Program evaluation of the Bachelor Programs in Computer Science and Computer Technology

Department of Informatics,

University of Bergen

February 2015

Table of contents:

1. Introduction

1.1 Background

This program evaluation is a result of a work that was conducted during spring 2014. This work resulted in changes in the department's bachelor portfolio.

The committee consisted of:

Petter Bjørstad	Head of department
Dag Langmyhr	Program sensor, external member from IFI, University of Oslo
Johan Rusvik	Master student and head of student committee
Anya Bagge	Post doctor, program development
Øyvind Ytrehus	Professor, Selmer centre
Dag Haugland	Professor in optimization and head of program board
Marc Rehmsmeier	Professor, bioinformatics
Ida Rosenlund	Study advisor and secretary

The appointed committee has had six meetings between January and April 2014. Between the meetings the members got homework. The committee's task has to been to suggest how the department of Informatics, with its existing resources, can offer the best possible teaching within ICT, teaching that is both internationally up-to-date and future-oriented. The teaching should establish a basis of knowledge that qualifies for work both in public and private sector, as well as open up for further studies (Master and PhD) within the field of informatics.

The work of the committee has resulted in this report where two new bachelor programs are described and where alterations are suggested to two of the three existing ones. After the report was discussed at the department's annual seminar, the suggestions were further developed until the Department Council decided to make changes in two of the existing bachelor programs, and apply for starting two new ones. The application was approved, and these changes will take effect starting autumn 2015.

A revision of the bachelor programs should increase the number of students completing their bachelor degree, since the department today has a high drop-out rate. Additionally, one should see an increased number of applicants. The last couple of years the bachelor programs have had an increasing trend, a revision should further strengthen this.

Today, the bachelor program in Computer Technology attracts a lot more students than the one in Computer Science. Spring 2014 Computer Technology had 115 students while Computer Science had 49. Ideally, we would like both programs to have a solid number of students. Especially more students studying Computer Science is desirable as this program is more related to further studies and research. In addition, our department is a theoretically anchored one, and an excellent environment for academic studies in informatics.

The programs we have today replaced a single BSc in informatics and were introduced in 2008. "Confining the BSc program to students with pronounced theoretical interest could lead to a significant reduction in the number of students recruited. Consequently, a program with a more practical/popular profile was needed, in addition to a theoretical profiled program" (Quote from Dag

Haugland's homework due 27.01.14). Feedback tells us that students have some difficulties seeing the difference between the two programs. As a response to this the committee suggests to make the distinction between the two programs clearer.

A very valuable and important reference for the work of the committee is the "Computer Science Curricula 2013", a report from ACM and IEEE Computer Society with international curricula guidelines. Over the last 40 years ACM and IEEE-Computer Society have put out several such curricula guidelines (typically every 5 years) for undergraduate programs in Computer Science.

The department aims to offer the students a qualitatively strong curriculum. The CSC-report describes an international standard and its recommendations are reliable, updated, soundly based, and entrenched in an empirical study of the two previous reports. By basing our bachelor program on the Body of Knowledge, recommended knowledge areas that should be covered in a BSc., we demonstrate that we are internationally competitive and that our students hold international standard when graduating from UiB.

1.2. Students perspective on today's situation

By student representative in the committee, Johan Rusvik.

To get an impression of the students choices regarding the bachelor programs Computer Technology and Computer Science we did an informal anonymous survey. With the combined input from this survey, opinions of specific individuals in the Student Committee and my own thoughts and opinions, we have a decent impression of what the student body thinks of the two bachelor programs.

To UiB, to the department of Informatics

The department of Informatics is not the only department offering IT related bachelor programs at the University of Bergen. The Faculty of Social Sciences offer a variety of programs and Bergen University College have a couple of Computer Engineer programs. When people choose to come to our department instead of any of the others it is mainly because of the hard science aspect. We offer more mathematics and theory. Many find this appealing because they believe it looks good to have a solid theoretical foundation. This will possibly open more doors regarding further studies and/or job opportunities. In the survey some also mentioned that they chose our department because of the selection of master programs and the information received about our department prior to applying.

When it comes to why some pick Computer Technology and some pick Computer Science, it is pretty straight forward. Those who pick Computer Technology do so because of the flexibility regarding mathematics courses. Also, the name and description give the impression of a more practical program which is appealing to those who know they want a job doing software development. Those who pick Computer Science do so because of the required mathematics courses and because of the impression that this program is harder theoretically.

