“Information Infrastructure-Prototyping” as a means of elucidating and elicit requirements:
A Case Study at Uppsala University

Jakob Malmstedt

June 21, 2014

Keywords: Prototyping, Infrastructures, Information infrastructures, II, Planning system, IT support, Administration

Supervisor:
Owen Eriksson
Abstract

Most literature in Information Systems are good at pointing out the advantages of having a well thought-out information infrastructure in place in order to support the activities of the organization. They are also good at mentioning the problems that these information infrastructures often have in reality. However, what the literature does not do too well is to provide guidance as to how the integration of a new system into an already existing information infrastructure can be planned and implemented.

As Hanseth and Lyytinen points out “a standard text book on object oriented methodologies (Mathiassen et al., 2000) spends exactly 1% (4.5/450) of it’s pages on systems integration and issues how to relate design to-be with the current environment” (Hanseth and Lyytinen, 2004, footnote, p. 211).

This master thesis explores how a new information system can be integrated in the as-is information infrastructure by analyzing the information infrastructure at Uppsala University, where they have identified a need to address the existing problems. This is an information infrastructure that does not fully support the activities of the organization which leaves them with a lot of unnecessary manual work that could instead have been automated. A possible solution to these issues has been found to be the integration of a new system to fill in the gap where the infrastructure does not support the organization. This is a complex operation and a complex issue which is also difficult to fully understand. Therefore this master thesis explores and proposes a method where prototyping is used for multiple purposes to elucidate and elicit both business and technical requirements concerning the integration of a system into an existing information infrastructure.

50 user requirements, 5 interaction requirements and 5 information infrastructure requirements were generated through the use of the presented approach. The results suggests that this approach indeed can be advantageous in situations where requirements concerning the new system as well as the interoperability with other systems in the information infrastructure are not apparent and thus difficult to define.
Acknowledgements

I would like to thank my supervisor Owen Eriksson for the opportunity to write my thesis about this interesting case and for the help, support and dedication.

I would also like to thank everyone else who participated and kindly offered their time in favor of this study. It is greatly appreciated.

Thank you!
Contents

1 Introduction 1

2 Literature review 3
  2.1 Information Infrastructures 3
     2.1.1 Background and history of II 4
     2.1.2 Definition of II 5
     2.1.3 Classifications of II 5
  2.2 Common issues in Corporate Information Infrastructures 6
  2.3 Information infrastructure strategies 7
     2.3.1 IRM 7
     2.3.2 VBS 8
     2.3.3 PAKS 9
  2.4 Design principles for IIs 9
  2.5 Service Oriented Architecture 11
  2.6 The problem with eliciting requirements 12
  2.7 Prototyping 14
     2.7.1 Definition of Prototyping 14
     2.7.2 Motivation for the use of prototyping 15
     2.7.3 Prototyping vs Agile methodologies 15
  2.8 The gap in the literature 16

3 Methodology 18
  3.1 Case study 18
  3.2 Design and Creation 19
  3.3 Data generation and analysis 20
     3.3.1 Document analysis 20
     3.3.2 Interviews 20
3.3.3 Demonstrations .............................................. 21
3.3.4 Analysis of transcriptions .................................. 21
3.4 Implementation/Approach ..................................... 21

4 The Context at Uppsala University .......................... 24
4.1 The Organization ............................................. 24
4.2 The Information Infrastructure ............................... 25
  4.2.1 Corporate level information infrastructure .............. 26
  4.2.2 Business level information infrastructure .............. 26
4.3 The Activity System ......................................... 28
  4.3.1 Establishing of new education .......................... 29
  4.3.2 Student recruitment ..................................... 30
  4.3.3 Budgeting on a subject level ........................... 32
  4.3.4 Budgeting on a course instance level ................... 33
  4.3.5 Teacher planning ....................................... 34
  4.3.6 Room booking ........................................... 35
  4.3.7 Course implementation and follow-up .................... 36
  4.3.8 Reporting of worked hours ............................. 37
4.4 Important concepts and definitions ....................... 37
4.5 Context summary ............................................ 38

5 Problem Description ........................................... 39
5.1 Problems in the Information Infrastructure ................. 39
  5.1.1 The isolated island TimeEdit .......................... 39
  5.1.2 A misguided conceptual apparatus ...................... 40
  5.1.3 Lacking documentation ................................ 40
  5.1.4 Lacking API in Selma .................................. 41
  5.1.5 Missing IT systems ...................................... 41
5.1.6 Non user-centered development ............................................. 41
5.1.7 Incapability of affecting systems ................................. 41
5.2 Problems in the Activity system ............................................... 42
  5.2.1 Problems for the Directors of Studies .......................... 42
  5.2.2 Problems for Main Teachers ........................................... 42
  5.2.3 Problems for Teachers .................................................. 43

