Visualizing courses

Improved Tools for University Course Planning

Andreas Olsson
Uppsala university
Andreas.Olsson.9761@student.uu.se

March 21, 2012

Contents

Abstract

It is important for University students to be able to have a clear picture of their education, both what they have accomplished so far and what is ahead of them. Students at the technical faculty have a lot of freedom to study a great variety of courses.

But with great freedom comes great responsibility. The problem is the lack of information given to the students about their progress towards graduation.

This master thesis was made to find ways of visualizing the Information Technology program. The goal was to find visualizations that, at a later stage, could be made interactive and serve as a guidance tool for students.

The work resulted in various graphs that presents the study program, as well as some interactive visualizations made from an application that served as a proof of concept on how the tool could work.

1 Introduction

Being a student in the technical faculty at Uppsala university comes with lots of responsibility and planning. Responsibility in a way that all students applies to each single course themselves, and planning because of how many different ways the courses are connected and related to each other. To be able to study in a program at the faculty, with a specified orientation, students must know a whole bunch of things before starting to plan their education. The relation between courses, which ones are compulsory, what classification a course have, and much more.

This master thesis aims at helping out in this issue, by providing aid in the form of graphic visualizations that, together with an application can help students in getting a better overview of their education.

The first part of the thesis will go through the basics in course structure, relations and classifications, in order for the reader to fully understand the problem domain.

1.1 Background

The program directors at the technical faculty has a desire to present the programs and then visualize it in a graphical way. The ambition is to get a clearer overview of the study syllabuses in order to describe, debug and improve them. Describing them to students, in the beginning of their education, gives them a better overview of the years ahead of them. For debugging and improvement, this could be a useful way to find courses with dead end relations, compulsory courses that are not required at higher levels, and other unwanted issues.

1.1.1 A brief course overview

The rules at the technical faculty are generally that, after three years of studies, students can pick from a wide range of courses and create their own study syllabus for their remaining period of studies. But passing compulsory courses are not the only restriction they are dealing with. Students must take courses of certain classifications in order to receive their graduation. For example, some courses are classed as being advanced courses, and a certain amount of advanced courses must be passed for graduation. Both information and validation about these classifications sometimes vary, depending on what year you engaged your studies, what kind of exam you intend to take, or simply by who you ask. This undermines the ability to create a study plan that are valid through the entire education, and it generates uncertainty among the students in terms of what courses to actually take. Today, the correct information is very hard to find, which is one of the problems that this master thesis is based on, and tries to solve.

The following sections will further explain the concepts of compulsory, technical and advanced courses. Other classifications exists, but are left out since they are not crucial for the exam, as these three.

1.1.2 Compulsory courses

Compulsory courses makes up most of the first years when studying in a program at the faculty. In order to graduate, one must complete all these courses. This sounds quite fundamental, but fuzziness arises because of the fact that one course may be compulsory for one year cohort, and optional for another, and this is basically where the problem starts. The reality is not that all students always finish their education in the contemplated time. A major part of the students studies for a longer time than intended which means that some fellow students have different compulsory courses, within the same program.

1.1.3 Technical courses

Some courses grants something called technical points. These points represents courses of technical character, and a certain amount of these points are required for graduation. Information about this also vary from year to year and the education planning, where the information is supposed to be shown, contains some errors providing extra uncertainty.

1.1.4 Advanced courses

Advanced courses could be described as the next level of courses available to the students. In order to take these courses, a certain amount of basic courses needs to be completed, and in order to graduate, a certain amount of advanced courses is needed. As mentioned earlier, students generally pick their own courses after three years of studies. This is because after three years of completed studies, students in general have granted access to the advanced courses.

The complexity and vague course information is not just a student problem, but also an administrative one. As a program director in the faculty, good overview and relationship between courses are essential to substantialize the program. Getting a clear structure and overview also helps others to understand how programs are constructed and what possibilities they bring. Something that could be used for future students.

This master thesis is based on these issues, projected by program director at the computer and information engineering program at Uppsala University in the spring term of 2011.

1.2 Objective

The objective of this master thesis is two sided. First is, analyzing and improving the original tool that is used today for visualizing courses. The goal is to concretizes what is actually shown, and put in comparison to what information that should be displayed. The thesis will examine different ways to visualize information and from that describe the methods best used for this domain.

Second objective is to build an application that can be used to build and configure and visualize study syllabuses, and from that present their essential information. Such information could include, the relation between courses, the direction of the syllabus, and progression throughout the period of studies. The progression is one of the challenges within this project. Today, there is no way to really visualize how knowledge gained from one course are used in subsequent courses. Lots of this knowledge are from laboratory practicals, with the intention to use later on. This application could be used as a debugging tool for these situations. Walking through a study syllabus one could point out where knowledge are left unused and regain the continuity.

Visualizing the outcome of the study syllabus could also help communicating the possible orientations the education has to offer.

1.2.1 Prior work

This project is based on an early project by Mattias Larsson, a fellow student at the computer and information engineering program. He created a Java Swing application for displaying relationships between courses, in a structure quite similar to the original study syllabus graph [9]. Some of these ideas are present in this project, but the main foundation is the substantial database structure that was created.

The database represents the relationship between course syllabuses, course instances, teachers, faculties (among many other things) in a way that is very close to reality. This saves a lot of time for this project since it provides a good structure on how things at university are connected. A good database structure of a system provides a lot more meta information than one might think at first.

In an early phase of this master thesis one idea was to simply continue working on the existing Swing application. But the job initiator pointed out that no special platform was desired. A web based application would be used to aid the graphs, basically since it would be easy to maintain and distribute. Creating a modular application is a good idea since the project time of twenty weeks would not be enough to cover this issue. A more realistic goal for this application would rather be as a proof of concept.

1.3 Current tool

Today, a graph editor is used to plan and visualize the study syllabuses. The editor, yEd, is a powerful tool for creating different kinds of graphs and diagrams. The downside to using such a powerful and complex tool all the unnecessary elements and functions. And despite its complexity, quite a few problems emerges. A tailor made application could provide a streamlined interface and better suited functionality.

The problems discovered with this tool can be classified into two groups. First, the work progress. How is the program to work in and what could, essentially from a GUI perspective, be improved. Second, what information can be derived from the graphs. How can it be used, and what conclusions can be drawn.

Figure 1.3 shows a slice of the graph, and the whole thing can be viewed in Appendix A.

1.3.1 Working with the graph

What becomes evident when working with a whole study syllabus in yEd is how hardware demanding it becomes. Trying to include all different courses in one tree structure would be quite heavy on most programs, so there is no say that yEd is particularly slow on this matter. The problem is the great amount of courses in one plain tree structure. The tool created for this thesis has no requirements on performance or that perspective whatsoever, but it should be able to scale down or isolate certain part of its corresponding structure in order to avoid resembling problems.

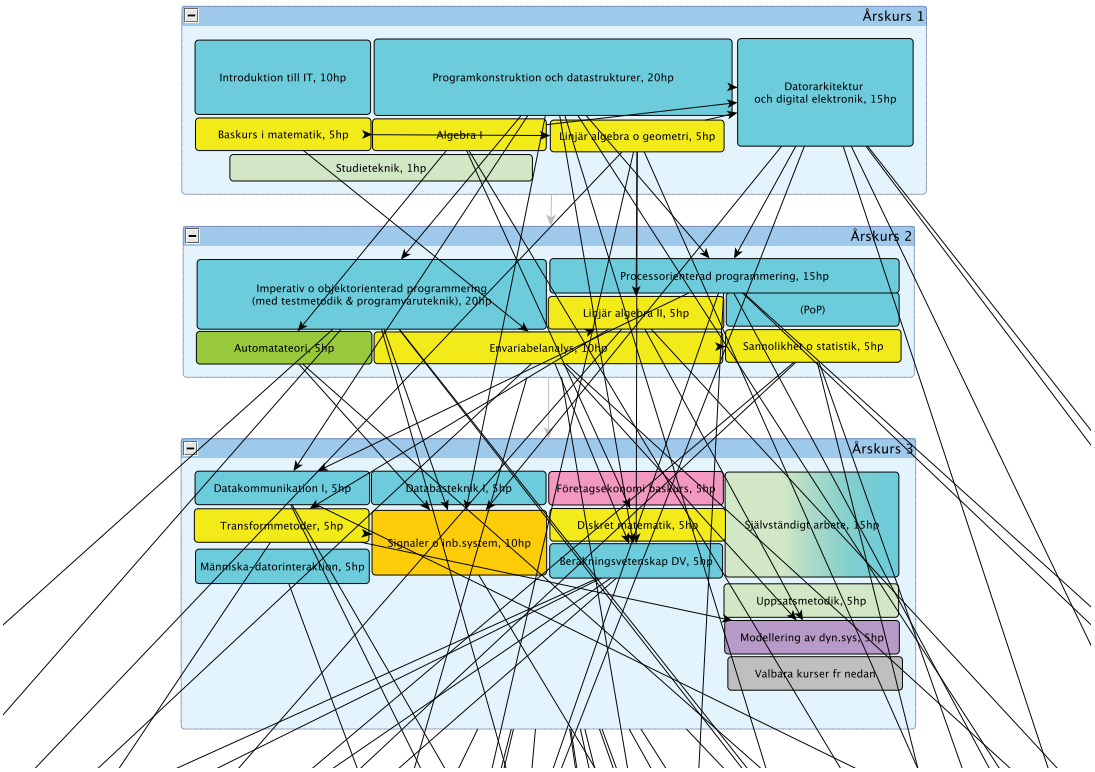


Figure 1: A piece of the current graph, created in yEd.

Another problem is that yEd isnt a very good tool for this specified task. Its a graph tool, which is fine, but graphs should be used for information visualization and they are just one way of doing that. A better approach to using a tool for this task would be to, first plan and build, and then visualize. Building in the visualization environment makes no real sense. Its a shortcut, where lots of crucial information is lost.

Also, planing with this tree structure makes little room for change. Given todays tool, its very hard to insert courses in the middle of the graph. Its quite like entering a new generation, between two existing, when making genealogical research. This is quite a large problem, since lots of courses changes from year to year. If the tools should be useful over a longer period of time there must exist a way to handle variations of the same study syllabus.

1.3.2 What you see is what you get

One of the fundamentals of this master thesis how to visualize course information and the current tool is the benchmark of this. From this, its crucial to analyze what information the current tool really tells.

As mentioned earlier, the graph used today is a complete set of all the courses in the faculty. This means that it tells us all information at the same time, but despite being so huge it contains very little information. So basically, what you see is what you get, or even less than that. Each course holds the information of its starting period, points, classification and dependency to other courses. These are the basics, but there are no correlation between them. No real conclusions, for example sum of points over a period of time, can be made from this graph. In this sense, the information is very static, and from that, quite useless.

As of today, the graph is often displayed to students, showing them the different options and possibilities within the program. One recurring event is the program course exhibition, a service where all courses are displayed for students. At the exhibition students can learn more about courses, orientations and their own possibilities. This is a good arena for this tool where this information is displayed. The problem here is that the information isnt personal for each student, but gives all information at the same time. From the student perspective, one would like their own path through the graph, and their selected courses. This is not possible with todays tool.

In a similar way, one would also like the possibility to isolate a certain orientation, year or part of the graph for a closer look. Most students arent in their first years and have thus completed a major set of courses. From their perspective it would be more interesting to look at their remaining options, rather than their past.

Given the analysis of the current graph tool, the returning problem is that the graph displays all information at the same time, and still fails to show enough. Things like detailed information, isolation possibilities or personal planning isnt possible. One thing to keep in mind when trying to find new ways for visualization, is the realization of using only one graph. Perhaps a combination of different tools could be a better choice. Perhaps the five year program itself is to complex to visualize in just one tree graph.

1.4 Limitations

This section is used to concretize the extent of the project. The parts described below are all left out on the same ground, to maintain focus on the main issue. This project should be about planning, testing and visualizing study syllabuses. Focus should be on usability and information visualization.

The purpose of this section is also to put this single-driven project in comparison to real IT projects, as a way of personal education.

Some of these parts could be seen as future work, and by keeping that in mind, the system should be that modular to support (or at least not undermine) the the possibility to add such extensions.

1.4.1 Cost

Naturally, since this is a master thesis, cost can be completely left out. In reality, money is a major player when carrying out a project. On some level, cost could be reduced by creating a modular application with rigid documentation, but that is rather a technical goal than an economic one.

1.4.2 Database synchronization

Lots of information related to this project are stored in a database by the name Selma. Selma is the university education database, storing information such as course syllabuses, information on course literature, course moments, and much more [12].