When we asked the question; “Have you learned what you expected?” most people answered with a simple “yes”. But what kind of expectations do applicants have? Prior to applying, do they care which courses are mandatory and not, or about the selection of courses, or do they simply want to experience “the student life”?. The majority of first-time applicants are in the age between 19 and the early 20’s, and think in a completely different way than the people that create the bachelor programs. Most people that want to study something that has to do with IT might not care so much about which program they pick. This is why the name and reputation of a bachelor program is so important.

When a student at the department of Informatics

When already a student at our department, several students switch from Computer Science to Computer Technology or vice versa. They do not understand the difference before they have actually been a student for some time, and only then do they understand they should have picked the other program. Those who switch from Computers Science to Computer Technology do so because they find the mathematics load to overwhelming. Those who switch from Computer Technology to Computer Science do so because they find some of the mandatory courses included in Computer Technology boring and/or worthless. Should you actually need to be a student for some time before understanding the difference between the two bachelor programs? No, definitely not. The public description of the bachelor programs and courses need to be at a level such that the applicants understand what is expected from them if they are accepted into one of the programs, not such that they understand it after several months after discussing it with older students, learning the necessary terminology and so forth.

It was difficult to obtain a legitimate view on why students quit a bachelor program at the Informatics Department, by asking those who are still in the program why they have not quit. Most students at our department stay because they are happy with both the social environment and the courses. We get a lot of good feedback regarding the reading hall for bachelor students at the department, which is unique at the faculty. This works as an area for both social events and discussing courses and exercises. “You can always go there and meet other like-minded students” is a popular quote. If we could get every student who chooses to end their studies at the department to answer a couple of questions about the reason for quitting, we could get some really valuable feedback.

2. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science by Association for Computing Machinery (ACM) and IEEE Computer Science

2.1 Summary of CSC13

As mentioned in the introduction, the two US computer organisations ACM and IEEE Computer Society have for many years produced reports on what should constitute a good computer science education. The latest report CSC2013 (Computer Science Curricula 2013) is from December 2013 and is available from **http://dx.doi.org/10.1145/2534860**.

CSC2013 divides the computer science area of knowledge into 18 areas:

- Algorithms and Complexity (AL)
- Architecture and Organization (AR)
- Computational Science (CN)
- Discrete Structures (DS)
- Graphics and Visualization (GV)
- Human-Computer Interaction (HCI)
- Information Assurance and Security (IAS)
- Information Management (IM)
- Intelligent Systems (IS)
- Networking and Communication (NC)
- Operating Systems (OS)
- Platform-Based Development (PBD)
- Parallel and Distributed Computing (PD)
- Programming Languages (PL)
- Software Development Fundamentals (SDF)
- Software Engineering (SE)
- Systems Fundamentals (SF)
- Social Issues and Professional Practice (SP)

The report suggests how much from each area should be present in a computer science bachelor and master education.[1] It also differentiates between ”need to have” education (called Tier 1) and ”nice to have” (called Tier 2).

The committee feels that the CSC2013 report offers very valuable tools for evaluating and planning a computer science program. It should not be followed blindly, but provide a basis for our assessment.

[1] CSC specifies the amount of education in “lecture hours”. It is not evident how this relates to the Norwegian ”studiepoeng” (sp), but, based on the total amount of hours in a program, we estimate that 3 lecture hours are equivalent to 1 sp.

2.2 Our programs in light of the guidelines

If we consider today's courses offered at II (and also relevant courses from UiB and HiB) we find that most areas are well covered:

