineering and Mathematics as well as student representatives. There were some requirements on the process such as minimizing the number of completely new courses. The main challenges were to select what should be included in the mandatory first three years while making room for project courses; to find a good balance between computer science, computer engineering, electrical engineering, software engineering and mathematics especially considering that this is Master of Science in Engineering program; and to find the right order of the courses so that the prerequisites are satisfied and the courses result in a clear progression through the program. We are very happy with the end result, even though we had to make some compromises.

Many engineering programs take a bottom-up approach to their subject. They start with mathematics and foundational courses the first three years and then study applications and specializations the last two years. A major problem with this approach is that students do not see the full picture until at the very end of their studies. There is clear indications that a significant share of the students that drop out do it because they lack this overview. Another fact is that students develop and mature a lot during their studies. This means that most of them are not able to fully understand and grasp the material the first time. The new program should therefore provide three iterations through the broad and deep areas of Computer Science and Software Engineering. By having three iterations, each basically covering the whole discipline, at different levels of detail, we believe that the students understanding will greatly increase while we also should reduce the number of students dropping out due to a lack of overview.

The first semester is the first iteration which gives a broad overview of the whole field and the whole education. The second iteration is the following 5 semesters which gives a solid foundation in computer science, software engineering and the relevant parts of electrical engineering. It is concluded with a bachelor project where the students clearly demonstrate their knowledge, skills and attitudes developed during the first two iterations. The third iteration is the Master profile the last two years where the students deepen their knowledge in important areas of computer science and software engineering based on interest and aptitude. Linköping University is a large university which means that we can provide a very large selection of advanced courses taught by active researchers based on current research. The third iteration and the whole education program is concluded with a full semester Master's Thesis where the student is required to demonstrate that he or she satisfies all the requirements for a Master of Science in Computer Science and Software Engineering degree.



	Fall Semester		Spring Semester	
	Period 1	Period 2	Period 1	Period 2
Year 1	Discrete mathematics (6hp DS)	Logic (6hp DS)	Computer Hardware and Architecture (6hp AR)	
	Perspectives to Computer Technology (7hp SDF++)		**Project: Mobile and Social Applications (11hp; NC 3hp, PBD 8hp)**	
			Internet (3hp NC); Web programming (3hp PBD)	Mobile programming (5hp PBD)
	Computer Systems and Programmering (4hp SDF)	Functional and Imperative Programming in Python (6hp SDF)	Object Oriented Programming and Java (6hp PL)	Formal Languages and Automata Theory (6hp; AL 3hp, PL 3hp)
	Professionalism for Engineers (2hp SP)			
Year 2	Software Engineering Theory (4hp SE)	Introductory Course in Calculus (6hp)	Calculus in one variable (6hp)	Multivariable Calculus (4hp)
	Data Structures, Algorithms, and Programming Language Paradigms (11hp; 6hp AL, 5hp PL)		**Project: Distributed and embedded systems (13hp; NC 5hp, PD 6hp, SF 2hp)**	
	Linear Algebra (8hp)		Concurrent Programming and Operating Systems (6hp OS)	Computer Networks (5hp NC); Distributed systems (5hp PD); Multicore (1.5hp PD); Embedded systems (1.5hp SF)
	Professionalism for Engineers (2hp SP)			
Year 3	Probability Theory and Statistics (6hp)	Physics and Mechanics (6hp)	Electrical Engineering (8hp)	Control Theory (6hp)
	AI (6hp IS)	Database Technology (6hp IM)	**Bachelor Project (15hp SE)**	
	AI-project (5hp IS)			
	Professionalism for Engineers (2hp SP)			
Year 4	Master profile and elective courses		Master profile and elective courses	
Year 5	Master profile and elective courses		Master's Thesis	

Figure 1: An overview of the program. Light gray boxes are computer science courses and software engineering courses, medium gray boxes are engineering courses, and dark gray boxes are math courses.

Figure 1 gives an overview of the program. Light gray boxes are computer science courses and software engineering courses, medium gray boxes are engineering courses, and dark gray boxes are math courses. Each year is divided into two semesters, one fall semester and one spring semester. Each semester is divided into two periods. Each semester consists of 30 ECTS credits called "hp". Since each semester is 20 weeks, 1.5hp roughly corresponds to one week of work. Courses are normally 6hp and given within a single period. The acronyms in parentheses refer to knowledge areas in the ACM/IEEE Computer Science Curricula 2013 [5]. The subjects mentioned for the two spring project courses describe their content in some more detail. The sum of the credits for the parts is equal to the total amount of credits for the course.