Future ambitions for these kinds of applications would naturally be to provide seamless synchronization in order to keep information updated.

1.4.3 User management

This system will not support user management, but that is probably something that can be found quite high in the stack of future work. The application allows one user, preferably the program director, to create and edit study syllabuses for presentation. But in a longer run, that functionality should be used for students to create their own syllabuses, since it is a big part of the issues that are the basis of this project.

2 Theory

This section presents the theory used for this thesis. A lot of this part deals with information visualization and how it can be applied to the domain of this project.

One of the key point is to visualize the courses, their parameters and relations. Information visualization engages humans to create internal constructions in their minds [13]. What this basically means is that, with the use of pictures, humans can spot relations, draw conclusions and summarize large amounts of data. This cognitive activity cannot be displayed or printed. Computers can display information, but the information processing is an activity that goes on in the mind. Information is the key in this project. From a given course graph, information visualization techniques and terminology was used to create new ways to display the same information. An optimal visualization would provide as much information as possible, with a very simple structure.

Building graphs that corresponds to, and adds information to the original graph can be a time consuming process. In general, this process can include requirements gathering, graph building and user evaluation. Given a time scope of twenty weeks, methods that can speed up such a process are of interest.

2.1 Information Visualization Basics

Whats important to acknowledge when it comes to information visualization, is that it is more than just graphs and charts. In everyday life, people are projected to large amounts of different information, from different informatics, saying lots of different things. All, from incoming e-mails to commercials or street signs are information that are processed, analyzed, remembered or discarded. This large amount of information tells us two things. First, its not very hard to display information. It seems like everybody are able to do it. Second, the hard part seems to be on how to visualize it the correct way. It does not seem like a good idea to just blow things out of proportion with blinking lights, alerts, paper clips, and other things to get the user attention. A better approach is to find out how to show the correct information.

Pictures and sketches are one of the most used tools when presenting ideas to each other. Visual representation helps us in describing something, perhaps a relation, that is to hard or complex to describe with words. Its often used as a way of giving the user as much information as possible on a very limited space [10].

Information visualization is not a new phenomena. The most common example of early information visualization (which seems to be mandatory for all books within the area) is the famous Napoleon's March, shown in figure 2. Charles Joseph Minard, a french civil engineer, created a flow chart that described the loss for Napoleons army during the Russian campaign of 1812. Vastly simplified, the graph shows the path of Napoleon's army and how it decreased, visualized by getting thinner, during the march towards Moscow. The black part shows the retreat and it shows the loss the army suffered from marching out to the homecoming.

Minored's graph is widely used and has been dubbed "one of the best statistical graphs ever" [14]. That is probably the reason for serving as a fundament in a major of the literature in this area. The graph displays quite large amount of information in a pretty simple figure, which is something to aim for in this master thesis.

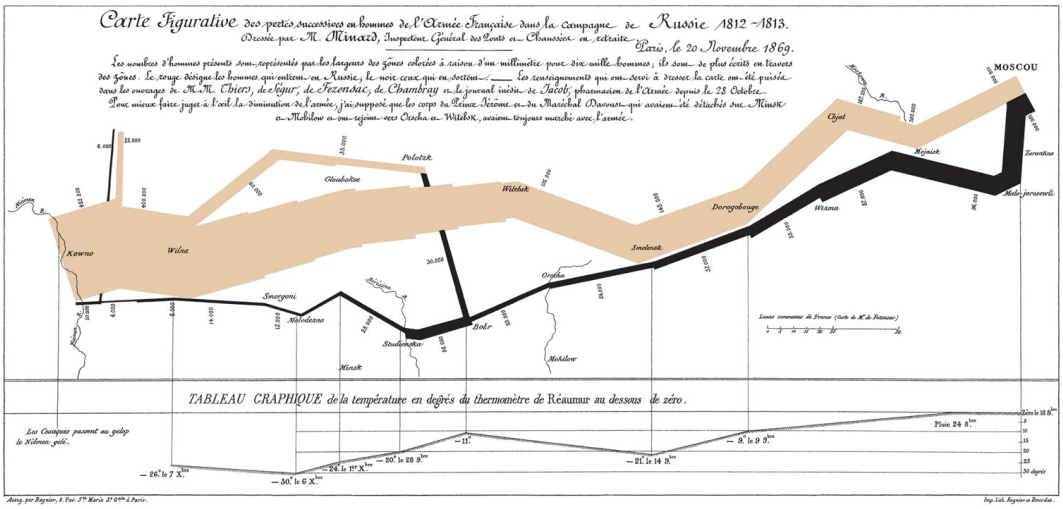


Figure 2: Napoleon's march

2.1.1 Explorative and Confirmative Analysis

Exploratory data analysis is an approach to analyze and summarize the main characteristics of data sets. In this problem domain, this could be summarizing the most evident information for each study period in a way that helps students in their choices. This is a well-used method for identification of relationships, patterns or properties among data sets. A common domain for this approach is by describing properties among the population on a map, for example electoral maps or the number of discoveries of a certain disease within a closed area. Explorative analysis was used when exploring new ways to visualize course and study information.

Visual analyzation also carries out confrontative analyses by looking at relationships. This is used to confirm hypotheses about future data. An often used example is the stock market. Looking at the correlation between two markets trying to predict how the affect each other.

2.2 Visualizations

As described earlier, information visualization is the process of forming a mental model of information. It is a cognitive activity, where processing of information goes on in the mind by forming images or mental models of the information. This is done by using something called mental models. A mental model is the explanation of the thought process about something in the real world. Its a way of describing the world around us. There are many examples of why the use of images are better then textual descriptions. First, the elements used in visualizations always have a physical localization. The relation between them, and their positions, can contain information. Second, using visualization minimizes the use of labels. Humans have better ability to remember shapes and forms, rather than text [4].

2.3 Processing the information of the current graph

The key problem with the original graph is that it does not support users to use mental models of the given information. It is hard to grasp the large amount of perceptual information that is presented. What this means in context of the course graph is that it should be able to produce information that is not directly displayed. An example of this could be telling information about when a course starts, by its position in the graph. The graph does not seem to provide any extra information than what is already printed on, and between, each course. At the same time, the most crucial information is also distracting.

The current graph basically provides four different sets of information. Course requirements and relations, course points and pace, when the course are held and the classification. Given the complexity of the graph, it is quite cheap on information. From this point of view, a number of questions occur.

- Can relationships among courses be described in another way than using large arrows? Using arrows to connect entities are by far the most common visual representation, and it is a straightforward one. The only problem is that the relation arrows catches too much attention. Usually when looking at courses, we are only interested in one relationship at the time. The current graph displays them all up front.
- What information is needed for the students? Should the graph contain more information, or provide the same but with a simpler structure? The most useful tools for students are the ability to see their required courses future possibilities, as well as the required number of required points for each classification. Most of this are not provided by the current graph.
- Is one visualization enough? Perhaps all information that is needed for the students can not be provided by a single graph. By separating the whole study program from an individual study syllabus, lots of different information can be reclaimed. This separation seamed necessary for providing a useful and somewhat dynamic tool.

2.4 Desired visualizations

Courses can be visualized in so many ways and tell different information. The key motivator behind choosing what visualization to use is to focus on supporting the users. Support for the students should be given in a way that makes it easier for them to grasp their educations, and support for program accountable in a way that it provides a good overview of the program and study syllabuses. To be able to debug what is planned, is one expressed requirements, prior to building the application. From two keywords, support and debug, the implemented visualizations is as follows.

2.4.1 Course graph

The course graph is, in many ways, the reason for this project to begin with. In that sense, it could be considered the main objective and as previously mentioned, this graph is an attempt to summarize an entire education by showing all available courses and their relations.

But one important question is whatever this is the best way to visualize information. To decide this, the graph is dissected and analyzed, in order to decide what information it actually tells us. Could that information be presented in a better way, and is there more valuable things that can be presented?

A major part of the work process embraces the concept of trial and error. By testing different ways of visualize the same information, new ideas and information can be extracted from that. From these ideas, new graphs is created, and so on.

2.4.2 Course graph meta information

Ideas of meta graphs, or supporting graphs is used. It is feared that not all information can be provided in one single graph, and it is therefore decided to elaborate with complementary visualizations. These can, among other things, tell such information as what type of courses are given, what skills are transmitted and what type of education forms are used.

Both students and program directors are target group for this kind of visualization. For the students, it may be interesting to see what type of profile their education are transforming into. For those in charge of planning, it may be interesting if the students are following the contemplated orientations, or if most of them are choosing their own path. Perhaps, new profiles could emerge from this.

2.4.3 Study syllabus feedback

Eventually, the goal of this project is to provide support for the students. And since students are given the freedom to choose most of their courses supporting graphs and visualizations within this area was also of high interest. The study syllabus graph should be used as a personal planer, allowing the students to get a clearer view of their past, present and future studies.

Feedback on the individual courses could provide useful information. Initially, its a good way of verifying that all mandatory courses within the program are taken. But also by showing what kind of education, in terms of skills and orientation, that is taken. Good visualizations could also help the students to orient themselves during their education. They could get a clearer picture on what to study in order to reach a specific course or education profile.

2.4.4 Study syllabus goals and progress

With the graphs that would display the individual study syllabus for each student, one of the key ideas is to show at which time in their education they would reach the critical points. Since the education requires enough technology credits, required courses and advanced courses, this visualization is one of the most important.

The idea was to show when, during their education, they would achieve this result, perhaps with a timeline where they could be highlighted.

With the use of a timeline as a reference, it would be possible to display the progress for each individual student, which would facilitate orientation. An indirect target for this is the ability to encourage students to give priority to the main courses that gives the desired points.

The graph of progress and goals will serve as an additional motivator for the students throughout the program and hopefully it can help to guide them when they seek new courses.

2.5 Information Visualization in twenty weeks

Information Visualization can be a rather complex. As mentioned earlier, it can be challenging to present the right information. Even though much time is spent during the pre phase of creating the visualization, it is no guarantee that we are successful in transmitting the right information.

As described earlier, the visualizations have a rather large target audience as both students and administrative staff are of interest. This would mean that the background research and requirements process can be quite rigorous. Given the time scope of twenty weeks and the ambition to present some kind of visible result, other methods are considered.

A method that could be used to produce graphs in a larger quantity is of interest. These graphs could then be matched to the goals of study syllabus feedback, goals progress and all other desired visualizations.

3 Understanding Requirements

Software Requirements, the general name given to the activities involved in finding out the requirements for a system and its functionality, is a large area within the human-computer interaction (HCI). It could easily engage a master thesis on its own, but since this thesis needed to produce some sort of product in a limited amount of time, this part of the project would have to suffer from cutbacks. Instead, the idea for this project would be to understand the importance of requirements elicitation and get practical experience of some of the methods that could be used. Before digging in to the definitions and methods used on this projects, some fundamentals should be straightened out.

In software engineering, the definition of a software requirement is a property that must be used to solve a particular software problem. What the problem are may vary at a wide range, but could be automation of a process or meeting security standards. It is not rare that problems rises from dissatisfaction of a current system, and the tricky part is often to verify the problem and transform it into a concrete requirement. Software systems where these parts have been poorly executed are often critically vulnerable [1]

Product and process requirements are also two significants that could easily be mixed up. Product requirements, which will be what were dealing with here, deals with requirements for a system or a product, while process requirements are linked to the developing organization. They are often connected to each other in such a way that process requirements can be used to specify how the product requirements will be satisfied.

Understanding the difference between, and specifying, functional and non-functional requirements are one of the first things to do. Functional requirements describes the specific functions of a software, or what the software should be able to do. Non-functional requirements deals with the surroundings, such as safety and reliability requirements. This project has only considered functional requirements. Non-functional requirements should be considered in later stages. If this software is to be used by a large number of students, performance will be a non-functional requirement. Also security will serve as a non-functional requirement if there should be a connection to the Selma Database, containing critical information about all courses.

3.1 Requirements Process

The requirements process is basically where all tools for gathering requirements are set. This stage includes defining process models, process actors, process support and management.

3.1.1 Process Models

Process models are mainly used for estimation, prediction, calibration, and optimization. What this basically means is that there are produced models for this four steps, within requirements engineering. Estimation is the concept of determining the values used in the observed space, whereas prediction has the goal to predict values of new user observations. The goal of calibration is to quantitatively relate measurements made using one measurement system to those of another measurement system. Optimization is used to maximize the outcome of a process and reducing the time spent on prefabricating the product at a later stage.