6 Results .............................................................................. 44
  6.1 The prototype ................................................................. 44
  6.2 User requirements .......................................................... 60
    6.2.1 Course instances ....................................................... 60
    6.2.2 User roles ............................................................... 60
    6.2.3 Budgeting ................................................................. 60
    6.2.4 Teacher planning ...................................................... 61
    6.2.5 Course instance parts ............................................... 61
    6.2.6 Reporting of hours ................................................... 61
    6.2.7 Activity planning ...................................................... 62
    6.2.8 Groups ................................................................. 62
    6.2.9 Room booking / Scheduling ...................................... 62
    6.2.10 Other .............................................................. 63
    6.2.11 General feedback ................................................... 63

  6.3 Interaction requirements .................................................. 64
    6.3.1 The integration with Selma ........................................... 64
    6.3.2 The integration with Ladok ......................................... 64
    6.3.3 The integration with Studentportalen ......................... 64
    6.3.4 The integration with budget system ......................... 65
    6.3.5 The integration with TimeEdit ................................. 65
6.4 Information infrastructure Requirements ........................................ 66
   6.4.1 Lacking system documentation ........................................... 67
   6.4.2 Lacking and misguided conceptual apparatus .......................... 67
   6.4.3 Isolated islands of systems .............................................. 67
   6.4.4 Inaccessible testing environments ...................................... 67
   6.4.5 Missing system ............................................................ 67
6.5 Consequences of implementation ................................................. 68
   6.5.1 Activity system ............................................................ 68
   6.5.2 Conceptual apparatus ..................................................... 68
   6.5.3 Information infrastructure .............................................. 68

7 Discussion .................................................................................. 70

8 Conclusions ................................................................................. 71
   8.1 Limitations ................................................................. 72
   8.2 Further research ............................................................ 73

A Activity Planning Prototype ERD (without user tables) .................... 76

B TimeEdit Actual Booking Request XML Structure .......................... 77
1 Introduction

It has been known for many years now that the Information infrastructure (II) within Uppsala University is lacking (Fiske and Eklund, 2013, p. 1) and is actively constituting negative effects on the organization (Fiske and Eklund, 2013, p. 51).

The problem is that the collection of Information Systems (ISs) in the II is not sufficient to properly support the business activities in the organization. In practice, this means that employees have to carry out manual work such as sending and receiving e-mails back and forth with reports of worked hours and double work such as entering in previous years budget once again in order to adjust it to fit this years circumstances. This effectively results in reduced efficiency for the organization and reduced working conditions for it’s employees.

Arguably the most serious of the effects is concerning the working conditions for individuals dealing with administrative work. The individuals who gets the hardest hit are the Directors of Studies but it also affects Heads of Department, Main teachers and everyone else involved in administrative work. Important to note here is that the administrative duties have, by means of the II, been decentralized outwards in the departments which has caused an increased work load for virtually all employees at Uppsala University (Fiske and Eklund, 2013, p. 51).

This situation is particularly complex and difficult to resolve due to the diversity of working practices and routines among the relatively high number of departments that exists in this type of University setting. This results in ambiguous requirements of the different supporting systems which may result in confusion and uncertainties about how the systems should be built, from the developers point of view.

There is also of course, a very political aspect to this problem because different stakeholders have different needs and agendas that they want to satisfy. Who gets to- and who should- decide on the design of the information infrastructure?

This issue is very important to resolve because it is far from a unique problem. In fact, the same issues are present in other Universities in Sweden (Eriksson and Ågerfalk, 2010).

Part of the reason behind this is that there is a shared Information Infrastructure, on a national “Business level” that all Universities has to connect their local “Corporate level” of information infrastructures to (Fiske and Eklund, 2013, pp. 31-32). This of course, adds to the complexity of the big collective information infrastructure, forcing shared standards upon the systems involved. Therefore, there is a lot in common between the information infrastructures used by the Universities in Sweden and many of the issues experienced by Uppsala University will also apply for other Universities.

This thesis is building upon a previous study (Fiske and Eklund, 2013) that focused on identifying and evaluating the II at Uppsala University by carrying out a work practice diagnosis. One of the main suggestions resulting from that study was that an Activity Planning System should be built to facilitate the Directors of Study and Teachers in their administrative chores (Fiske and Eklund, 2013, p. 63).

This thesis is focusing on that particular missing Activity Planning System and it’s relation to the existing II. If this system could be successfully developed and integrated it would significantly increase the working conditions of the affected employees by decreasing and simplifying the
administrative work while also streamlining the activity system in the organization by eliminating double work, and manual work.

It is vital that this system is developed correctly from the beginning so that the issues gets fully resolved and that the system ideally suites the needs of everyone involved.

But in order to achieve this there are also several underlying problems that needs to be addressed first. These include a lack of interoperability in the information infrastructure, isolated islands of systems and a misguided and diffuse conceptual apparatus - all of which are common problems existing in most information infrastructures (Axelsson and Goldkuhl, 1998, pp. 10-11).

The “as-is infrastructure” or “installed base” is important to get right because it has a significant influence on how a new part of the information infrastructure can be designed and how it, and the information infrastructure, can evolve (Hanseth and Lyytinen, 2004, p. 214).