4.1 Design Principles

The design is based on the following general principles:

1. There should be a clear progression through the program, both with regard to the main subjects Computer Science and Software Engineering and with regard to how the work is being done and what is required from the students. The students should be continuously challenged to improve and develop.
2. The first three years should consist of mandatory courses to provide a common foundation while the last two years should provide the students with the largest possible freedom to develop their own interests and specializations.
3. There should normally be three courses in parallel, one of these should be a math or theory course and another should be a programming course. This should stimulate both those who are mainly programming oriented and those that are more math oriented. It also improves the work load balance.
4. Each semester the first two years should focus on a single programming language to provide focus and avoid confusion. The students should learn multiple different programming languages and paradigms with a clear emphasis on the concepts and principles which will allow them to quickly understand and learn any programming language. The set of languages should include at least Python, C/C++ and Java. The set of paradigms should include at least functional, imperative and object oriented programming.
5. Each semester should have a project course characterized by a larger and relatively open-ended assignment that the students should do, either individually or in groups. Projects are very important to integrate and apply theoretical concepts and techniques while developing software systems. A project course may also cover new theory.
6. The project courses should use different software development methodologies to provide experience and understanding of a wide variety of industrial methodologies. At the same time it is very important that the choice of methodology is relevant for the project, otherwise the students will not realize the benefits but rather feel that it is something artificially added. As with programming languages, it is essential to teach the concepts and principles behind the methodologies and to make the students reflect over how the choice of methodology affects the outcome of a project.
7. Mathematics should be an essential and integrated part of the program. It is important to emphasize the strong connections between math and computer science since this provides a foundation for computational thinking. The first year should focus on discrete math and logic which are directly relevant for computer science. The second year should include the continuous math needed mainly for the electrical engineering courses. The third year should include applied math courses, which directly uses the continuous math, where probability

theory and statistics are essential for both computer science and electrical engineering.

8. During the first three years the students should have projects related to the application areas stated in the goals, i.e. mobile and social applications, modern large scale distributed and embedded systems, and AI and data driven decision-making to provide a broad understanding of these important domains.

9. Topics such as testing, usability and security are essential to modern software development and should be a natural part of most software courses, especially the project courses.

10. The program should give a basic understanding of computer architecture, electrical engineering and control theory, which provides the basic understanding of the very interesting interplay between the outside world and computers through sensors and actuators. This is especially important since this is a Master of Science in Engineering program.

4.2 Year 1

The first semester consists of two math courses, discrete math and logic, three computer science courses, perspectives on computer technology, computer systems and programming, and functional and imperative programming in Python, and the engineering professionalism course. The perspectives course provides a broad overview of computer science and related areas. The two programming courses introduces the Unix computer systems used and programming in Python. The functional and imperative programming paradigms are both covered with a special emphasis on solving problems both iteratively and recursively. Recursion, proof by induction and their relations are central concepts. The first programming course also requires the students to try out programming in shell script, Prolog, Haskell and SQL. The purpose is to give a feeling for different languages and different programming styles early on. In the second period, the students do their first project in the perspectives course. The purpose of the project is to do something exciting and to have something to show at the end of the first semester. The projects are quite large, groups of three students each supposed to spend 130 hours on the project, and open-ended. There are projects on programming Lego Mindstorm robots, Nao humanoid robots, web-based map applications using the Google Maps API, programming automated XPilot players, and writing programs to play capture the flag strategy game. The main goal of this project is to make the students realize that they are capable of developing a significant piece of software while at the same time realizing that they have much more to learn. An important side benefit is that the students have something concrete to show at the end of the first semester. Being able to describe to friends and family what they are doing is important both for the students and for recruiting new students to the program.

Last, but not least, the course on Professionalism for Engineers which covers all the first three years. This is an "integrating" course with groups of students from all the first three years. The inspiration to the course comes from a similar course at KTH and is a course with many functions [3]. There are two main purposes of the course. First, to integrate both students from the first three years and the program as a whole. Second, to teach the students about engineering professionalism which is a very important subject that includes soft skills, ethics, personal leadership, and team work. A novel and very exciting improvement to the course, compared to the courses at KTH, is that it uses the dialog seminar method for reflexion and learning from experience [2]. The method is developed by researchers at KTH and Combitech AB, a consultant company with connections to SAAB. Combitech is also involved in developing the course and training the teachers in the methodology. We believe that the reception by the students of the course will be greatly improved by developing the course together with industry that really knows what is essential for being a professional engineer.