3.1.2 Process Actors

This topic of the requirement process introduces the roles in the requirements process. There are often many people involved, and often do they have different interest. A major part of requirements gathering is to define all roles and se how they are connected. It is not unusual that some roles have contradictory interests in a system, which needs to be sorted out and compared. In similar manner, some roles may have corresponding requirements, implicating that this may be something to focus on. Example of actors are often, users, stake holders, customers and software engineers [1]. In this project the client will serve as both user and stake holder.

3.1.3 Other topics

Other topics within the requirements process describes process Support and Management, which links process activities to costs, human resources, and other things. Along with process quality and improvement, they are more suited for large scale projects and are not used in this thesis.

3.2 Requirements Elicitation

Elicitation is the process of defining where the requirements comes from and how they are collected. Requirements elicitation is often the first step in understanding the problem domain and building the software to solve it.

The elicitation techniques that has been used in this project is interviews and user observation. Both techniques are well suited for projects where the users are quite a few in number. They are both time consuming, but very giving.

3.2.1 Interviews

The interviews for this project is conducted with the potential user, and client. The interviews shows how different it is to interview someone on the same side of the fence. Working with information technology and being at the faculty for a long time, the client knows more then well about the processes of eliciting requirements. This makes it quite hard to be in control, since the expert role shifted towards the interviewee, who always seems eager to "cut to the case".

All in all, interviews are one of the most basic techniques for eliciting requirements, and its still one of the best. Although some may say that it is time consuming it gathers a solid foundation of information to base the future work on. One great advantage with the technique is that it is simple and cheap [3].

3.2.2 User observation

What was discovered in this project is how user observation can serve new ideas. Observing how someone works, while describing what they are doing often raises questions on how things could be different. This is something that needs to be taken into consideration for the observer, not to ask questions or plant ideas in the workers head.

3.3 Requirements Analysis

Requirements Analysis is the process of analyzing the requirements gathered from the previous step. This is done to expose conflicting requirements and defining the boundaries of the system [1]. Conflicting requirements was not really an issue in the sense that different actors had conflicting interests. However, the work within this project needed to match the requirements for this master thesis. Setting the boundaries to suit a twenty week project was the bigger issue here.

Conceptual modeling is a method often used at this stage of the process. The conceptual model, which should be independent from design and implementation phases, is used to clarify terms and concepts that are often used within the work domain. This is way to sync the language between the involved parts of this project. For this project, not much effort was put in this section since all actors were on the same page. The conclusion that was drawn, after studying the idea behind conceptual modeling is that it could be used when at least one of the following conditions occur.

- If the problem domain is complex and the requirements analysts needs knowledge of special conditions, terminology or concepts.
- If the project involves a larger group of actors. This could easily lead to misunderstanding which could be avoided by syncing the terminology.
- If the system is at a larger scale. In this case using tools as Unified Modeling Language (UML) or Entity Relationship Modeling. These tools could be used to describe the problem domain in a graphical way.

Requirements classification is the process where we start to digging in to the requirements in order to sort them as functional or non-functional, product or process requirements, how they will change over time, among other things. In this project, this was a fundamental part as it had to focus on functional requirements that could be solved within the given time span. From the elicitation process, three main requirements or objectives were defined.

3.3.1 Debug

The tool should be able to use as a debugging tool when experimenting with study syllabuses. This is derived from the idea that each course within a program should have some meaning, prepare the student for coming courses, or be within a certain area of science. This tool should be able, from a given set of courses, be able to visualize their relations to make sure these conditions exists. If a course is presented as mandatory, but without any openings to new courses, it would be very hard to motivate its existence.

3.3.2 Plan

System users should be able to plan new study syllabuses. This affects teachers, course classifications, start dates, among other things. A fundamental part in ease the planning process is by giving feedback. More about the use of feedback within this project is described in section 5.2.6.

3.3.3 Visualize

A large area of use for this system will be to visualize courses to students and personnel at the faculty. This requirement derived mainly from the current tools, that was used to create a giant graph of all courses at the program. All in all, the giant graph provided the correct course information, among with their relations. But it lacked scaleability. All information was showed at once, and it was made worse by displaying it on a projector with limited screen resolution and inferior sharpness. This not only gave room for direct improvements, but opened the door to new ideas.

Visualizing courses, relations, progress can be done in so many different ways and towards different audiences. This area had the potential to grow out of proportion if not tighten at an early stage. Within the given time span, it would seem fair to experiment with some different ways to visualize courses, but further evaluations of this area should be left for future work.

3.4 Requirements Specification

In software engineering, requirements specification often means producing documents that could be reviewed and evaluated [1]. These documents records the system requirements which are often presented in a system requirements specification and a corresponding software requirements specification. These documents can be set in proportion to other variables of a software project, such as cost and performance. The relations are often done with help from quality indicators.

For this project, no documents on this level are produced since it is considered to be time consuming without giving any extra input. Instead, the documents used to represent this step are interview transcripts and ideas scribbled on notes and papers. The choice to not go through this step completely has much to do with the fact that there is really just on actor involved. Potential involvement of more could be possible when evaluating on how to visualize the courses. That could not be done until the next step.

3.5 Requirements Validation

The documents produced in the requirements specification is the foundation of the next step, the validation to requirements. This is the part of the process where the users and stakeholders are reconnected in prototyping, validation and acceptance testing.

3.5.1 Paper prototyping

Paper prototyping is one of the prototyping methods that seems most useful when designing a graphical user interface (GUI) for a system. This is a way for developers to validate the requirements that was set in previous steps. It seems like an interesting method since it is widely used, even after the introduction of different prototyping software tools. One key benefit to letting simple paper structures visualize a GUI is that it focuses on workflow and interaction, rather than design. If it is obvious that this is not the final product, the users (or who ever tries it) are less focused on design details. For this project it should work well enough when designing a paper prototype of the planning tool. This part of the system consists of simple and well known elements, such as forms, buttons and textfields, which could easily be transformed to a simple paper mockup.

But there are some when dealing with visualization. Presenting different types of graphs and graphical elements is not suited for the paper prototyping method. Some graphs should, for example, feature direct feedback on mouseover events, which is hard to implement on paper. Another problem is related to the kind of data presented. The problem is to create a simple paper mockup of a complex and interactive graph, that responds of user input. With these kinds of problems in mind, new methods of creating mockups should be considered in future projects.

3.5.2 New input from user

One interesting thing about paper prototyping is how the process seems to spark new ideas. The expected result is that paper prototyping should test the workflow from a settled functionality. But while walking through the process with the client, new ideas grow and is enthusiastically expressed. This could probably be more controlled if extended input is put in the preparation and implementation of the process. But there is one dubiety that sticks with the whole process, that could have something to do with how it seems to plant all new ideas. This insecurity is about how detailed the design mockups should be. The common idea is that the design should consist of rough sketches, just enough to project a clear image of what to come [3]. This is because the client should not focus on design details, such as colors and images. This is a good thought, but for this process it seamed to backfire. The image that this paper prototype process signals is that this is work in progress. By showing sketches of a future interface the client thinks the door for new input is wide open.

New input and ideas should never be considered a problem, but with a tight deadline and a problem domain that is in desperate need of constraints, there are few other ways of looking at it. The solution would be to have a simplified paper prototyping earlier in the requirements elicitation. One where the client can participate in the formation of the interface. This idea alone would be contradicting to the ideas of letting the expertise design the application and it is interface, but it can be used as a complement to extract more thoughts and ideas from the users.

3.6 Speeding up requirements

Gathering and understanding requirements are important. There is no doubt about that. But the complexity of the graphs, the large target audience and the ambition to produce a result in twenty weeks calls for faster methods. The target audience is more open in a sense that it doesn't consist of a closed workplace and the fact that the renewal of students is quite large.

Also, in a more distant future, a working web application may be a part of the Student Portal and used by hundreds of students. There is a risk that requirements gathered from interviews and paper prototyping only manages to pick up a limited amount of feedback on how the optimal application should work. When working with a larger target audience the idea of working with, and updating, a live version may be better. One can use the approach of rolling out different versions of the system, and then analyzing user behaviors on a larger scale. This way of testing is refereed to as A/B-testing and can be useful when determine the how the interface and application workflow should be.

With these facts, there is reason to evaluate how much of the project that should focus on requirements.

4 Methods

The methods used are basically focused on producing a visible result. Instead of User Centered Design (UCD), a more result-oriented approach is used. The ambition for the application is for it to serve as a foundation of further development. With that in mind, lot of work is spent on building it according to a well-used software architecture.

4.1 Result-oriented design

As described in previous section, the ambition is to produce visible result from a complex problem, during the limited time scope of this master thesis. The large target audience makes requirements hard to gather and grasp, and feedback can also be given during the production phase if the tool is a web based application. With these problems in mind, this project will use the concept of using an inspiration-based design instead of, the more commonly used, User Centered Design (UCD). The idea behind this is to try out a newer and supposedly more efficient method when designing graphs and the supporting application. UCD involves numerous iterations of analyzing and testing. The drawback of this method is that its time consuming, and not suitable for this master thesis. Trying out the UCD method, and use it the right way, could by itself engage a master thesis. Instead, the idea is to use this result-oriented method. Sometimes described as Genius Design, is somewhat of an opposite approach, relying on expertise of the team, without external user input. ITs significantly faster than UCD, but requires more expertise and experience.

The clearest benefit of using this approach is that allows fast production of graphs in a limited time scope. The graphs will be mockups created in large quantities, rather than of great quality. They will be iteratively improved. Some methods will however involve the user, primary during requirements gathering. Interviews and paper prototyping.

An application will be built, as a proof of concept. This application is web based and created according to the Model View Controller (MVC) pattern. The motivation behind using MVC is a mixture of personal development and actually build according to a structure that can serve as a API to further developments. The application will feature some planning possibilities with drag-and-drop functionality where the user can move courses between periods to create a study syllabus. The study syllabuses will be summarized in some graphs and visualizations displaying things like orientation, how courses are related and what types of courses that are significant for each period.

In a way, these tools also describes the three cornerstones of this project and sums it up towards the goal; building a web based application for planning and visualize courses, and doing it the right way.

4.2 Why web based application?

Building a web application seems like a good idea, essentially because of two things. The first being that web applications are a easy to distribute. In the context of this master thesis the application will be used by faculty staff when planning, debugging, and presenting the program. But a further goal would be to link this to the students in a way that they could create their own study syllabus. The other study related tools used by students are online (ping-pong, studentportalen), so introducing this one could be done simply by integrating it in a website.

Web based applications has increased enormously the last couple of years. Thanks to new standards, such as HTML5, CSS3, jQuery, among others, the limitations of what can be done with web applications has been significantly reduced.

Some of the downsides of this approach is outside the context of this thesis. They will only be reviewed briefly.

4.2.1 Security and personal integration

The security aspects of building a web based application is not considered in this project, but should be acknowledged in the future. There are lots of things to consider, such as SQL injections and other user input. Critical user information and passwords should be encrypted and database information should be secured [11].

4.2.2 Speed and performance

In terms of speed and performance, the rules that applies to web sites naturally applies to web applications. Since this also is outside the project domain, very little effort is given to improve performance. However some basic CSS rules is applied, such as using sprite images and reduced quality. No effort is putted in minimizing database requests which could probably slow down the system substantially.

4.3 Using a software architecture

Building a complete system of this magnitude in a time span of approximately twenty weeks is not a realistic goal. In this case, a major part of this project focused on pre studies,user requirements and user interaction. Therefore it seemed essential to use some kind of architecture to build the application on. A higher order structure would make it easier for others to adapt and continue working on this project.

A software architecture is basically what the name suggests, an architecture for the software. In general, it defines the structure of a system, its elements,their relations and their properties. The motivation behind using a certain structure or pattern are often based on design decision. For example, one would like to separate domain specific information from the presentation. This is actually the main motivation behind the software architecture that was picked for this project, Model-view-controller (MVC). But before diving into that specific architecture, we will look into the basics, the cons, and some different design patterns that adapts and concretizes the use software architecture.

The basic idea behind software architecture is structure and separation. Using a good structure provides better overview of a complex system, while separation allows for multiple elements to be worked on simultaneously. In larger project different teams often work on different parts. There may be user interface experts working on the front-end while data security experts has focus on back end. A good structure would allow teams to work on each end of a project while maintaing a common goal. Even if this particular project does not involve such magnitude, it shouldt throw a spanner into the works for future developers. Growth must be supported. Except from a smoother transition to other developers and modularity, using a software architecture has a additional advantages [An Introduction to Software Architecture].