The task of developing and integrating this system is therefore much more complex than it might first seem, and there is also a more general problem, existing outside of this particular situation, which is important to highlight. As we will see, there are currently no available strategies designed to help gather the necessary requirements needed for a successful planning and integration of a new system into an existing information infrastructure (see section 2.8). This of course makes this process more difficult, especially in situations where developers have failed to properly take into account the interoperability between their information system and systems surrounding it.

The purpose of this thesis is therefore explore the use of prototyping as a means of elucidating and elicit requirements of both the new system and the information infrastructure it needs to relate to in order to create a successful integration and interoperability between the system and it’s environment.

The standard way of illustrating how different systems work and interoperate is by using the models, descriptions and APIs that were made during the development of the different systems. These documents are in most cases inadequate means of building a proper understanding for most system users.

The idea behind this thesis is that a prototype is needed in this scenario in order to illustrate the problem from a user- and a design perspective. This prototype is therefore meant to be relatively functional so that users can click around and get a sense of how this system might work if used in their everyday work, and how the system could interact with other ISs in the II. In that way they can easier relate to and quickly get an understanding of the advantages or disadvantages of such a system and therefore be able to give valuable feedback as to what is good and bad about a particular solution.

This will be further explained in the coming sections. The next section will look at the previous work that has been done in this domain in the Literature review (section 2). This will be followed by the Method used in this thesis (section 3) and then a more detailed description of the situation at Uppsala University will be presented (section 4). A detailed problem description will then be presented (section 5) followed by the results of the study (section 6). The thesis ends in a discussion (section 7), and the conclusions, including limitations and suggestions for further research(section 8).
2 Literature review

This section is looking closer at what has previously been written about these issues and topics in order to build a basic understanding of these and how they have been treated earlier.

The section begins with a brief introduction to the concept of Information infrastructure. This is then followed by a historical perspective of II describing it’s emergence from the late 60’s and onward. This leads us to the formal definition of the concept and the ways in which they can be classified. We then take a look at common issues in Corporate IIs and how these issues might be avoided through the use of strategic planning and the pursuit of various design principles and techniques. After that, the difficulties in describing complex Information systems and infrastructures is examined and what role prototypes might have in the illustration of these. The section is concluded with a summary and reflection of the literature review.

2.1 Information Infrastructures

Information infrastructure is a slightly diffuse concept because there are many different names for the same concept and also many similar types of infrastructures that have slightly different names and meanings.

This has for example been illuminated, in 1980, by E.A. Ben-Nathan (cited by Hansson and Nygren, 2003, p. 10) who wrote:

"Information systems architecture is one of these simple-sounding words with a very fuzzy meaning".


It is still slightly fuzzy what exact meaning the different authors put into these four terms. But the understanding in this thesis is that these four terms “Information system architecture”, “IS-Infrastructure”, “ICT system infrastructure” and “Information Infrastructure” refers to essentially the same concept.

Goldkuhl and Axelsson (1998) also illuminates and emphasizes the importance of differentiating between an “IT-infrastructure” and what they call “Information system architecture”. They point out that an “IT Infrastructure” refers to the fundamental technical infrastructure consisting of hardware, peripherals and basic software such as operating systems as opposed to the “Information system architecture” which instead of the technical aspects refers to the activity and information aspects.

In this thesis we are using the term Information Infrastructure, meaning, the set of information systems used by a user base and the interfaces and relationships between them. The term also acknowledge the inseparable nature of information infrastructures and what is referred to as activity infrastructures which consists of an organizations business activities.
Before we go on to looking at the formal definition of Information Infrastructure, we will first look at the concept in a historic perspective to get a deeper understanding of the phenomena.

2.1.1 Background and history of II

In the late 60’s and beginning of 70’s there was an ideal among many information systems researchers about a “total system” for the organization or what they called a MIS (Management Information System) (Andersen et al., 1994, p. 500). This was meant to be one big integrated system for the whole organization.

This ideal soon broke down due to the realization that a single system did not have enough flexibility to comprise functionality for all functions of the organization (Andersen et al., 1994, p. 500).

As a reaction to this, organizations instead started building separate systems for different organizational activities but kept the idea of central control of the systems. During the late 70’s and early 80’s the development went from having central computers to having departmental computers to having personal computers.

The system development then shifted towards smaller and simpler systems, which resulted in a boom of information systems. This led to the realization of a need to integrate these systems and make them interoperate with each other (Andersen et al., 1994, p. 500). But at this point there was no real holistic view of the compositions of Information systems. The important thing was that they “worked together”.

In the middle of the 90’s most organizations used many different information systems, but the integration and interoperability between these systems were still troublesome to get working. The acquisition of new systems often entailed a lot of unforeseen consequences that affected the rest of the systems. There was at this point a growing need for new philosophies and strategies for these types of Information Infrastructures. (Andersen et al., 1994, pp. 500-501, p. 516).

Since then, the research in this area have grown and new strategies and philosophies have been developed to make different and heterogeneous systems interoperate. At the same time major advancements in the performance of computer hardware as well as new software techniques, philosophies and languages such as SOA (Service Oriented Architecture), SOAP (SOA-Protocol) and XML (eXtensible Markup Language) has made possible the development of complex compositions of integrated information systems.