		AL	AR	CN	DS	GV	HCI	IAS	IM	IS	NC	OS	PBD	PD	PL	SDF	SE	SF	SP	
Obligatoriske emner i *Datateknologi*																				
INF100	Grunnkurs i programmering															10				
INF101	Videregående programmering															7	2	1		
INF102	Algoritmer, datastrukturer og programmering	10																		
INF111	Introduksjon til informasjonsteknologi						1	2	2									3	2	
INF112	Systemkonstruksjon																10			
INF142	Datanett										10									
MNF130	Diskrete strukturer				10															
DAT103	Datamaskiner og operativsystemer		5									3		1				1		
Obligatoriske emner i *Datavitenskap*																				
som *Datateknologi* uten INF111+112 men med																				
INF121	Programmeringsparadigmer														10					
Valgemner																				
INF143	Sikkerhet i distribuerte systemer							10												
INF170	Modellering og optimering	2			8															
INF207	Sosial nettverksteori				8														2	
INF210	Datamaskinteori				10															
INF219	Informatikkprosjekt																10			
INF220	Programspesifikasjon				10															
INF223	Kategoriteori				10															
INF225	Innføring i programoversetting														10					
INF226	Programvaresikkerhet							10												
INF227	Innføring i logikk				10															
INF234	Algoritmer	10																		
INF235	Kompleksitetsteori	10																		
INF236	Parallellprogrammering		2											8						
INF237	Algoritme-engineering	10																		
INF240	Grunnleggende koder							10												
INF244	Grafbasert kodeteori							10												
INF246	Informasjonsnettverk							10												
INF247	Kryptologi							10												
INF251	Grafisk databehandling					10														
INF252	Visualisering					10														
INF270	Lineærprogrammering	5			5															
INF271	Kombinatorisk optimering	10																		
INF272	Ikke-lineær optimering	10																		
INF280	Søking og maskinlæring								5	5										
INF282	Bioinformatiske metoder I			10																
INF284	Bioinformatiske metoder II			10																
Valgemner ved SV																				
INFO103	Informasjon og kunnskap								5											
INFO110	Informasjonssystemer								10											
INFO115	The social web																		10	
INFO116	Semantic technologies				10															
INFO123	Utvalgte emner i databasehåndtering																			??
INFO125	Datahåndtering								10											
INFO207	Sosial nettverksteori				10															
INFO214	Systemutvikling																			??
INFO216	Advanced modelling				4				6											
INFO232	Logikkprogrammering				2				2						1					
INFO262	Interaction design																10			
INFO282	Knowledge representation and reasoning									10										
Valgemner ved Høyskolen i Bergen																				
DAT101	Modellering og databaser								10											
DAT104	Systemutvikling og webapplikasjoner										3						7			
DAT105	Programmering i C++ og algoritmer														2	8				
DAT152	Videregående webapplikasjoner										2					8				
DAT153	Mobile og distribuerte applikasjoner												10							
DAT154	Programvarearkitektur og applikasjonsutvikling															10				
DAT155	Grafiske metoder					10														
MOD250	Avansert programvareteknologi																	10		
MOD251	Moderne systemutviklingsmetoder																10			
MOD252	Agentteknologier													10						
MOD350	Modellbasert programvareutvikling																10			
Sum		67	7	20	97	30	1	62	50	15	15	3	10	19	23	43	59	15	14	
Anbefalt i «Lag 1»		6,3	0,0	0,3	12,3	0,7	1,3	1,0	0,3	0,0	1,0	1,3	0,0	1,7	2,7	14,3	2,0	6,0	3,7	55,0
Anbefalt i «Lag 2»		3,0	5,3	0,0	1,3	0,3	1,3	2,0	3,0	3,3	2,3	3,7	0,0	3,3	6,7	0,0	7,3	3,0	1,7	47,7
																				102,7
(Alle verdier er angitt i studiepoeng.)																				

Detailed View

Examining our course offerings in detail, we see that a large part of the recommended curriculum is covered (often to a larger degree than the minimum required), but there are also some holes in our curriculum. Our 200-level and 300-level courses often cover advanced topics within the CSC2013 areas. Some of these courses are also taken by our bachelor students, while others are difficult to use in a bachelor degree, either because they're too advanced or have irregular scheduling.

Basic programming and algorithms (SDF, AL):

Our courses are INF100, INF101 and INF102. Together, they cover:

- all of SDF (Software Development Fundamentals);
- a little bit of PL (Programming Languages), mainly object-orientation and basic type systems;
- much of AL (Algorithms), with the possible exception of some algorithmic strategies (covered in INF234), and some automata things (covered in MNF130)
- some parts of SE (Software Engineering)

Discrete Structures (DS):

MNF130 covers all of the recommended DS curriculum, and more.

Computer Networks (NC):

Our course INF142 covers the entire recommended NC curriculum. Additionally, the social networking elective is covered by INF207.