High level relationships Its important to have a clear picture of the application structure in order to understand it is functionality, without drowning in details. This helps us in building new systems as variations on the current.

Establish the architecture before writing code Before digging in to writing code, the structure, functionality and relation between system components should be settled. By doing this it is possible to, at least in some sense, streamline the code and optimize the system. It can be used as a debugging tool on higher level, as illogic in the system is exposed. If the architecture is not established, there is a predominant risk of slow systems that is built on scattered structure and overlapping functionality.

Exposing alternatives Detailed understanding of the architecture also allows for system developers to consider alternatives before implementation has begun.

By establishing an architecture we present a template that tells us how to add new parts to a system. This creates the flexibility which basically is the motivator to use it in this project.

4.3.1 The Model-View-Controller Pattern

The The Model-View-Controller pattern is a software architecture, or a methodology, that presents a way of separating domain, presentation and user input elements. The first known description of the pattern was made in 1979, by the Norwegian computer scientist Trygve Reenskaug who wanted to present multiple views of of the same data. The pattern is based on three components Model, View and Controller, whose relation is presented in figure X. Over thirty years later, the MVC pattern is probably the widest used software architecture for web applications. Later, a web specific version emerged, which is described further in the next section.

The Model basically contains the building blocks that are used, such as data and functionality. For most web applications this means that this is where databases are included. The Model may use observer pattern to notify the other components data changes and the observer pattern is what actually what separates the View form the Model. The View and Controller maintains direct links to the Model and are all associated as singular entities. What this means is that each Model is associated to a single View and vice versa.

The View is where the material, or application content, is presented. In most web applications this would be where each HTML page is defined. The functions that are used to make the application interactive (a web application that only consists of static material isnt really an application) are defined in the Controller, which lets the other two components communicate with each other.

4.3.2 Web specific MVC patterns

Using MVC patterns in web applications is somewhat of a natural step since they allow sending dynamic content to clients by processing HTTP requests from the server side. As applications on the web became more advanced the need for a good software architecture increased. In order to process data and requests, a web specific MVC structure emerged.

When looking at the structure in figure X2, the most notably difference in web specific MVC is the Front Controller. The figure can be somewhat misleading in making believe that the Front

Controller is visible to the users. What this component does is handling incoming requests sent by the user. The Front Controller routes the incoming requests and then dispatch any actions. What this means is that it bulks all responses to a request and returns them when the process is completed.

The other parts are similar to the general MVC structure. To make a very simplified example of the web specific Model-View-Controller architecture in the domain of a web application, the Model can represent a database, the Controller holds all functions that may use that data and the View present the web interface to the users.

4.3.3 Other software architectures

In web application development, the web specific MVC seems to be, by far the most popular. But in order to get an overview of the alternatives, some other architectures is fairly examined. Most structures that is examined are basically variations on the MVC theme, and could probably be used. But having the large amount of documentation and help from different blogs and web forums, it seems like they would have a hard time in challenging the good old Model-View-Controller.

4.3.4 MVP

One architecture that, by the name sounds quite the same as MVC, is Model-View-Presenter (MVP). The basic difference is the Presenter which performs similar tasks as Controller in MVC. Also, the View deals with the UI events, which is typical Controller tasks, leaving the Model to strictly handle the domain (for example database connection).

4.3.5 MVVM

The other architecture, MVVM consists of the parts DataModel, ViewMode and View. This architecture is targeted towards rich UI applications, such as Microsoft Silverlight which is an application framework to run advanced internet applications.

Both architectures could probably be adapted to this project, but it seamed natural to follow the basic MVC structure.

4.3.6 Zend specific MVC structure

Before digging in on the whole Zend application framework, some things should be considered. The Zend framework features functions and concepts to help the architecture as well. This is the basic idea of having an application framework. The way for an outsider to continue work on this project is to follow the Zend specific MVC structure that is set by the application framework. This is one of the major advantages of Zend, that the architecture is so solid. By following the structure set by Zend, one would follow build applications according to the MVC structure.

4.3.7 Zend Framework

For the same reason as using a software architecture, an application framework is also a good idea when building this system. In short, the Zend framework is based on the MVC structure, so this is basically the next step in the process of deciding what tools to use. Skeptics to application framework often asks why they should bother setting up an environment and use predefined function in order to build PHP applications. There are two main reasons in this project for doing that. First, as nagged about in the software architecture section, it provides a good structure that is much easier to adapt for those who are introduced to the system. Zend offers tools for setting up the backend environment such as database connections and building functions from there. When new functionality are added to the system, its not hard for the developer to figure out how similar functions were made, and go from there.

Second, once the environment and backend are in place its quite easy to create new functionality to the front end. Part of this project is to explore different ways to visualize course data, and this is is often done from the same database tables. Once they are set, visualization could be done quite easy by passing JSON objects to jQuery functions in the front end.

4.3.8 Zend alternatives

Before digging deeper into Zend, we will examine the alternatives and explain why Zend was the winner. Most application frameworks are actually quite similar [2]. There may be difference in how certain semantics or design issues are solved but to most users it seams like a matter of personal choice and taste. Popular alternatives such as CodeIgniter [] lacks built in support for Ajax, while others only provides support for a single database, which didnt cut it if the Selma database is to

be included later on. All of the popular ones (CodeIgniter, CakePHP, Yii) are base on the MVC structure.

Zend comes with quite a rich community and some well known sponsors, such as Google [7] and IBM [8]. Since Selma is a IBM DB2 database that connection became the key part in settling for Zend. The Zend framework has embedded support for DB2 databases which could be useful in future versions of this system, where Selma is integrated.

Zend framework features some shell script commands called Zend Tool. These can be used to build create new Models, Views and Controllers, among other things. Zend Tool can be used to create a new project. When doing that Zend Tool creates a directory structure, controllers, actions, views, and some project details.

4.4 Creating an API

As described, the benefits of using the MVC structure when building this application is that the database is separated from the rest. This serves as a simple Application Interface (API) where functions are created to deal with the required data. The benefit of using an API is that it requires very limited knowledge of the surrounding operations. Generally, all that is needed is the right variables in order to extract the right data. When using a fully functional API, developers can focus on other things like dumping data into different graphs and visualizations.

5 Result

This section describes the result of this master thesis. The most interesting part is the different graphs created, which is summarized in the last part of this section. The application is used as a foundation for planning, where courses can be created, teachers and requirements added, and more.

5.1 Workshops and User Interaction

Workshops and seminars are mostly used, directed to the client while gathering requirements. At the end of the project, some simple types of seminars and open discussions are used among a small group of students to evaluate the compiled graphs.

It should be noted that limited effort was put in workshops, in favor of other priorities. One major reason for including this part is mostly for personal development.

Building a lucid graphical user interface is also part this project. Even though the application is proof of concept, building it the right way is still high priority. There are a great number of books, essays on what good GUI design is. And not the least, endless examples of both good and bad GUI on the internet. The interface for this applications is aimed at following some basic principles, as well as avoiding some common design mistakes [5].

A good GUI design should be built around the specified tasks of the user. The designer is, and should be, the expert, but there is a thin line from taking the overhand and steering the user in their direction. For example, some GUI designers wants to control the users navigation by graying out areas or disabling buttons. An application of this kind should be event-driven, meaning that the user is in charge of the events. Allowing too much user freedom may expose bugs and design faults, which can seem a bit scary to some designers [5].

This shows the complications in GUI designs, because another common design mistake could occur if the event-driven design concept is misinterpreted. The user should be given freedom, but not all at once. It may be tempting to include all functionality in one single view, which may cause the functions to be in the way of the applications functionality.

Other GUI design basics that is considered for this application are interface structure and the use of colors. How this was solved is described in section 5.2, which covers the structure of the Application, and the ideas behind them.

5.2 Application

This section describes the application that is built to support and visualize some of the ideas of this project. As mentioned earlier the original idea is to build a complete application within the time frame of the thesis. That idea is scrapped at an early stage. However, a simple application is built with the desire to build it in a decorous way. The ambition is to build it on well-used techniques, mostly for personal development.

The back-end structure used for this application is described earlier so this section focuses on front-end, such as elements and workflow, along with concepts and ideas.

The previous work that resulted in this thesis provided (among other things) a solid database that described most of the relations needed to build that application. It provided tables and relations for teachers, courses, institutions, programs and many more. This is all taken into consideration for this work since, even if it would turn out into a proof of concept, it should still be modular and something to work further on in the future. Therefore, some of the surrounding information that could be stored in the database is implemented and can be managed through the application, such as managing teacher information, institutions and programs. Some would probably argue that this information is in the periphery of the actual subject. But all information that could be visualized or compiled should be taken into consideration. Imagine the future possibility of a graph tool that could easily describe what available teachers there are for the following year, or charts that can tell which program has the highest rated courses?

This section will describe some of the key elements in the application, forms and draggable boxes, as well as workflow and feedback.

5.2.1 Forms

This application will deal with quite large amounts of data, from detailed course info to surrounding data about teachers and tags. Even though a feature vision for this is a seamless synchronization with existing databases, the current version will rely on using forms to provide the correct information. A typical form in the application can be viewed in figure 5.2.1.

Using form has always been a keystone in modern interfaces and todays users are quite confident using them. Being confident using a form is not only regarding security aspects, but also usability aspects. Nowadays, most people are familiar with basic text boxes, drop-down menus and checkboxes.

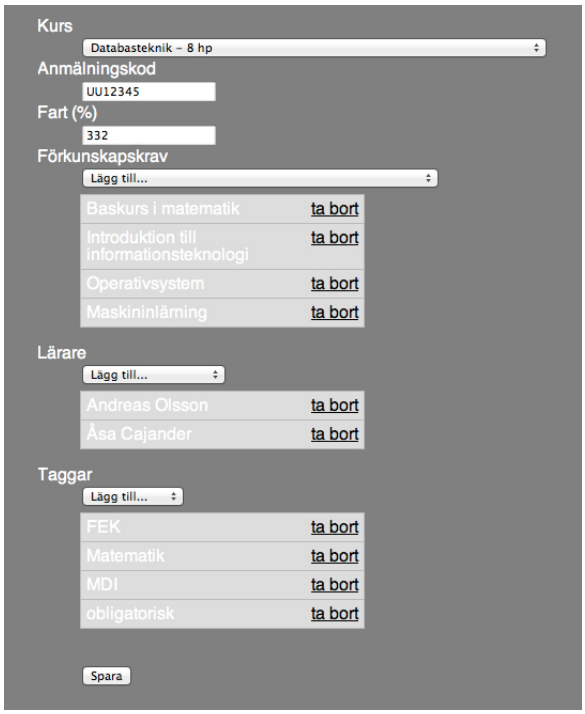


Figure 3: A form in the application where course instances are handled. For each course there is the possibility to add tags, prerequisite and teachers.

A natural conclusion of this would be that designing forms nowadays are a lot easier than before. But one particular side effect of being more comfortable with is that users are more demanding than before. They expect smooth flows and are less willing to tolerate bad forms. Creating a good form has moved from being about providing comfortability and gain the users trust, to be more directed towards smoothness and quick feedback.

One interesting theory about using forms (or rather why using them are so rejected) are elaborated in the book "Forms that work" [6]. Here, they are looked at from a plain usability perspective. The forms were dissected and pieces such as fields, text areas and overall structure were examined in order to understand the difficulty with using form. The theory that emerged divided forms into three layers: relationship, conversation and appearance. The basic idea is to pinpoint what the form communicates. Within this project it is discovered that the idea of using these layers could be applied, but has grown rather old. The key difference between the theory and the discoveries within this project is, as described above, what people expects from forms. The security issues has stepped aside, on behalf of feedback and smooth transitions. But before twisting their theory, lets examine it.

5.2.2 Relationship and reaction

The first layer is describes the relationship between the form taker and the organization asking the question. A lot of this parts focus on how to establish trust between these parts. This is, without a doubt, both an interesting and valid approach. But this layer assumes one thing, that forms are external, which is a very narrow way of looking at form. For example, adding an event in a basic calendar application (i.e. iCal) or sending an email requires filling out numerous text fields. That should be seen as a form. And the relationship is a non-issue after several times using it.

A more interesting approach to this layer would be to look at it from a reaction perspective. What happens when a form is filled? What kind of response is given? In the given theory, the relationship layer also includes a reward perspective. What reward is given when filling out a form. What is rather sad about this point of view is that it has the base idea that we think of filling out form as something we would rather get rid of. It may be a bit too ambitious trying to design a form that people are dying to fill out. But a good goal in designing forms should be this: do not let them realize that they are filling out a form.