Although still in the relatively early stages of Information infrastructure theory, the popularity and importance among organizations of having truly integrated information infrastructures has increased drastically (Byrd et al., 2006, p. 101), (Axelsson and Goldkuhl, 1998), (Eriksson and Ågerfalk, 2010, p. 434). Byrd Lewis and Bradley for example notes that “A truly integrated IS infrastructure is one of the top concerns of firms” (Byrd et al., 2006, p. 101).

This transition entails new ways of planning and designing our information systems and information infrastructures (Axelsson and Goldkuhl, 1998, p. 14) (Hanseth and Lyttinen, 2004, p. 208). Shared standards naturally becomes more important and decisions has to be made at a higher level, crossing IS boundaries.
Next we look at a more formal definition of the concept of an Information Infrastructure.

2.1.2 Definition of II

There are several definitions of Information Infrastructures with very different scopes and detail. But in order to avoid confusion, this thesis will stick with Hanseth and Lyytinen’s definition as follows:

An Information Infrastructure is “a shared, evolving, heterogeneous installed base of IT capabilities among a set of user communities based on open and/or standardized interfaces” (Hanseth and Lyytinen, 2004, p. 208)

This definition highlights a number of features that every Information infrastructure is said to have in common. These are that they are shared by a user group, they are evolving which means that they are under continuous change, they are heterogeneous because they are made up of many smaller subsystems, and they have an installed base which each increment has to relate to. This definition also emphasizes the important role of shared standards to information infrastructures.

2.1.3 Classifications of II

Information infrastructure is also a very wide concept because “information infrastructures can vary enormously in terms of their scale and functionality” (Hanseth and Lyytinen, 2004, p. 216). Therefore, it is also helpful to be able to classify them into different subtypes. Hanseth and Lyytinen (2004) has identified two different approaches to do this. One based on the “horizontal scope” of II and one based on the “vertical scope”.

The Horizontal scope of II consists of three different classifications. In this approach the strategy is to decompose an information infrastructure into simpler ones by distinguishing between business functionality and infrastructure services. The three horizontal classifications are:

- Application infrastructure
- Transport infrastructure
- Service infrastructure

The two latter of these can also together be referred to as support infrastructures.

An Application infrastructure is and II consisting of the application functionality and is constituting the value making functions and is the reason for building the information infrastructure in the first place (Hanseth and Lyytinen, 2004, p. 217).

The Transport infrastructure is an II consisting of data transportation techniques such as TCP/IP and SOAP (Hanseth and Lyytinen, 2004, p. 217).

The Service infrastructure does not deal with data transportation but rather other helpful utilities such the DNS (Domain Name Service) that resolves textual web addresses on the internet into their respective physical address (IP address) (Hanseth and Lyytinen, 2004, pp. 217-218).
The **vertical classification** also consists of three types of information infrastructures and are distinguished from each other mainly by their scale. These are:

- Universal Service Infrastructure (Internet)
- Business sector Infrastructure
- Corporate Infrastructure

The **Universal Service Infrastructure** is an Information Infrastructure on a global level, which means that it can be used virtually by anyone and any application on earth. This type applies for example to the Internet (Hanseth and Lyytinen, 2004, pp. 216-217).

The **Business sector Infrastructures** are Information infrastructures that are shared between different organizations in the same business sector (Hanseth and Lyytinen, 2004, p. 217). This type includes for example B2B (Business to Business) applications where different businesses need to exchange services from each other such as wholesaler and manufacturer.

The third type is called **Corporate Infrastructure** and refers to Information infrastructures that are owned by a corporate but can also be shared with various partners outside of this corporate (Hanseth and Lyytinen, 2004, p. 217).

2.2 **Common issues in Corporate Information Infrastructures**

The drastically increased use of Information Infrastructures entails a shift in how the development of these systems has to be organized/structured and planned. A decade ago, the usual way that an Information Infrastructure emerged was that each Information System was developed one at a time without much concern for how these systems might communicate with each other later on (Axelsson and Goldkuhl, 1998, p. 9). Goldkuhl and Axelsson (2008) described this approach as Information infrastructures evolving like a patchwork (Axelsson and Goldkuhl, 1998, p. 10) and Beynon-Davies (2013) calls this “Piecemeal ICT systems” (Beynon-Davies et al., 2013, p. 188). This approach means that the information infrastructure is developed with no holistic plan in mind and therefore there is no plan for the interaction between the ISs that constitute it (Axelsson and Goldkuhl, 1998, p. 10).

Such an approach can make the infrastructure incomprehensible and resistant to change (Axelsson and Goldkuhl, 1998, p. 10). The effect of this is lacking **interoperability** between the systems. In practice this might mean that there are isolated islands of systems that do not directly communicate with any other system.

The existence of such isolated islands of systems can have a lot of negative effects. One of them is referred to as duplication of data or what Goldkuhl and Axelsson also calls redundancy. This means that the same information needs to be stored in multiple places due to the lack of interoperability between the systems (Beynon-Davies et al., 2013, p. 188). This in turn might lead to unnecessary maintenance, unnecessary manual work and inconsistencies in the information that is stored (Beynon-Davies et al., 2013, p. 188). This can also lead to different
systems performing the same processing, this is what Beynon-Davies et. al refers to as redundancy (Beynon-Davies et al., 2013, p. 198) (Axelsson and Goldkuhl, 1998, p. 11).