Security (IAS):

We offer INF143 (security in distributed systems) and INF226 (software security). INF143 covers a small part of IAS (Information Assurance and Security); together with INF226 we have full coverage, but INF226 is irregular and elective. Additionally, we offer advanced courses, covering subjects beyond the scope of CS2013.

Programming Languages (PL):

The basic courses cover object-orientation. INF121 partially covers functional programming and logic programming. Topics related to semantics, compilation/interpretation, advanced constructs and advanced type systems are poorly covered (though much of this is optional in the CS2013 recommendation).

Computers and Operating Systems (OS, AR, SF):

Our offering here is DAT103 from HiB. It seems to cover:

- parts of OS (Operating Systems), but missing realistic treatment of concurrency;
- most of the important bits of AR (Architecture and Organization), possibly without a realistic treatment of modern memory architecture (and definitely without modern assembly language);
- small parts of SF (System Fundamentals). Other parts of SF are covered by MNF130, INF210 and INF236. If we stick to our 100-level courses, we're missing roughly half of SF (15 hours).

Software Engineering (SE):

The basic concepts are covered in INF101; much of the rest is covered in INF112. Some parts are also

covered in Master-level courses at HiB. Missing, are small parts on software evolution and software reliability. Also, students on the applied track should probably have advanced testing and software quality (from SWEBOK), which is poorly covered.

Databases (Information Management – IM):
Not covered currently at the department, but there are courses at INFOMEDIA (INFO125) and HiB. The CSC2013 recommendation is weak on this; most likely it is better covered in SWEBOK.

Parallelism and concurrency:
The INF236 elective covers much of Parallelism and Distributed Systems (PD). Proper concurrent programming is not covered currently; this is tricky to get right so students should get a solid foundation at the University (CSC2013 has weak demands here).

Human-Computer Interaction (HCI) and Graphics (GV):
HCI is poorly covered by our current portfolio. While we have excellent advanced visualization courses, we are weak on the basics of GV (media applications). Basic GV is only 3 hours, and we can probably expect students to figure this out on their own. HCI is also small (8 hours), and we have no staff dedicated to this, so we may have to live with not offering it.

Social and Professional Issues (SP):
A few of the topics seem to be covered by INF111.

Computational Science (CN):
Has moved to, and is covered by the maths department. Not relevant for our degrees.

Intelligent Systems (IS):
Is at INFOMEDIA. Not relevant for our degrees.

Platform-Based Development (PDB):
Is purely elective. INF226 probably covers the web platform bits.

Overall, the main weak spots are:

- Concurrency / multi-core programming for application (i.e., not using parallel algorithms). This includes synchronization primitives, locking, memory coherency, threading in common languages, how to reason about concurrency, how to debug.
- Software testing and quality – other than unit testing. For example, integration testing, functional testing, user testing, maintenance, evolution, and so on. Though we could reasonably say that some of these things are Master-level topics.
- Programming languages and semantics: practical functional programming; learning the exact meaning of language constructs and how to reason about them; advanced generic type systems.
- Databases.
- HCI. Basics of building user interfaces and interactive programs; user testing; accessibility. (INF111 used to cover a lot of this.)

- Social issues. Copyright, licensing, open source, labor rights, cost of bad systems and bad management. (INF111 covers some of this.)
- Hardware-oriented parts of AR and SF – but we can probably choose to skip this, since we offer software-oriented degrees.

The curriculum in a social context

As a research university, we have a responsibility to educate new researchers, but also to provide a general professional education according to the needs of society and the students themselves. As researchers ourselves, taking care of the former comes naturally, and this is reflected in the wide range of advanced courses we offer (three times as many 200/300-level courses as 100-level).

Part of the motivation for evaluating ourselves against the CSC2013 recommendation is to ensure that we offer a good and relevant professional education. Most of our students, even those with specialized master degrees, will go on to do professional work in industry; hence we should try to cover the fundamentals to the best of our ability.

Some topics are probably outside what we can comfortably offer. For example, hardware-related matters, artificial intelligence, natural language processing and numerics. Fortunately, such degrees are offered by our sister departments at the University, and by the University College. Thus, we can safely focus on offering good degrees in computer science and software development.

Most of our weaknesses are in software-related areas. Many or most of our students end up directly or indirectly producing and maintaining software, so this seems particularly relevant to our social responsibility, especially given the high cost of unreliable and insecure software, and failing IT projects.