5.2.3 Conversation

The second layer, conversation, is also quite complex for this project. But the keystones should definitely be included. They deal with what different elements, and their positioning, are communicating.

One interesting approach is to try to avoid unexpected elements, for example by using multiple steps. A good form quite easy to overview.

5.2.4 Appearance

The appearance layer deals with what the forms look like. It is not quite the same as conversation (even though simpler forms may allow these layer to overlap). Research has been made with eye tracking technology proposing that labels should be placed on top or on the left of fields. Placing the label at the left of the field requires less eye movement, which is good. But it is not the only consideration. For example, drop-down menus could work better with labels on top of the elements, depending on their width.

Other appearance considerations does not vary much from regular GUI rules, such as coloring and structure.

5.2.5 Other considerations

The overall difference, from the more complex theories about user input [6], is that this application mainly uses is smaller, but recurring, forms. Some ideas that is brought up considers how to get users to understand and fill out more complex forms. The interesting question in this case is rather how the structure should be suited for returning users.

There is one form that could not be left out in this project. It is both extremely simple, and extremely reused: the Google website. Since Google has billions and billions of users every day, it is quite easy for them to create big case studies, simply by tweaking the design for a limited time [15]. But what they are aiming for really interesting elements, making it quite fun to fill out the Google form.

- Quick response: The new instant search...
- Suggestions:
- Smooth navigation: Tab navigation

These are three keystones that could give forms a more interactive experience.

5.2.6 Drag-and-drop

The section where study syllabuses are planned is without doubt the key of this application. The concept is to let the user construct study syllabuses by dragging and sorting blocks of courses. These courses can be picked from a drawer in the edge of the screen. The concept of drag-and-drop is to let the user recognize the concept of moving a physical object from one place to another. The interessting part of this is how widely used this is, considering that there are easier and more effortless ways of moving objects. Dragging requires more physical effort than moving the same pointing device without holding down any buttons. Because of this, a user cannot move as quickly and precisely while dragging. A screenshot of the application can be viewed in figure 5.2.6.

5.3 Analyzing and rebuilding graphs

With all the information that is to be visualized it is clear that a multiple set of graphs should be used. Simpler structure of the graphs supports the process of forming mental models of information. Also, multiple graphs within the same area is a way of analyzing and testing them on students. The whole process of creating graphs are quite open-ended in that sense that there are not a finite set of graphs to create. Many of the graphs created ultimately displays additional information than originally planned. They may, for example, show information on how courses are related and how they are planned in relation to each other.

5.3.1 Course graph

The program graph, used to visualize the entire program, should be based on the information from the old graph, but with enhancements. The basic problem with the old course graph is that it tries to bring too much information which made things messy. From the information given in the original graph, new ones are created.

One major inconsistency with the current graph is how time is represented. The basic structure is built upon classes, creating a five level tree graph. This suggests that the time line is represented by the y-axis. But for each level, each period is represented by a time line along the x-axis. This may be a logic way of describing both year classes and periods, but brings forth the following problems.

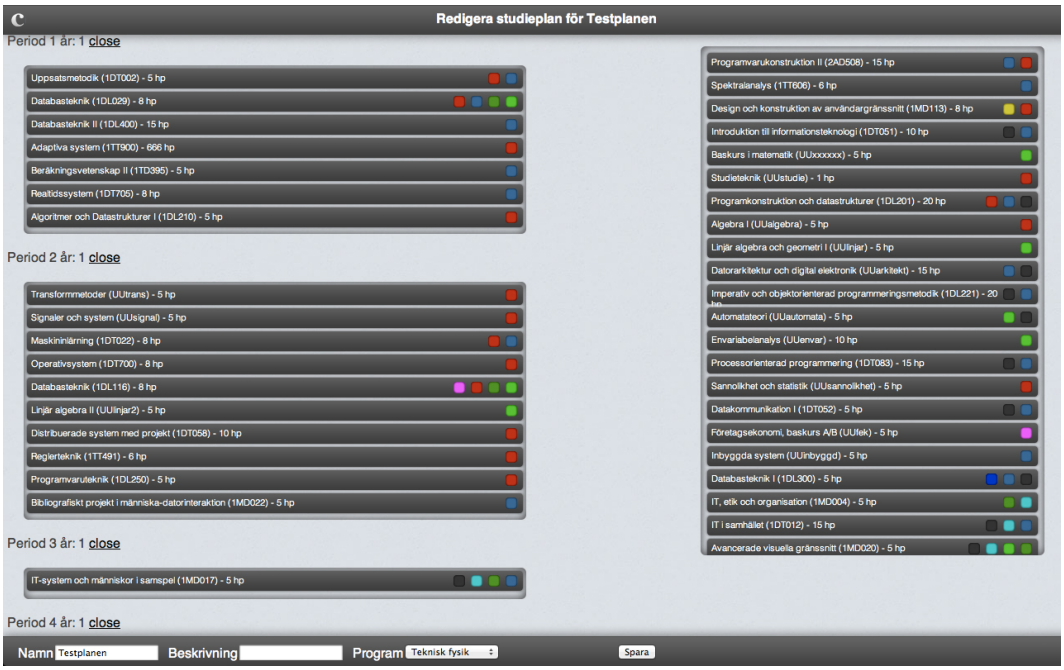


Figure 4: The planning view of the application

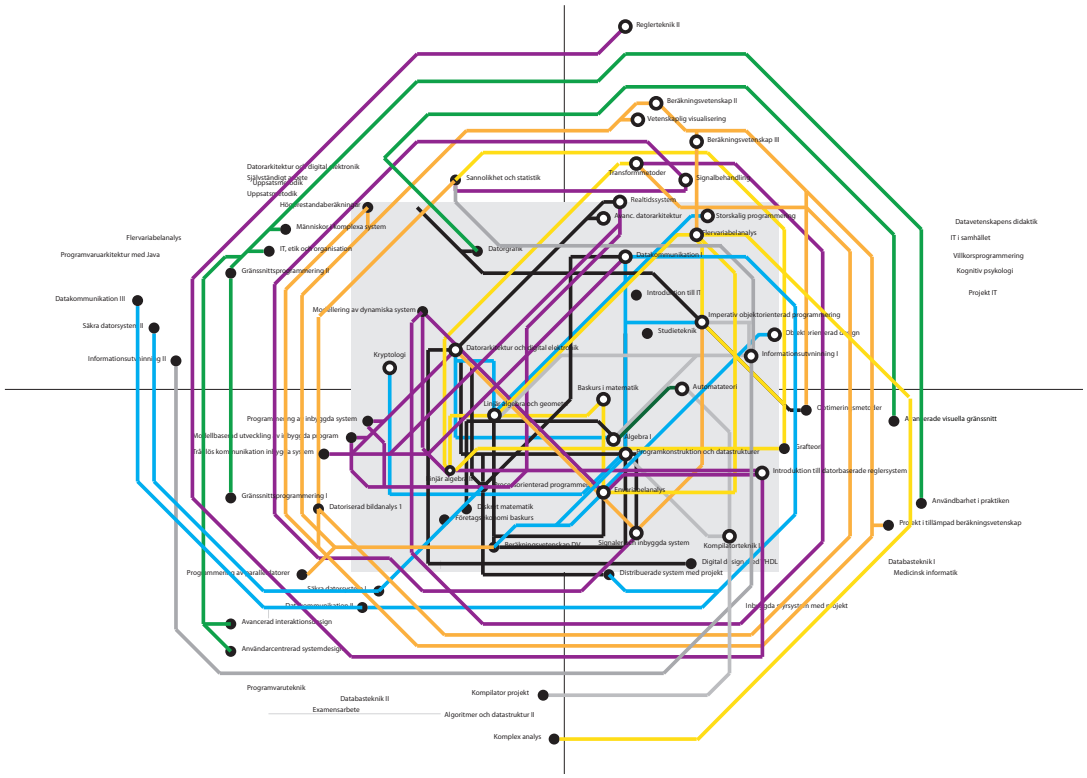


Figure 5: The first visualization that uses the idea of emulating a subway map. This graph places all courses in four time zones, representing the four study periods. The courses in center represents mandatory courses. The ambition is to create a graph that grows from oligo into different branches for each program profile.

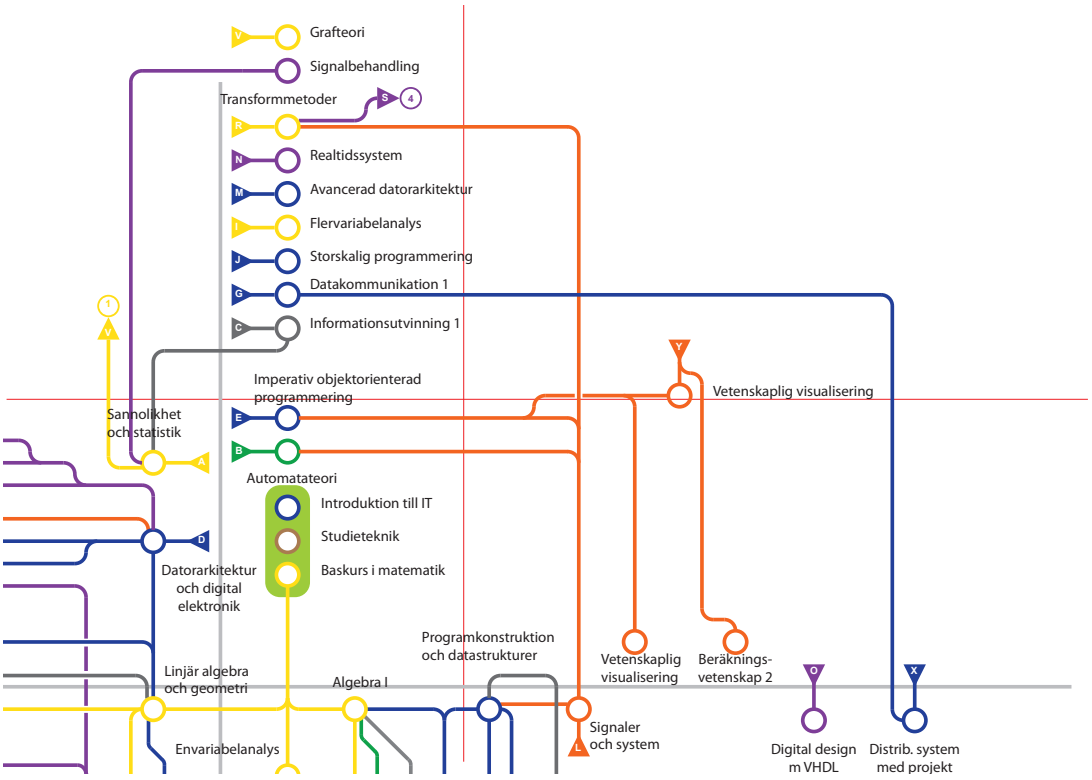


Figure 6: The second subway graph cuts away all connecting lines that extends over two study periods. It provides a clearer graph, but with a lack of continuousness. A larger version is shown in appendix B.

- No courses are actually connected to a certain year, but more importantly connected by their classifications and requirements. What this means is that there is no rule saying that a certain course should be taken in a certain year. This is rather a recommendation presented in the graph. For students that has the prior knowledge for every course, the year based time line is useless. The same also appeals to students that have slipped behind in their studies.
- Using multiple time axises makes a quick overview harder. For students right in the middle of their studies, its very hard to orientate themselves in such a graph.

With this in mind, the new graphs that are created uses a circular time axis that extended over four time periods, just like a clock, or the four seasons. The fact that no courses are connected to certain years makes it easier to place them around a circular time axis. The idea of using the clock metaphor as a replacement for the timeline leads to the next question, whether classes should be used at all.

The graph in figure 5.3.1 is a sliced quarter of the first course graph. This uses the clock metaphor where each course are placed in a circle, in their respective period. This graph is cleaner than the original course graph, by many means.

- It is easier to use a circular course graph when presenting the program, since the graph is scalable. It is possible to just present the current period, and still make it lucid and good looking.
- Since it focuses on periods rather than class it has potential to catch the students with unfinished courses. By showing all courses, students are reminded of these unfinished courses when planning next period.
- It is easier to plan since the graph has constant focus on the next period.

When looking at the program based on the tree graph, one thing are clear. Classes can be removed. Removing the concept of year from a five year science program may seam impossible, but is actually a good idea. As described above, no course are dependent on a certain year, but rather on classifications and requirements. There is no saying on how one should plan their studies as long as, for each taken course, the student fulfills the requirements for that course. The year is basically a way of grouping courses and make sure no incorrect requirements prevents students from further studies. In a way, its a secure way of making sure students have access to courses and are committed to full time studies.