After the first semester the students should have a good general understanding of computer science, programming and the formal foundations of discrete math and logic.

The second semester consists of a theory course, a computer engineering course, a programming course and the first large project course. The course on formal languages and automata theory gives an introduction to theoretical computer science and continues the math track. The course on computer hardware and architecture gives an introduction to computer engineering including the basics of digital circuits, the main components of a computer and some assembler programming. The course on Object Oriented Programming and Java adds a new programming paradigm to the imperative and functional paradigms introduced in the first semester. Finally, and maybe most importantly, the project course on Mobile and Social Applications introduces an important application area and more systematic ways of developing software. The focus in this project course is the individual developer designing usable mobile applications with a social component. Special emphasis is placed on considering privacy related issues. The software engineering principles introduced includes design prototyping, unit testing, and code reviews.

4.3 Year 2

The third semester consists of a course on theory of software engineering, a large course on data structures, algorithms and programming paradigms using C++, a course in linear algebra and an introductory course in calculus. This semester concludes the foundational programming courses. The programming paradigm part is especially important for this, as it puts all the programming concepts and languages learned into perspective. It should also show the students that the same concept or principle can be realized in many different ways and that designing a programming language basically means making a set of deliberate choices. After this the students should be competent programmers.

The choice of having the software engineering theory course after the first project course is a deliberate decision. The reason is that we want the students to have experience with software engineering before learning the theory. This way, the students are mentally prepared for the content and have maybe already encountered several of the issues that software engineering handles.

The fourth semester covers single and multi-variable calculus and computer systems. So far, the focus has been on individual computers or devices. This semester the concept of a computer system is studied in great detail and from many different perspectives. The foundation is a computer which can be connected in a network of computers. The concept of an operating system controlling the basic functions of a computer and the concept of a computer network are essential. Starting from this core, extensions to multi-core computers, parallel computers and embedded computers are naturally made. Another important step is to view a set of networked computers as a single distributed system and design and implement programs for these. From a software engineering perspective concepts such as fault tolerance, scalability, security and testing of distributed systems are central.

4.4 Year 3

The third year covers AI and databases which are two very important areas of computer science, probability theory and statistics which are essential for empirical aspects of engineering, and physics and mechanics which provide a basic understanding of the physical world. This knowledge is extended and put to very good use in the electrical engineering and control theory courses where the students learn the fundamentals of electric circuits, signals and systems including measuring, modeling and controlling them. The fifth semester project course is focused on AI and data-driven decision making. The goal is to develop a system with an AI component. The project applies many of the math courses since machine learning and data mining is based on linear algebra and statistics.

The Bachelor Project corresponds to half the work done in the sixth semester. This is a large programming project where students work in teams of 6–8 students for an external customer. The goal of the project is to develop a program or a system for the external customer in the most appropriate way that satisfies the customer's requirements and expectations. This means that the students have to decide which software development methodology is appropriate and then follow it. They also have to deal with issues such as customers changing their minds and coworkers leaving (which is simulated by swapping people between groups). Each student should write a Bachelor thesis based on his or her work.

When all the courses the first three years, including the Bachelor project, are finished the student is awarded a Bachelor of Science in Computer Science and Software Engineering degree. This opens up possibilities to change universities and study somewhere else for the Master's degree. From our experience very few students choose this option, most complete the full five years at the same university.

4.5 Year 4-5

The last two years of the program corresponds to a Master's Program and consists almost exclusively of elective courses. To get a Master's degree a student needs to take 90hp (three semesters worth of) courses on the advanced level including the 30hp Master's Thesis. To guide students there are a number of master profiles consisting of selected courses from which a student has to take at least 36hp of which at least 30hp on the advanced level.

Two new master profiles will be developed for this program: Software Engineering and AI and Data-Driven Decision-Making. There are also existing master profiles from the Computer Science and Computer Engineering program such as Programming and Algorithms (the main computer science profile), Game Programming, Safe and Secure Systems and International Software Engineering (which gives a double degree with Harbin Institute of Technology).

The master profile in Software Engineering deepens and broadens the knowledge and skills of the students in this area. At least two important new courses will be developed. The first, a large scale software engineering course based on an open-source project is essential to give students experience of working on really large software projects together with many other people. Important skills practiced are to take an existing software system and learn to understand it. To be able to take existing legacy code and make solid extensions and improvements is a skill in great demand. Our industry representatives have been very clear that this is an important skill and something that many software engineers are not very good at, since they have mostly developed small programs from scratch. The second course is on software entrepreneurship. The software industry is very exciting and interesting from a commercial perspective. This is probably the industry where the step from an idea to a commercial product is the smallest. A brand new company can easily get global distribution of their products either through the Internet or an app store. The software industry is an extremely expansive and innovative industry where new inventions are made daily and where the right idea at the right time can result in a billion dollar company within a year (such as Instagram).