Of the listed weak spots, concurrency is becoming very important, with practically every computer and portable device having multiple cores. CSC2013 is perhaps not as aggressive on this as it should be, so this is an opportunity for us to be ahead of the curve for the next few years. The points on testing, quality and programming languages are highly important for developing reliable, robust and secure software. We should also try to give students a working knowledge of databases and HCI, as this will be relevant in many jobs, but given the focus of our staff, we should not expect to be able to educate experts in these areas.

3. Changes in our bachelor programs as a result of the work

3.1. Bachelor program in Computer Science

The changes are:

- Replacing STAT110 (Introductory statistics) by INF122 (Functional programming) in the third semester;
- Replacing INF121 (moved to the third semester) by any of the courses INF226/INF220, INF234, INF240, INF252, INF170, or INF280, in the fifth semester. The idea is that the student can choose a course introducing her/him to one of the research groups at the department, and thus get a flavor of possible master programs.

In the new proposal, we have kept MAT111 (Calculus I) as mandatory in the first semester. Students in the Bachelor Program in Computer Technology normally take MAT101 (User course in mathematics). It can be argued that this is also feasible in the Computer Science program, since no mandatory courses have MAT111 as a prerequisite. On the other hand, with the current proposal, the first semester is more consistent with the theoretical and mathematical profile of the program.

The changes are motivated by a desire to:

- Have a clearer profile, separating this program from the Computer Technology program.
- Motivate the students to explore our research areas as early as possible, so they are better informed when making a decision about their master's.

3.2. Bachelor program in Computer Technology

The program in Computer Technology is aimed at students that want a general education in computer science and technology, with a practical rather than theoretical focus. The degree should give them the necessary background for entry-level work in industry and public sector (e.g., as a junior developer), or for pursuing a Master's degree in Computer Science or Software Engineering. Rather than be the “lightweight” program, the intention is to make solid, programming-oriented program fostering excellence in applied computer science.

The changes are:

- Replacing INF111 with INF115 databases and modelling, in the second semester.
- Replacing STAT101/STAT110 with INF122 Functional programming, in the third semester.
- Replacing INF142 with INF222 Programming languages, in the fourth semester.

- Replacing two electives with INF226 Software Security and INF214 Concurrent Programming, in the fifth semester.
- The last semester consists of electives so that students can study abroad if they wish so.
- Since this is broadened and more applied, suitable for a practically-oriented degree, MAT 101 is selected as the compulsory math course the first semester.

The changes are motivated by a desire to:

- Cover holes with regards to CSC2013 – particularly in concurrency, programming languages, functional programming. The choices for this program should fill most of the holes, and put us well ahead of the recommendations in important areas such as security, reliability and multi-core programming.
- Provide a modern, software engineering-oriented degree for students that plan to work as developers and consultants.
- Give a broader programming profile, making candidates even more desirable for industry. ("Would I hire this candidate as a developer?")
- Have a clear profile and make it easier for students to choose.

3.3 Bachelor program in Computer Security

In addition to changing the study plan for the existing bachelor programs in Computer Technology and Computer Science, we have made two new bachelor programs, in Computer Security and Bioinformatics, which will both start autumn 2015.

Why a bachelor in Computer Security at UiB?
The program can attract students to the department of informatics in general and to the research group working with these issues in particular. Since Gjøvik University College is the only educational institution in Norway with a similar offer today, we can get students with interest in this field from all over the country. We can provide this group of students a bachelor program with academic depth within the special fields in our department. The department should also take use of the existing teaching capacity that is located in the Selmer Center and in the department as a whole.

A bachelor program in Computer Science can also increase the number of applicants to the department's master programs.

Goals of the program
Computer communication is an essential and critical infrastructure for modern society. If this infrastructure doesn't function as optimal as it should, it can have severe consequences.
Defects and deficiencies can occur because of accidental events and ill-natured attacks.

The bachelor program in Computer Security studies methods for developing, implementing and analyzing communicational infrastructure that is robust to threats.
The object of this program is to give a solid background for actual work within this field. The

methodological approach is rooted in natural sciences and it is necessary to understand research results in both informatics and mathematics. The program will educate candidates with technological qualifications in secure and reliable communication, and the students will be eligible for further studies in computer science.

3.4. Bachelor program in Bioinformatics

Why a bachelor in bioinformatics at UiB?