A better way would to only focus on periods. This would make it much easier for students in the middle of their studies to orient themselves. To be able to view all courses for a certain time period could serve as a reminder for students to complete unfinished work. A theory evolved that the use of user recognition could be used to ease the interpretation of the graph.

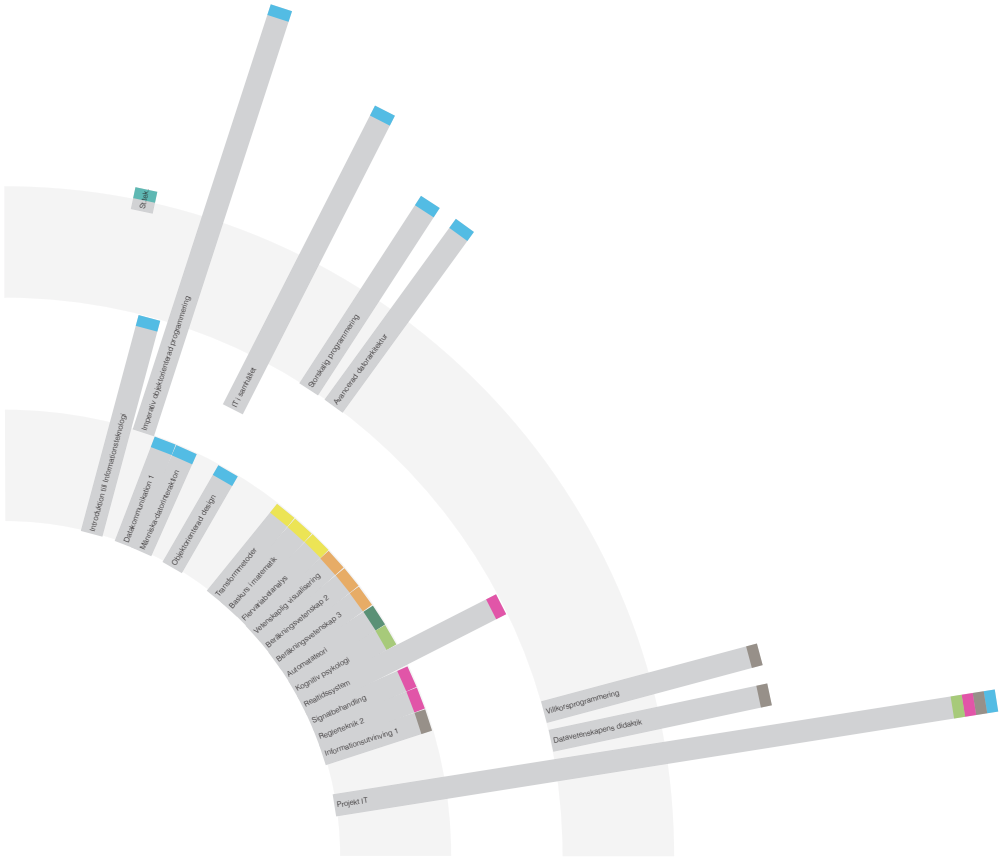


Figure 7: A quarter slice of the circular course graph is used to represent one period of 15 study points, where each ring represent 5 points. Courses that extends over the radius is continued into the next period. The full graph is shown in appendix C.

Another thing that is removed is study pace. The study pace is a percentile number indicating the required engagement from the student. For each period a student should study at least a total of one hundred percent. But this number doesnt fill its purpose. Lets say that two courses are given in a period, where one has a study pace of 33 percent and the other pace is 66 percent. This would suggest that students deposit the double amount of time and work for the second one. But in reality, the needed commitment for a course depends on the difficulty of the course and the capability and talent of the student. The actual pace is individual for each student. With this in mind it is decided that pace isnt useful information and shouldnt certainly steal any focus. With this in mind, it would be more useful to visualize the classifications since this probably would correlate better with how committed the students would have to be.

Int the current graph, dependency arrows is supposed to serve as a support for students in making their course choices. But the large amount of courses just makes up a big pile of directed arrows. Most of the arrows are crossing and mashed up, making it difficult to follow them from one course to another. It could be told right away that using some kind of dynamic graph that only shows dependencies for current courses would be a better alternative. An alternative without arrows is also created. The idea behind this is to group them differently and try to use the time line as requirements indicator. In the current graph, one arrow displays the requirements from one course to another. But the requirement itself prevents students from reading the two courses in the opposite order, so perhaps the use of a timeline could provide enough information on what order to study connected courses.

Using dependency graphs is a quite popular way of visualization, but with that much information the connectors doesnt provide any good information. All they tend to form is good looking information. But looking good isnt equivalent to being good.

The circular course graph in figure 5.3.1 may use dependency arrows to display how courses are connected. The problem is when displaying all arrows at once. These types of dependency graphs seems quite popular, but their functionality may be discussed. Displaying all dependency graphs in a

small circle is not really usable. It could be used as an interactive variant where dependency arrows are displayed on hovering.

From the decision to remove years and pace left the old graph with only one time related variable, periods. The periods are returning for each year, much like seasons of the year. With this in mind comes the idea of building the new graph as a clock, where each quarter represented a period. All courses would be grouped according to that model. Using the model of a clock made students recognize the structure and could easy navigate around the time axis.

From this structure, the students can get a better overview of each term, which is the basis of what courses to pick. Prior to each term, student are obligated to pick the courses the want to take for that time period. Using a clock, or wheel, based structure one could just display the current term by displaying half of the circle. This would be a much more comprehensive image for students to grasp, especially in the domain of a keynote presentation where lack of screen space (or more correctly, screen resolution) demands smaller visualizations.

A new type of graph is also built, with recognition in mind, that is based on the circular course graph in figure 5.3.1. The new graphs uses a classic subway map as a metaphor. The map is based on the circular graph in the sense that periods are aligned clockwise in four square areas. What we want to do is to find a way to connect the courses, in another way than the dependency arrows from the original graph. The idea is to connect them like stations in a subway map. The advantage with the subway metaphor is the clean and structured map it produces. The lines between courses are easy to follow and the different education profiles can be color coded, just like the subway lines. The graph is still based on time periods, forcing the placement of the courses in their corresponding area in the graph. Unfortunately, this produces a bit to long distance between some of the courses, and creates a spiral like pattern, as seen in figure 5.3.1, when trying to maintain the chronological pattern of previous graphs.

The graph in figure 5.3.1 shows the further development of this graph. This graph features a design rule, as an effort to remove the longest lines. Each connection line that reach over two periods is replaced by a link by the shape of an arrow, containing a key letter. This letter could be found in a corresponding arrow, continuing the line towards the course destination. The problem here is that many lines suffers from this long reaching dilemma, causing the graph to be filled with arrows and letters instead. One conclusion of the program could however be drawn from this. Most courses that grants access to other courses are not using this knowledge directly, but rather a couple of periods later. Let's say for example that one course provides skills that grants knowledge to another course. What this should mean is that these skills are to be used in the second course. But in this program, most of these second courses are not planned directly after the initial courses, but rather a couple of periods later. This may be an interesting fact, both for visualization of progression, and when discussing how to motivate students.

5.4 Course graph meta information

The graphs are created to show meta information about the program and is mostly directed towards the program directors and administrators. They tells us information that probably is not that interesting to students. By summarizing and grouping course information, such as orientation and points, some interesting patterns emerges.

The first graph, in figure 5.3.1, shows how courses are distributed among the four periods. Color density are used to visualize the amounts of courses within each time slot. The motivation behind this graph is to support the administrators in their planning of new courses. When planning it is desired to. When planning the courses of each year, you want such a proliferation of courses so that courses of the same score ends up in various points, as much as possible. It would therefore not have any 15 credit courses in the same period. This graph facilitates this distribution.

The other graph, in figure 5.4, is used to visualize the distribution of orientations within the university program. This could be interesting if compared to other master engineering programs at the faculty, or by comparing master engineering programs of the same type on different universities. This could serve as help in orienting the program among the others.

This graph does not really reveal any new or surprising information. The largest circle represent courses of computer orientation, followed by sets of different orientation, such as computer science, human computer interaction, embedded systems. The only eye opener is that there is not really that much math courses in comparison with how the master science courses are generally described. This graph, or variations of it, could be used to present the university program to external parts, such as future students.

5.4.1 Study syllabus feedback

The feedback graphs for each student should serve as support and guidance in their study. What this basically shows is the orientation of the finished courses, summed up by their study points. This

Figure 8: The density graph is used to display the density of different (in terms of study points) course for each period. This is directed to those planning and inserting new courses.

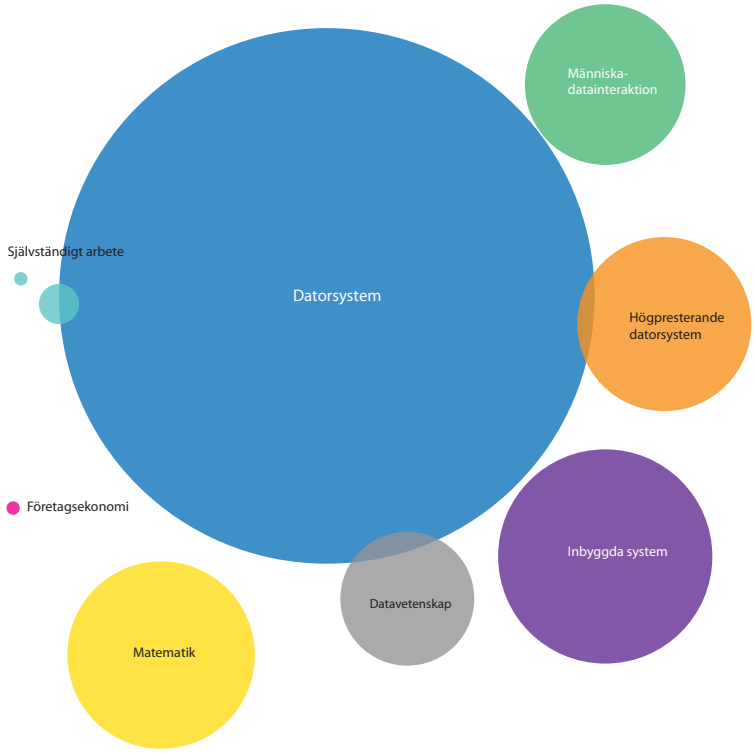


Figure 9: This graph sums up the different orientations within the program, by study points. This could be used to compare different programs with each other.

gives each student a sense of their own profile. One would know what type of course to study in order to become increase their knowledge with a certain area.

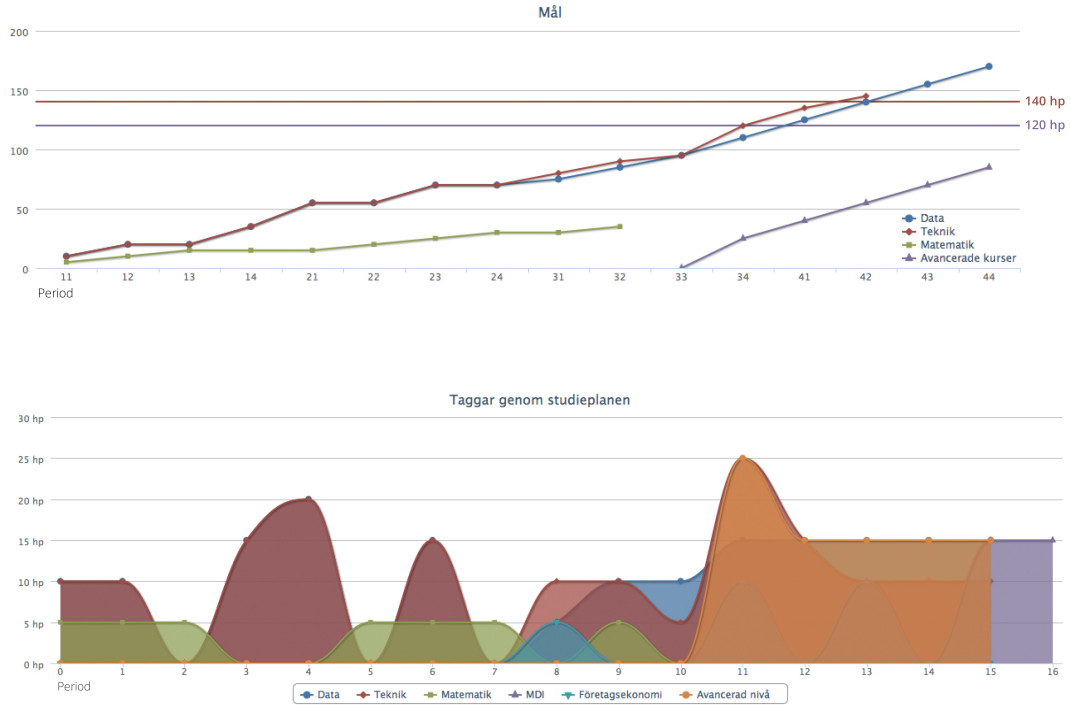


Figure 10: Two feedback graphs from a personal study syllabus. The first one visualizing at which period a certain goal is reached. The graph underneath visualizes the density of each orientation for every time period in the education.