This “piecemeal” or “patchwork” approach also likely leads to inconsistencies in the conceptual apparatus because there is no clear consensus about what concepts and terms that should be used for different information objects. An ambiguous conceptual apparatus leads to difficulties in terms of comprehensibility and interaction between the systems and also poor information quality (Axelsson and Goldkuhl, 1998, pp. 10-11).

Another common problem is what Hanseth and Lyytinen refers to as “Technological lock-ins”. In practice this means that because of technical design decisions, made in the passed, the II can get resistant- or difficult to change later on. This can be related back to what was previously mentioned (in section 2.1.1), that in the 90’s the acquisition of new systems often entailed a lot of unforeseen consequences that affected the rest of the systems. The reason for this inflexibility is because of dependencies between systems, we say that systems are “tightly coupled”.

This unplanned emergence of information infrastructures described for over one and a half decades ago are likely the same reason behind the problems that exists in today’s information infrastructures. Beynon-Davies points out that; although this is not how we design ISS today, there are still systems that was designed a time ago that are still in use and suffer from these issues (Beynon-Davies et al., 2013, p. 188). Important to keep in mind here though, is that “how we should design systems” is not the same as “how we actually do design systems” and there are of course cases where the design is still carried out this way.

In the next subsection we look at what strategies and principles have been proposed in order to avoid these problems.

### 2.3 Information infrastructure strategies

In their book “Strukturering av informationssystem” meaning “Structuring of Information Systems”, Göran Goldkuhl and Karin Axelsson (1998) are examining two different strategic methods for planned structuring of Information Infrastructures (or with their terminology “Information System Architectures”), namely; IRM (Information Resource Management) and VBS (Activity Based System Structuring). They also propose their own strategy PAKS (Process, Activity and Component -based System Structuring). These strategies are described next.

#### 2.3.1 IRM

In IRM, information is seen as central to the IS and the information infrastructure is seen as a shared resource in the organization (Axelsson and Goldkuhl, 1998, p. 15).

Information is seen as a resource in the same way as other organizational resources such as employees. Therefore it needs to be planned and managed in a similar way (Axelsson and Goldkuhl, 1998, p. 35).

In this view the structure and integrity of information and data is emphasized. The IS and the database is seen as a model or a mirror image of reality. The challenge is to find stable objects.
That is — objects in the organization that can be represented in a computer safely without changing significantly over time (Axelsson and Goldkuhl, 1998, p. 35).

The goal is to represent the organization without letting the users’ information needs be a determining factor for how the structure of that database is designed. This is because the information need is said to change over time while the data structure stays the same (Axelsson and Goldkuhl, 1998, p. 36).

It is also important to unravel all the important concepts, synonyms and homonyms used in the organization to minimize the number of objects and thereby simplify it’s computerized counterpart (Axelsson and Goldkuhl, 1998, p. 35).

It is believed that if you build your database and structure and manage your data and information in a way that maintains integrity of the data, you will save a lot of time and money because of fewer errors and mistakes will be introduced in the lifetime of the IS. It is also believed that managing the data in this way will result in more flexibility to change because the structure of data is stable because it reflects reality. It is also said to increase the quality of the ISs (Axelsson and Goldkuhl, 1998, p. 43).

2.3.2 VBS

VBS is a strategy in which ISs are seen as decentralised resources. This means that certain parts of the organization have responsibility for certain parts of the Information infrastructure. For example, the finance department would have the responsibility for ISs in the information infrastructure dealing with financial functions that support their activities (Axelsson and Goldkuhl, 1998, p. 45).

In a similar way each IS should have the responsibility of a specific organizational function. So in this view each information system have the responsibility to provide functions and information to a certain part of the organization (Axelsson and Goldkuhl, 1998, p. 45).

The understanding in this view is that shared resources creates dependency relationships and diffuse responsibility borders. Instead it is emphasized that the same responsibility allocation that exists in the organization should also be reflected in the responsibilities for the Information infrastructure. In that way each organizational department will be more autonomous and have a freedom of action over the part of the information infrastructure that concerns their own work (Axelsson and Goldkuhl, 1998, p. 45).

While each function of the organization has responsibility for the Information systems concerning their work, VBS also realises that someone needs to have the responsibility for the whole information infrastructure. According to VBS this responsibility should be put on the board of the organisation. This is because the management of the whole II requires a holistic perspective over the organization and it’s overall goals. In this responsibility is also included to make sure that the overall concepts, terms and objects in the Information systems are the same and that they are representable for the whole organization (Axelsson and Goldkuhl, 1998, p. 51).
2.3.3 PAKS

PAKS is a process-oriented approach to structuring information systems (Axelsson and Goldkuhl, 1998, p. 184). In this view it is the working activities and the desirable results that should determine the structuring of systems (Axelsson and Goldkuhl, 1998, p. 185).