The master profile in AI and Data-Driven Decision-Making focuses on the very exciting area of artificial intelligence, machine learning, and data-driven decision making which is becoming more important for all types of software. This profile is based on the very strong research in the related areas at Linköping University.

5. DISCUSSION

5.1 Computational Thinking

The general skill of computational thinking is central to the program. This includes recursive decomposition of problems into smaller and more manageable parts; finding and generalizing patterns by for example systematically solving small cases; and using simulation to either approximate solutions or generate and test hypotheses. These problem solving strategies use concepts from both computer science and mathematics and often rely on programming.

An important challenge is to tie the programming and the mathematics courses together. In the programming courses the teachers try to show the connections explicitly, for example by pointing out the relations between inductive proofs and recursive computations and to use functions studied in the math courses as programming examples (the most classical example being the factorial). This shows the benefits of mathematics for programming. Our current challenge is to show the benefits of programming in mathematics. To this end, we are currently exploring how computational thinking can be used in the discrete math course, which is the first math course for these students. As part of this we have conducted a survey to get data about the attitude towards mathematics and programming and the view on the interplay between mathematics and programming. Preliminary data shows that the students have a very positive attitude towards mathematics and an extremely positive attitude towards programming. It also shows that the students realize that there is an important connection between the subjects, but that they do not have the skills to actually combine mathematics and programming yet. To show the students the importance and usefulness of the interplay as well as teach them some basic skills we have extended the normal homework assignments with advice on how to use programming as a tool to explore the mathematical problems as well as to find candidate solutions, which they then need to prove are correct. If our activities are successful, the students should be much better at using mathematics as a tool when programming and programming as a tool when solving mathematical problems.

5.2 The ACM/IEEE CS Curricula

The ACM/IEEE CS Curricula [5] is a very important and impressive document clearly describing the knowledge that every computer scientist should have. Our view is that they have done a very good job in capturing the required content. We have therefore used the curricula as a guide and as a measuring stick for making sure that we cover all the relevant aspects of computer science. In the overview of the program, Figure 1, we have classified the content of each course relative to the ACM CS Curricula 2013. When we look into the details of the knowledge areas we clearly see that we cover most of the content of all the knowledge areas.

As stated earlier, Karlfeldt compares 10 Swedish computer science related Master of Science in Engineering programs relative to the ACM CS Curricula 2008 [4]. Figure 2 shows the computer science content in the new program (blue) for each knowledge area in the curricula compared to the average content in the 10 programs studied by Karlfeldt. It is very important to notice that the data in the figure only shows the obligatory courses during the first three years. This means that the courses studied during the last two years