5.4.2 Study syllabus goals and progress

This graph is mainly used to make sure all requirements for graduation is fulfilled. As mentioned, in order to graduate the students must complete enough points of courses labeled as technical and advanced, as well as mandatory ones. The above graph in figure 5.4.1 is basically a set of plots, one for each goal, and a horizontal line displaying at how many points each goal is reached. From this, it is easy to locate in what periods the plots corresponds with its respective goal.

The graph below in figure 5.4.1 represents the density of each profile for each period. This is used to visualize how the different types of courses are distributed along the entire education. The goal would be to have a even mix of different courses throughout the education to maintain motivation among students.

The graph in figure 5.4.1 is all courses from the program grouped in bubbles, much like the ones in figure 5.4. The idea behind this graph is an interactive one where completed courses are bolded. This is a way of promoting the profiles within the program and support the students who wish to take all courses within a certain area. The circle area represents the total amount of points within this area, and the smaller circle on the inside represents the completed amount for the student. The outlined circle within some of the circle represents the required level (if such exists) for each area.

5.4.3 Summary

The graphs is then summarized with regards to some key factors. This summary is intended to give a quick overview of what each of the graphs shows, in terms of traceability, simplicity and more concrete factors like points, periods and course meta information.

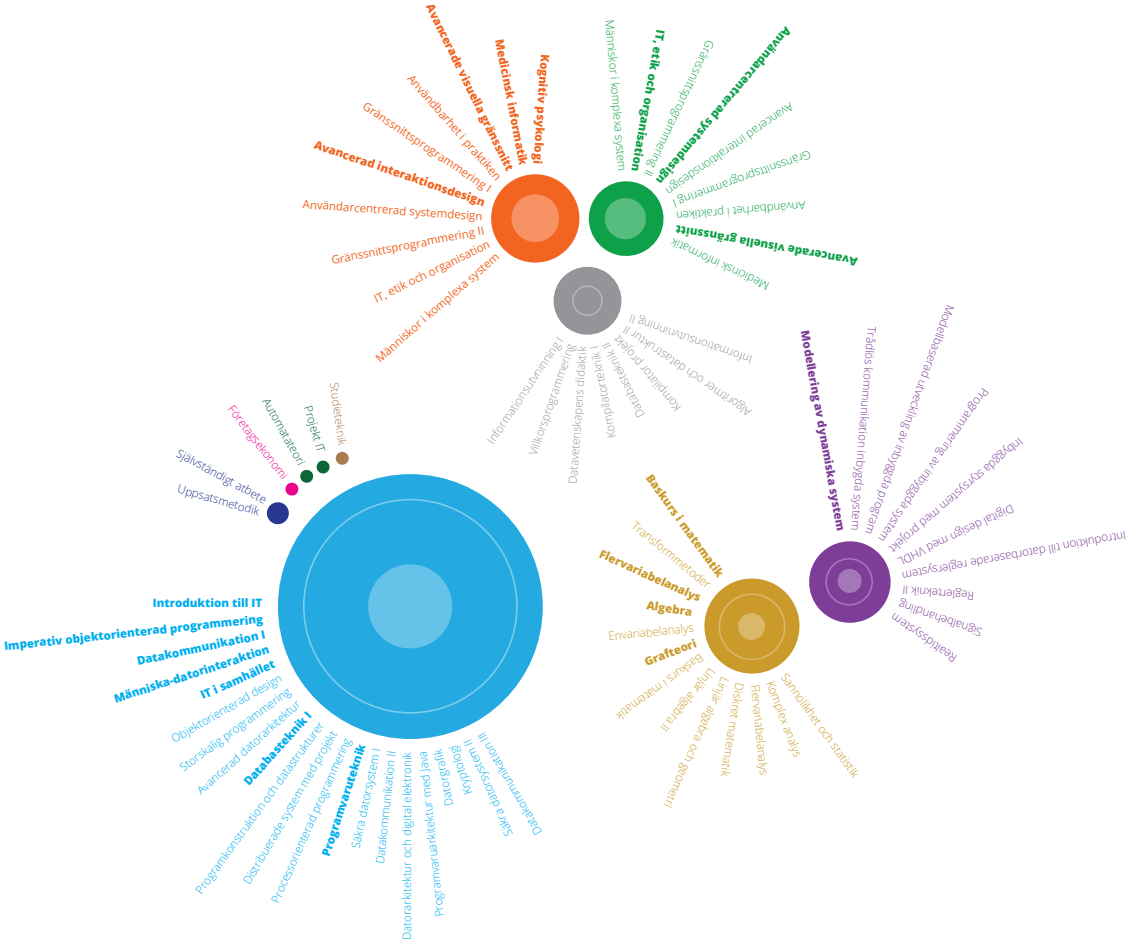


Figure 11: This graph has grouped all courses by their respective orientation. The purpose is to show the completed points within each area in an interactive way. Bolded courses are available to the student. The circle area represents the total points within this profile, and the smaller represents finished courses for the student. The outlined ring represents the amount of points that are mandatory for this orientation. A larger version is shown in appendix D.

Graph	Summary
Original graph	Has a clear structure and traceability. It is easy to see when the courses are held, how they are connected and what their orientation is.
The circular graph (fig. 5.3.1).	It is easier to orientate and get an overview of the available courses. The drawback of this graph is that it does not show how courses are connected. This graph is much cleaner than the original, since no arrows are drawn over the boxes that represents courses. It uses a more natural way of grouping courses in periods and erase the problem with multiple time lines, that the original has.
The density graph (fig. 5.3.1)	Good summary of the program, terms of orientation and points. Cheap on information. Could be useful when comparing programs. This is a complementary graph, but it provides information that is not visible in the original graph. There is no way of grasping what period contains the most courses or points from that graph.
Subway graphs	Simple structure and easy to follow. Gets messy for larger programs. Still, this deals with the overuse of arrow in the original graph. In the original, all crossing arrows makes it quite hard to go from one course to the next. This is handled with this graph.
Bubbles graph (fig. 5.4.1)	Clear overview of courses. Logic grouping and clean feedback. Lacks connectivity and relations between courses. It provides a better overview. It can be used during the whole education, which was a problem with the original. It's impossible to get a quick overview of completed courses from that graph.
Feedback graphs (fig. 5.4.1)	Clear overview for each student. It shows mostly meta information, but none of this information can be drawn from the original graph.

6 Discussion

The first and absolute clearest reflection of this project is that it was slightly too large. Either was a clearer focus, or more time, required to fulfill the original goals. The whole project idea was based on a desire to build a program where you would be able to construct a study syllabus graph. The graph in question had already been made in another program which, among other things, left the flexibility and modularity much to be desired. However, the focus shifted quite quickly towards improving the existing graph. From this focus shifted more and we wondered what other graphs that were interesting. What else can students and course administrators might want to know?

The application itself worked to create the study syllabus, both for individual students and entire program. Nonetheless, the version that should be considered as a proof of concept.

From a personal point of view, this project has been invaluable. First of, the freedom of choosing platform was a great way of develop personal skills within web programming. Second, the numerous of tools and visualizations that has been discovered throughout the project is now an arsenal of both inspiration and available options for future work.

This chapter will go through the methods, graphs and the system, and list some reflection as well as possibilities for future work.

6.1 Methods discussion

The methods used in this project was merely used for personal development. They could, and probably should, be done more consistently to produce better results. But with the short time span that a master thesis has to offer, this has been one of the cutbacks.

6.2 System improvements

Making a list of system improvements is just slightly harder than inventing a time machine, but those listed here are improvements that could be possible in the near future.

6.2.1 Export possibilities

Being able to export the visualized data is a feature that was not discussed within this project. However it is something to keep in mind for future development. Being able to export graphs in vector format could make them usable in external presentations using Microsoft Powerpoint, Adobe Creative Suite, etc.

One preferable format would be to export information to the DOT language, which is a description language for graphs. The DOT language supports function-like statements to create different types of graphs, attributes, relations etc. The .dot files can be edited in plain text editors. Also, the CSV format, when dealing with large sets of data.

6.2.2 Multiple views

One of the goals of this application was to be able to visualize the courses, there correspondence and connectivity. This should be made from planning and testing different study syllabuses. The natural users of such a feature should be the students. Being able to see the effects and possibilities from their picks, and from that ease their way to an exam is a powerful tool that could be implemented on some domain such as the Student Portal.

6.2.3 Selma Connectivity

A big part of making this application useful for students is by providing a connection to the actual course database, Selma DB. Accurate course information is a must if a live version of this application should see the light of day.

6.2.4 Better user feedback

The planning sections could feature better and more diverse feedback. For example, displaying amount of points selected or displaying whatever some mandatory courses has been left out.

6.3 Graph improvements

The graph could, of course, be endlessly improved, but the goal for this thesis is that these graphs has planted some new ideas in the head of the reader.

New graphs could for example be produced to visualize the courses in relation to their points in the course evaluation. This could be useful when doing inventory, and perhaps, decide on new courses within the program.

Graphs that visualized skills that are not formally assessed in courses, are also of great interest. This could be used to show the skills that are not directly connected to a study program. For example, finding a way of visualizing courses in which report writing, or perhaps code writing, are used and developed.

7 Conclusions

The project seemed quite successful. In particular, the graphs produced in response to the original course graph. This is something that hopefully can be further developed where the next step is a fully operational system oriented in a larger user group.

It has been useful to twist and turn on the different parts, both the graph and application, to know what should be created in a first sharp version. The program coordinator had a very clear picture, which was probably similar to his original graph, but the project has questioned this and effectively come up with several possible options, many of which would work better.

The project did not reach the original ambitions and wish of the program director, that it would result in a finished product. But I believe that we've come a far way and answered some questions that may have arose during the implementation. Mainly, if the original tree structure graph was the best way of describing and visualizing the program. This has not yet been decided, but chances are big that some of the graphs created within this thesis, or alternations, will be used.

The web based prototype was quite useful in demonstrating what could be done with a web interface. There were no platform requirements at the beginning of the project, but this now looks like a settled issue. The use of web interface enables incorporation with other University services, such as the Student Portal.

Some of the possible objections that some might have towards this thesis should also be addressed. There are many issues that needs to be solved before this thesis can be concretized into some kind of product. For example, this report doesn't deal with any of the security issues that comes with the use of a web based application.

The project should also feature a better synchronization with the course database, Selma. It has not been proposed how to deal with the information when testing new setup of courses and playing around with ideas. From the perspective of this application, Selma is a read only database. No information should be altered from this tool. But some information might just have to be, in order to try out new ideas. This multiple database structure and synchronization is yet to be solved.

The biggest success of this master thesis was analyzing, questioning and present improvements to the original graph. The tool is mainly used to present courses to others although it has never been a subject of analysis or evaluation among its audience.

The problem has been concretized. The next step is to do the same with the solution.

8 Acknowledgements

Mattias Larsson, for his substantial work prior to this project. Bjrn Victor, whom with little constraints allowed me to take this project in a direction where I developed my skills in web programming. Mikael Laksoharju for guidance, feedback and patience. Åsa Cajander, for all great feedback. My fellow colleagues Erik Hedberg and Anders Lisspers for their priceless programming knowledge and input when building the application.

Written in LaTeX

References

[1] Alain Abran and James W. Moore. *Guide to the Software Engineering Body of Knowledge*. Unknown, 2004.

[2] Douglas Brown. Zend framework vs CakePHP framework, December 2008.

[3] Michael Christel and Kyo C. Kang. Issues in requirements elicitation, 1992.

[4] Mary Jo Davidsson, Laura Dove, and Julie Weltz. Mental models and usability, November 1999.

[5] James Hobart. Principals of good GUI design, October 1995.