Goldkuhl and Axelsson has defined three pillars that PAKS is based on. These are: “Process thinking”, Activities, and Components (Object orientation) (Axelsson and Goldkuhl, 1998, p. 185).

“Process thinking” is emphasizing the horizontal processes rather than the vertical functions. The horizontal means “what is being done” and the vertical is “who is in charge” (who has responsibility) (Axelsson and Goldkuhl, 1998, p. 186).

With the pillar “Activities”, Goldkuhl and Axelsson wants to emphasize that communication also are actions, or more specifically communicative actions (Axelsson and Goldkuhl, 1998, pp. 189-191).

Components means that you should use an object-oriented approach. Object orientation is a programming paradigm in which you strive to encapsulate functionality and data in different objects that are interacting with each other through the exchange of messages. Object orientation also makes possible a hierarchy of objects that can inherit functionality and properties from each other which increases code reuse (Axelsson and Goldkuhl, 1998, p. 191). As opposed to a strictly data-driven approach — advocated by for example IRM — where data and functionality are separated, the OO approach instead keeps these together.

Information systems are in this approach seen as a means to perform actions and the actions are the central in ISs. According to this view ISs consists of “action potential”, the “performing of actions”, and “action memory” (Axelsson and Goldkuhl, 1998, pp. 194-195).

Action potential means the actions that the system permits it’s users to perform. The “performance of actions” are the actions that are performed using the system and the “action memory” is keeping necessary records of what actions has been performed in the system (Axelsson and Goldkuhl, 1998, pp. 195-196).

Information in this view is also seen as communicative actions rather than only as a mirror image of reality (IRM) or information supply (VBS) (Axelsson and Goldkuhl, 1998, p. 194). The information infrastructure is seen as a holistic solution for supporting activity processes (Axelsson and Goldkuhl, 1998, p. 206).

2.4 Design principles for IIs

In order to make different IT systems interoperate, Hanseth and Lyytinen, propose another approach which is not a strategic planning approach. Instead this approach is based on the idea that II development cannot be planned, instead a number of guidelines for II-evolution have been proposed that could guide the design of IIs that are both useful and interoperate.

1. Design initially for usefulness
2. Draw upon existing installed bases

3. Expand installed base by persuasive tactics

4. Make it simple

5. Modularize by building separately key functions of each infrastructure, use layering, and gateways

(Hanseth and Lyytinen, 2004, Table 4 - “Design principles for a design theory of II”, pp. 233)

**Design initially for usefulness:**
It is acknowledged that initial users are facing greater risks and costs investing in completely new IT solutions. Therefore, it is important to make the platform cheap, easy to learn and give maximum benefits for the first users in order to make the investment worthwhile (Hanseth and Lyytinen, 2004, p. 222).

**Draw upon existing installed bases:**
This principle suggests that you should use supporting infrastructures that are already used among the potential target group because introducing completely new supporting infrastructures could seriously build barriers to the adoption of the new infrastructure (Hanseth and Lyytinen, 2004, p. 224).

**Expand installed base by persuasive tactics:**
Once a small user group has started using the simple initial version of the infrastructure, efforts should be placed on expanding this user base quickly rather than developing new functionalities. This principle can be summarized by the slogan “users before functionality”. This is because “an infrastructure will gain value primarily through the size of it’s user base, and not from the extensive quality of it’s functionality” (Hanseth and Lyytinen, 2004, p. 224).

**Make it simple:**
In order to avoid technological lock-ins, it is vital that the II has a good amount of flexibility to change. This principle therefore says to develop the information infrastructure as simple as possible because “it is easier to change something small and simple than something large and complex” (Hanseth and Lyytinen, 2004, p. 225).

**Modularize by building separately key functions of each infrastructure, use layering, and gateways:**
It is also important to build the infrastructure by a clean decomposition into several subs-infrastructures where each of these exposes clean interfaces “so that they can evolve independently of each other”. The use of gateways to connect different parts of the infrastructure is also suggested because it decreases the dependency in the infrastructure and thus decreasing the risk of technological lock-ins due to the additional flexibility (Hanseth and Lyytinen, 2004, p. 225).

This design principle is specifically designed to resolve the problem with technological lock-ins created by dependencies that was previously explained in section 2.2. As we will see, it is thus in principle referring to the design philosophy “SOA” (Service Oriented Architecture). We will in the next section explain what SOA is and what role it has in the design of information infrastructures.
2.5 Service Oriented Architecture

In order to understand SOA, an understanding of the inherent terms; “Service”, “Service orientation” and “Architecture”, must first be acquired. We will here go through each of them.

A **Service** is in everyday life used to refer to “an act of helpful activity” (Dictionary.com, 2014b). In IT, Services can be seen as building blocks, each of which is a representation of a repeatable business activity (OpenGroup.org, 2014a). A Service is self-contained and can be composed of other services. When talking about Services in the form of software there are two types of agents. The agents providing the service are called “providers” while the ones making use of a Service are called “consumers”. A Service can be seen as a black box for consumers, which means that it hides implementation. It combines information and behavior and exposes a relatively simple interface (Sprott and Wilkes, 2014).