The recruitment of good bioinformatics PhD students to Bergen has been and continues to be difficult. Having no bioinformatics master students has become the norm. These circumstances make it exceedingly difficult to maintain internationally competitive bioinformatics research activity. A bachelor program in bioinformatics has the potential to build a pool of interested and well-qualified students, from the bachelor to the master to the PhD level.

The Research Council of Norway supports the establishment of a national research school in bioinformatics, and Bergen will be the leader of such a research school. Bergen is a founding member of the European Bioinformatics Research and Education Workshop (BREW) series and the national coordinator of the Norwegian arm of the European Elixir project. A bachelor program in bioinformatics can strengthen Bergen's position in research and education, both nationally and internationally.

Goals of the program

Bioinformatics is indispensable in modern biology and thus in the understanding of life and development and health and disease. As a consequence, bioinformatics in its supporting and enabling role is of particular value to society. The goals of the program are to educate students in the basics that are necessary to successfully apply bioinformatics methods to biological questions and to lay the foundation for master studies in bioinformatics.

What should we offer?

A strong manifestation of bioinformatics – one that we can offer at the Department of Informatics – has a solid basis in computer science thinking (eg. abstraction, algorithms) and functioning knowledge (eg. programming, software development). This computational basis needs to be complemented with the basics of domain-specific knowledge (eg. molecular biology, genetics), the basics of statistics (eg. hypothesis testing, p-values), and the established body of knowledge in bioinformatics (eg. dynamic programming, proteomics, genome- and transcriptome-analysis).

How do we do that?

The informatics department offers and will continue to offer an excellent program in computer science. In the bioinformatics bachelor program, we include existing computer science courses, in addition to new courses in bioinformatics. These have been developed by taking the existing bioinformatics courses (mainly INF280 and INF282) and carefully adapted them to the new program. We also use existing basic courses from other departments such as in molecular biology, genetics, mathematics, and statistics.

4. Discussion: First year programming course

4.1 INF100

As it is compulsory in all our Bachelor programs, taught in the first semester, and thereby the first experience most of our students get with computer programming, INF100 is a course of great importance. We believe that quality in teaching INF100 matters a lot to how our programs are perceived by our students. In this section we present suggestions for possible future directions of the course.

Current situation

The curriculum is today focused around object based programming, using Java as programming language. That is, the students learn how to use classes and objects in their programming, whereas the object oriented programming as such is deferred to INF101 in the following spring semester. The course has a practical orientation with 3 graded programming projects and 8 minor exercises. Five out of the exercises must be passed in order to pass the course. The projects count 30\% in the final degree, and the score on a final exam constitutes the remaining 70\%.

With some exceptions (such as 2013), the teaching responsibility for INF100 has for almost a decade circulated within a small group of 2-3 permanent faculty members. The same group has been responsible for defining the curriculum. To account for the heavy teaching load, they have been granted one semester without teaching duties after having taught INF100 twice (and some other course once). Besides, one PhD student is involved in designing projects and exercises, and in administrating the grading. Master students and Bachelor students in the final year work as lab assistants, and do the grading of compulsory work.

The exam has until the fall semester of 2013 been a written one (duration 5 hours) with the traditional paper-and-pencil format. Then the students could opt for a computer versions, in the sense that they could import the questions to and type their answers in a program editor with syntax highlighting. Compiling or running the programs was not possible on the exam. Other changes undertaken this semester include video recordings of the lectures, which were made available to the students.

Possible future directions

Department of Informatics needs clear directions for how to pursue INF100. Amongst other, this applies to the curriculum, selection of teaching personnel (lecturers and TAs), teaching formats, lab work, and evaluation/grading.
As a starting point for discussions, we give some justified recommendations in each of these issues.

Curriculum:

The current INF100 curriculum is generally appropriate as a 10 ECTS introduction to programming. Although other universities go further in their introductory programming courses, experience tell that the bulk of the students find the work load of INF100 heavy, mainly because of the many practical labs. After passing the course, the students should be well prepared for learning object-oriented programming in INF101.
It has been discussed whether the curriculum should be extended to contain use of pre-written test cases. Introducing the concept, and applying it on simple programming examples, might be good for

improving the students' capabilities in identifying and fixing bugs.
We recommend that we keep the 30 ECTS triple INF100 (introductory programming), INF101 (object-oriented programming) and INF102 (data structures and algorithms) in the first three semesters of our Bachelor programs. Replacing them by only two courses of 20 ECTS in total does not seem to be a good idea, as we doubt it would give all students sufficient maturity in programming.