[6] Caroline Jarret and Gerry Gaffney. *Forms that Work Designing Web Forms for Usability*. Morgan Kaufmann, 2008.

[7] Sean Michael Kerner. Google data joins PHP zend framework. *InternetNews.com*, December 2006.

[8] Martin LaMonica. IBM backs open-source web software. *CNET News*, February 2005.

[9] Mattias Larsson. Systembeskrivning kursverktyg, 2011.

[10] Riccardo Mazza. *Introduction to Information Visualization*. Springer-Verlag, 2009.

[11] J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla, and Anandha Murukan. Improving web application security: Threats and countermeasures, June 2003.

[12] Christer Nilsson. Selma DB tabellbeskrivning, 2010.

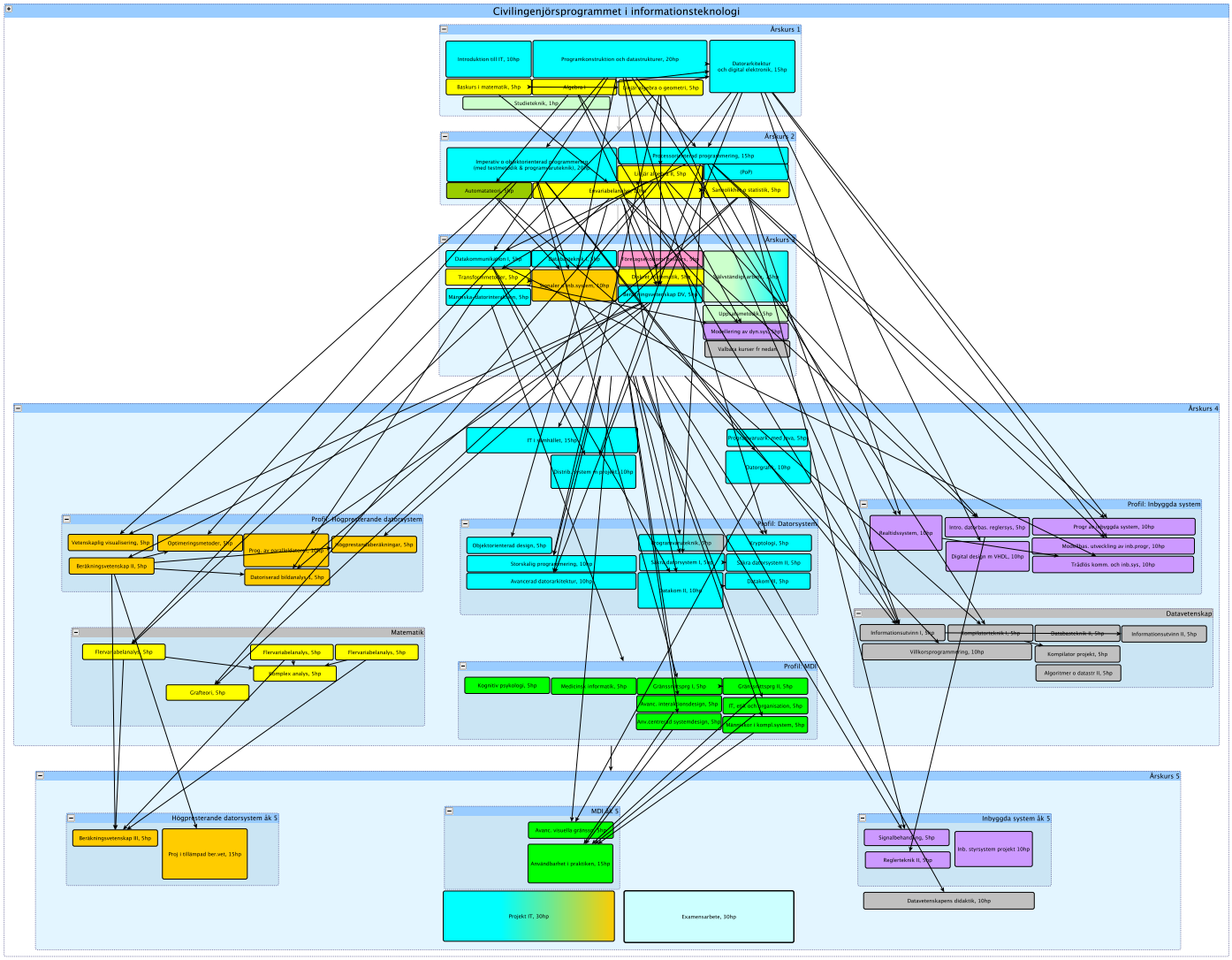
[13] Robert Spence. *Information Visualization*. Addison-Wesley, 2001.

[14] Edward Tufte. *Beautiful Evidence*. Graphics Pr, 2006.

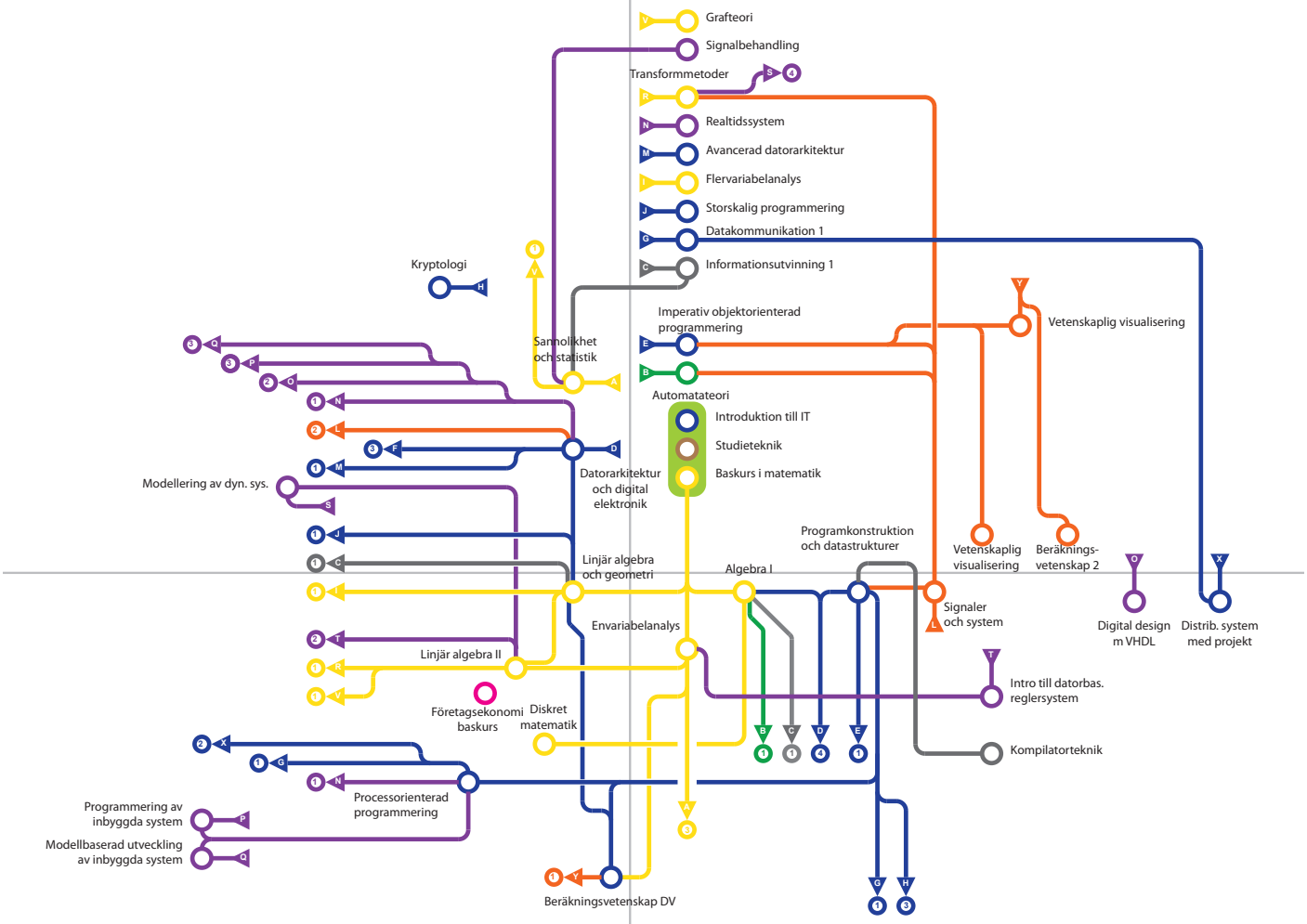
[15] Sven Trnkvist. Smart buisness, May 2011.

Appendix A

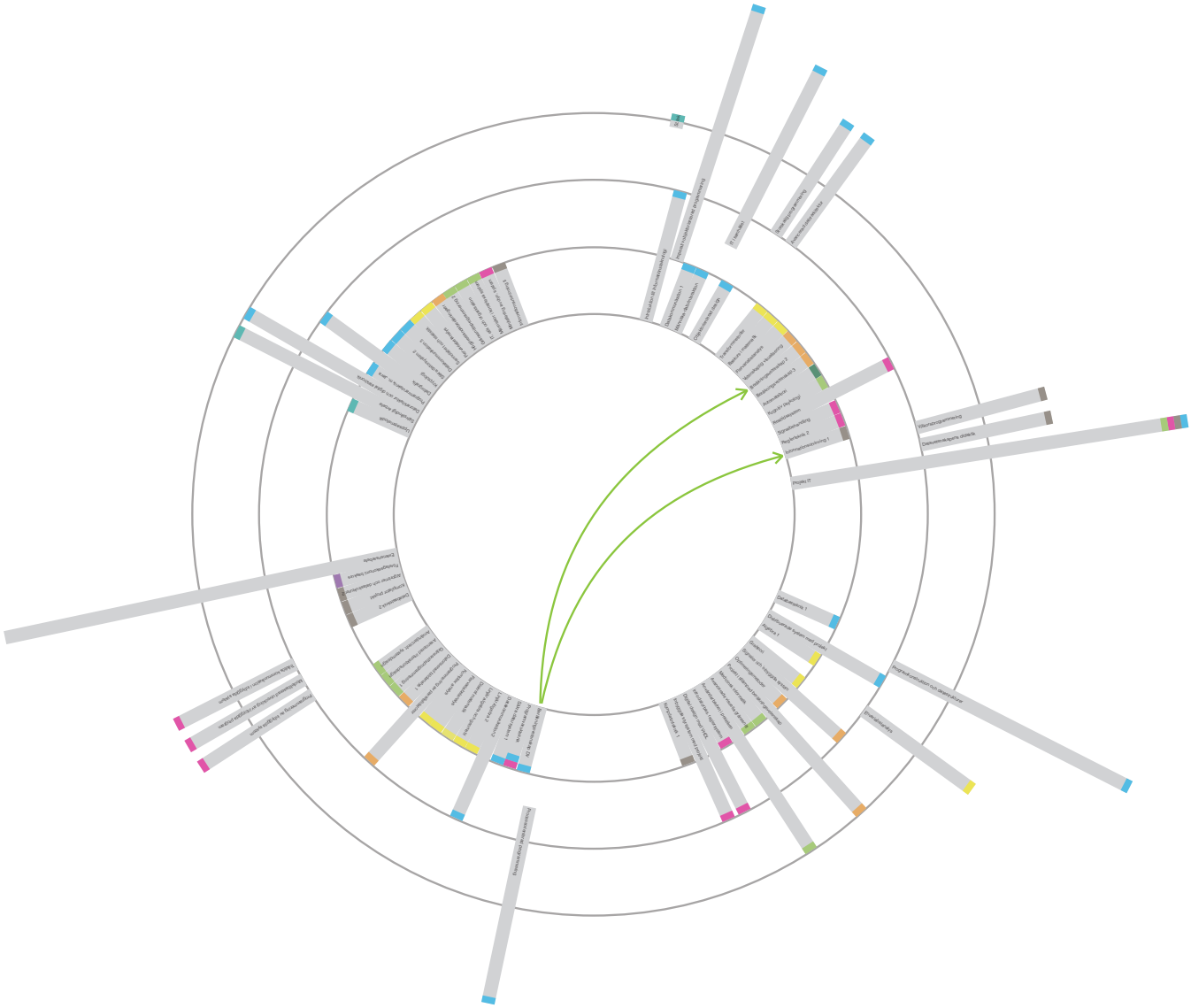
The original course graph, created in yEd by Björn Victor.



Appendix B
The subway graph



Appendix C
The circle graph



Appendix D

The study syllabus graph