**Service orientation** is a design paradigm revolving around the theory “separation of concerns” which means that a larger problem can more easily be solved if it is divided into a set of smaller problems (ServiceOrientation.com, 2014a). The benefit of this is that we can create “building blocks” of implementations that are designed to solve smaller coherent concerns (ServiceOrientation.com, 2014a), and then combine these however we like to solve greater problems while reusing already built solutions. For this to be made effortlessly, Service orientation includes a set of principles listed below:

- **Standardized Service Contract** - “Services within the same service inventory are in compliance with the same contract design standards.”
- **Service Loose Coupling** - “Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment.”
- **Service Abstraction** - “Service contracts only contain essential information and information about services is limited to what is published in service contracts.”
- **Service Reusability** - “Services contain and express agnostic logic and can be positioned as reusable enterprise resources.”
- **Service Autonomy** - “Services exercise a high level of control over their underlying runtime execution environment.”
- **Service Statelessness** - “Services minimize resource consumption by deferring the management of state information when necessary.”
- **Service Discoverability** - “Services are supplemented with communicative metadata by which they can be effectively discovered and interpreted.”
- **Service Composability** - “Services are effective composition participants, regardless of the size and complexity of the composition.”

(ServiceOrientation.com, 2014b)

**Architecture** can refer to a few different things. Outside of IT, it can for example refer to the following things: “the character or style of building”, “the action or process of building”, “the result or product of architectural work”, or “buildings collectively” (Dictionary.com, 2014a). In
IT, it has a very similar meaning except that the product of the architectural work is either computer hardware or software related.

Because of some rather wide inherent concepts that SOA is made up of, there has been many contradictory understandings of what SOA really is (Sprott and Wilkes, 2014). SOA can on the one hand be seen as a type of architecture that is resulting from the application of Service orientation (Arsanjani et al., 2014). On the other hand it can be seen as an architectural approach that focuses on “building systems from autonomous services” (Linthicum, 2014).

In this thesis we will use this rather inclusive definition of SOA:

“The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface (CBDI, cited by Sprott and Wilkes, 2014)”.

The main benefit of SOA is that it delivers “enterprise agility” which means that it is adaptable to changing business needs and requirements (Opengroup.org, 2014b). This “enterprise agility” is a collective name for many different ways that an enterprise can be agile, each which requires different measures to accomplish. Some examples of properties that makes the enterprise agile are: functionality can be re-combined, new functionality can be rapidly developed, operations can be scaled to meet different demands (Opengroup.org, 2014b).

A SOA can deliver many of those agile properties but they require different and costly supporting infrastructures so it has to be chosen carefully according to the business needs (Opengroup.org, 2014b).

As we have learned, SOA is not a technology but rather a design philosophy and needs other technologies and standards to be realized. SOA can for example be realized with the help of Web Services, REST(Representational State Transfer) and SOAP (SOA Protocol). However, these are beyond the scope of this study.

2.6 The problem with eliciting requirements

The IT intensive world we live in demands an increasing involvement of system users in the decision making process and design of ISs and IIs in organizations (Weistroffer, 2009). This is because these systems are their primary and everyday working tools which makes them critical to maintaining efficiency for the organization. Systems like these that have poor usability or that is not properly supporting the activities of the organization can therefore act as a bottle neck for efficiency for the organization and quite possibly become a working environment issue for it’s employees due to a lack of user satisfaction.

But information systems and in particular information infrastructures are complex phenomena that can be difficult to understand for system users provided only conventional methods including models and descriptions.

One of the most difficult problems in Information Systems Development is the gathering and elicitation of requirements. As Brooks expressed this in his well known “No Silver Bullet” article:
“The hardest single part of building a software system is deciding precisely what to build” (Brooks, 1987, p. 17).

Brooks also argues that no other part in the development process can have such “crippling” consequences if done wrong and no other part is so difficult to correct later on (Brooks, 1987, p. 17).

This shows just how important and how difficult the process of requirements elicitation is. So what is so difficult about this process?

First of all, as Agnis Stibe and Janis Becevskis (2009) explains; the gathering of requirements can be done with three different approaches.

The first approach being that the “requirements are formulated by business-oriented people, who are less experienced in information technology (IT) issues” (Stibe and Bicevskis, 2009, p. 8). This approach often results in informal, incomplete and inconsistent requirements that the software developers find difficult and diffuse to work with.

The second approach is that the “requirements are formulated by IT specialists” (Stibe and Bicevskis, 2009, p. 8), who on the other hand does not have adequate knowledge of the organization and it’s processes. This leads to “very well defined” and formulated requirements, but might not correspond to the actual needs of the organization (Stibe and Bicevskis, 2009, p. 8).

So to fix this problem software developers came up with the solution to build teams consisting of both IT specialists and Business-oriented people in order to get “the best of two worlds”. But now the only problem is that these two kinds of people speak different languages. That is, they use different terminology and have different knowledge and understanding of things (Stibe and Bicevskis, 2009, p. 8). So how are these two going to communicate?