The lecturer:
A permanent faculty member with strong and proven pedagogical skills should be assigned the INF100 responsibility. We do not claim that INF100 cannot be lectured well by temporary staff like postdocs, or people external to UiB. Since INF100 is a course of special importance, we recommend that the uncertainty involved in having inexperienced or external lecturers dictates that this should be chosen only in exceptional cases.
A committee consisting of faculty members involved in teaching the basic programming courses should be established. The committee will be responsible for the curriculum in INF100, INF101 and INF102, and for ensuring that the course contents are harmonized with each other. In particular, teaching and deadlines can be coordinated so that the first year gets a more coherent feel, rather than giving the impression that we are competing with each other for the students' attention and time. Also, the committee could ensure that the same terminology is used, make sure that appropriate tools are in use, and verify that the syllabus and the exam is in line with the curriculum. This may also be useful for other courses in the study plan (e.g., INF142 has had varying content over the years). If needed, Department of Informatics must give necessary compensations such that the burden of teaching introductory courses like INF100 does not become unreasonably heavy.

Laboratory work and programming projects:
INF100 should stick with its current content of practical exercises. When introduced to programming for the first time, the students need a lot of practical experience. Programming is a practical discipline, and knowledge to it is acquired by applying the learning-by-doing principle. Weekly exercises and graded programming projects should still be compulsory.

Exam:
Department of Informatics should try to make it possible to arrange INF100 exams involving program compilation and running. The format that was new in the autumn semester 2013, where all students were given the opportunity to type their answers on a computer, represents a major improvement over the paper-and-pencil format. We find that this can be improved even further if the exam resembles the practical situation that the students face in their lab exercises. We are however aware of the risk that when having to compile and run the programs, lots of stress can be created, details that do not reflect the students' skills can steal a lot of time, with the outcome that otherwise clever students get a low score. Hopefully, it will be possible to shape the exam in such a way that this risk becomes small.

4.4 Debate points for further development of our programs

Debate Point 1 – Switch INF101 and INF102 in the study plans for all programs
Why:

- INF101 teaches abstraction and engineering principles that are not used again in INF102
- Algorithms taught in INF102 would be useful examples in INF101

If we do the switch, INF102 will have to cover more of the Java language itself.

Debate Point 2 – Languages for the introductory courses
A point for discussion is whether to switch the language used to teach programming in the basic courses (currently Java), but it is not clear that there is a substantially better alternative available.

Still, if we want to switch, the following points should be considered:

- A simple language like Python may work well for INF100 and INF102.
- Scheme/Racket has excellent teaching material and learning environments available, and is in use at several US universities.
- Some universities (e.g., Utrecht, and (previously?) MIT) teach functional programming in the first semester, and the Java/OO in the second semester.
- INF101 should have a statically type language with generics.
- The most important thing is probably to pick something our lecturers are comfortable with, and use on a regular basis.

Debate Point 3 – Social Issues
We have some holes in our coverage of social and professional issues, particularly if we remove INF111 from the recommended curriculum. We could distribute these topics over several courses, ensuring that also non-computer technology students are taught this.

Particular points to consider:

- Basic ethics, plagiarism, citation (INF100?)
- Copyright, licensing and open source (INF101?)
- Cost of: unreliability (INF101?), insecurity (INF101?), bad project management (INF112?)
- Labor rights and responsibilities
- Accessibility

We may be able to use invited guest lecturers to deal with this. If included in other courses, the topics should be on the exam, so students take it seriously.

5. Epilogue

After the evaluation committee finished its work in April 2014, the report and proposals of the committee were discusses at the annual department seminar. These discussions were further developed and concluded by work groups at the department. The main result is that the two proposed new bachelor programs have been approved and will start in the autumn of 2015. Alterations to the two existing programmes will also take effect from the coming semester.

Regarding the drop-out rate mentioned by our student representative in the committee, the departed started a drop-out project in 2014, which so far has consisted of surveys to former students and round table conversations with active students. This project is still ongoing.

The department is currently developing the new courses that will be a part of our new study programmes. The discussion about our first year courses is ongoing. We look forward to see the consequences of the changes we have made, and continue to discuss the rest of the debate points from this evaluation.