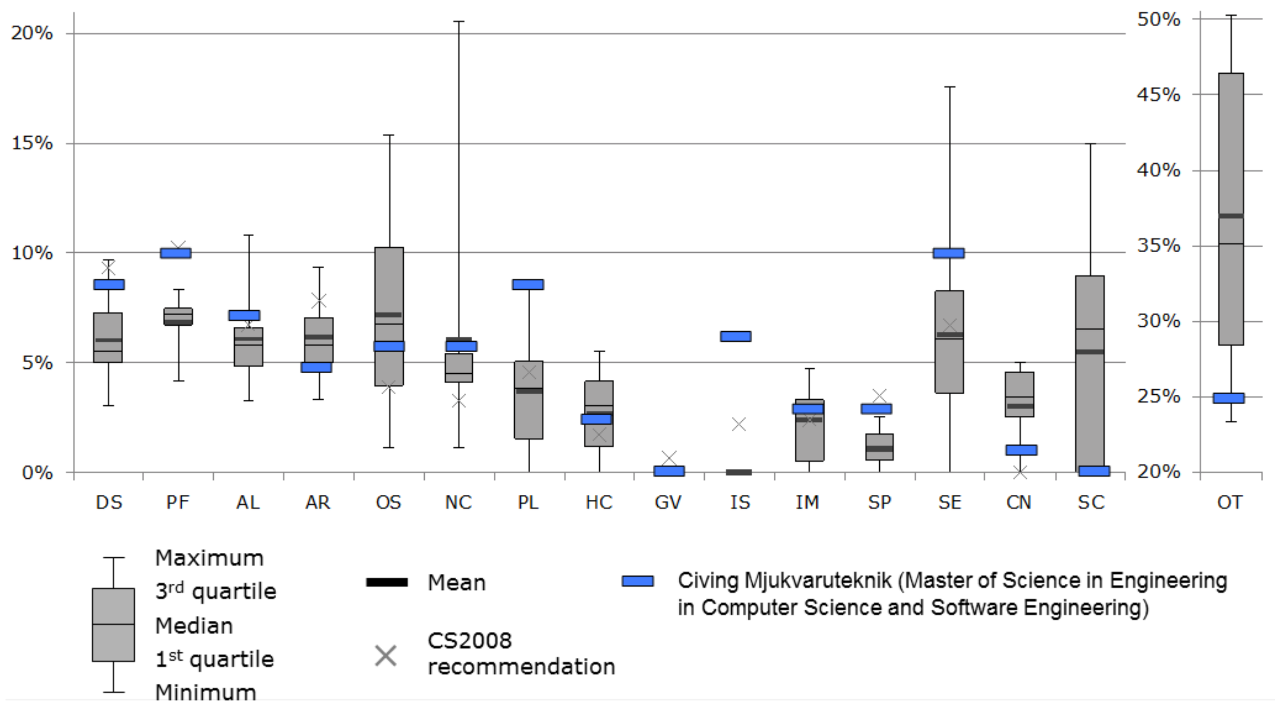


Figure 2: A comparison among Swedish CS related programs based on ACM CS Curricula 2008 based on Figure 1 in [4].

are not shown. Percentages is used instead of actual number of credits due to the different number of obligatory credits.

Two important conclusions can be drawn from Figure 2, first that the new program satisfies the ACM CS Curricula 2008 during the first three years and second that the new program has significantly more computer science content compared to the other programs as it has more than the average share of each knowledge area. There are programs with more content in specific areas, but not the same broad coverage of the whole computer science discipline.

5.3 Engineering Aspects

To be able to specify and verify the goals in a structured and systematic way, all engineering education at Linköping university is based on the CDIO concept (Conceive, Design, Implement, Operate) [1]. This is a concept for developing, executing, evaluating and engineering education including among other things a basic document – the CDIO syllabus – specifying expected knowledge and skills that a student is expected to have. All engineering educations at Linköping university uses a variety of the CDIO syllabus called the LiTH syllabus. The document has four main headlines:

1. Knowledge and reasoning in mathematics, natural sciences and engineering;
2. Personal and professional skills and attributes;
3. Interpersonal skills: teamwork and communication;
4. Conceiving, designing, implementing, and operating systems in the enterprise and social context.

There is a mapping between the LiTH syllabus and the national requirement for a 5-year engineering degree, which means that the national requirements are satisfied if the LiTH syllabus is satisfied.

The syllabus covers technical subjects but also personal, professional and interpersonal skills. These skills are part of the project courses, and is the main focus in the new course Professionalism for Engineers described in section 4.2.

6. CONCLUSIONS

In this paper we have described the vision, goals and design of the first 5-year engineering program in Computer Science and Software Engineering in Sweden. The goals are providing a holistic perspective on modern large scale software development, providing a deep and broad understanding of computer science and computational thinking, and encouraging innovation and entrepreneurship.

Designing a new 5-year program in Computer Science and Software Engineering is an exciting, challenging and very rewarding endeavor. We believe that we have managed to make the appropriate trade-offs when selecting and ordering the content of the program to achieve its goals. The student response has also been very good with more than 600 applicants to the 30 slots, of which more than 130 had this program as their first choice among all programs in Sweden. We are now working hard at making sure that the program really lives up to its vision and provides the best possible software engineering education in the world.

7. REFERENCES

[1] The CDIO initiative, http://www.cdio.org.

[2] B. Goranzon, R. Ennals, and M. Hammeron. *Dialogue, skill and tacit knowledge*. Wiley. com, 2006.

[3] V. Kann. En programsammanhållande kurs med många funktioner (in swedish). In *Proc. 3:e utvecklingskonferensen*, pages 153–156, 2011.

[4] J. Karlfeldt. Den svenska dataingenjören – en jämförande studie av 10 svenska civilingenjörsutbildningar inom det datavetenskapliga området. Master's thesis, KTH, 2012.

[5] M. Sahami, S. Roach, et al. Computer Science Curricula 2013 – Final Report Pre-release Version 0.9, Oct. 2013.