The standard way of illustrating different structures in ISs is by using UML (Unified Modelling Language). This language is of course well known to the IT specialists but is not very straightforward and easy to grasp for the business people (Stibe and Bicevskis, 2009, p. 8).

So a big part of the problem of gathering requirements is that the people who build the IT solutions and the people who are in need of- and are going to use the system have a hard time understanding each other, which causes misunderstandings when the software is developed.

Another part of the problem, which is also illuminated by Brooks (1987), is that “the client does not know what he wants” (Brooks, 1987, p. 17). This also means that no models or descriptions in the world will help users understand the implications of such ideas which means that they are not adequate means of gathering requirements. Brooks therefore emphasizes the importance of the continuous and iterative communication between client and designer, which is also a core principle in agile software development methodologies (Beck et al., 2001).

Brooks goes even further arguing that “it is really impossible for a client, even working with a software engineer, to specify completely, precisely, and correctly, the exact requirements of a modern software product before trying some version of the product” (Brooks, 1987, p. 17). Brooks mentions Prototyping as one of the most promising efforts of dealing with this problem. In the next subsection Prototyping will be examined and what place prototyping has as a tool in software development.
2.7 Prototyping

Software Prototyping is a tool used in software development that enables developers to, in a quick way, illustrate certain aspects or ideas of a software solution to their customers. In doing so, they can get early feedback from the customer before the actual system is built thus increasing the chances of success (Arnowitz et al., 2007, p. 3).

There are many different approaches and scopes of prototyping. Prototypes can be as simple as a drawing on a piece of paper or as complex as a piece of software with actual working functionality. A simple paper approach is often referred to as low-fidelity prototyping while a more elaborate approach as high-fidelity prototyping (Arnowitz et al., 2007, p. 86).

There are also two different dimensions of prototyping as mentioned by Jakob Nielsen (1993). These are: Vertical prototyping and Horizontal prototyping. Horizontal prototyping illustrates multiple features of the application but at a shallow level of detail while Vertical prototyping illustrates fewer features but with full functionality (Nielsen, 1993, p. 94).

These two dimensions often serve different purposes. While the horizontal dimension might be more appropriate to illustrate, to the client, an overall idea of how a system should look, feel and work, the vertical dimension is more useful when the developers are interested in testing various technical aspects under realistic circumstances such as database design or the interoperability possibilities between two systems (Nielsen, 1993, p. 95).

The main prototyping types are Throwaway prototyping (sometimes also called Rapid prototyping) and Evolutionary prototyping (Davis, 1992, p. 70).

Throwaway- or Rapid prototyping means that a “quick-and-dirty” prototype is built for the sole purpose to elicit requirements from the client. Once the requirements are gathered the prototypes are thrown away and the actual system is built (Davis, 1992, p. 70).

In Evolutionary prototyping the prototype is seen as a work in progress and are systems that evolve through iterations with the client in order to correct misunderstandings and improve the product after the client’s needs (Davis, 1992, p. 70).

There are also of course an array of different prototyping methods and tools but these are out of the scope of this thesis.

2.7.1 Definition of Prototyping

The following definition of prototyping is used in this thesis:

“Prototyping involves building various representations, or early versions, of an information system, which are shown to clients and end-users in order to get feedback” (Beynon-Davies et al., 2013, p. 373).
2.7.2 Motivation for the use of prototyping

“Prototyping approach eliminates barriers between business and information technology people, leading to common understanding of web site goals and success in delivery and implementation” (Stibe and Bicevsksis, 2009, p. 6).

“it is really impossible for a client, even working with a software engineer, to specify completely, precisely, and correctly, the exact requirements of a modern software product before trying some version of the product” (Brooks, 1987, p. 17).

2.7.3 Prototyping vs Agile methodologies

A distinction has to be made between Prototyping and Agile methodologies such as Scrum and XP. Although similar in many ways, particularly compared to Evolutionary prototyping (where they almost overlap), they are not quite the same.

Agile methodologies are systems developments methodologies that focus on streamlining and making the development process more efficient. This is done by removing unnecessary documentation and other overhead and instead focus on close contact with the customer to get feedback. This is done in short iterations (normally around 2 weeks) that contains a short and concise version of the complete systems development life-cycle (planning, requirements elicitation, design, coding, testing, writing documentation) (Dennis and Wixom, 2009, p. 53).

One of the main individuals behind XP (eXtreme Programming) and the Agile manifesto, Kent Beck, argues that the creation of prototypes is less time effective than the use of XP (Ferreira et al., 2007, p. 1). This is because agile methodologies, such as XP, are designed to embrace changing requirements, even late in the development (Beck et al., 2001). This means that no extra time is spent to develop prototypes but rather, changing requirements are welcomed during the development of the software. This practically means that the requirements elicitation process is made, to a big extent, “on the fly”.

But this way of working assumes that the customer has a relatively clear idea of what should be built. Prototyping instead focuses on gathering requirements as its main purpose, and as such it is mostly used as tool in the requirements elicitation process. It is difficult to start building something “real” if there exists many uncertainties about what should be developed.

Evolutionary prototyping is comparable to agile methodologies because they see the prototype as evolving into something that is going to be used in the end product. However, in evolutionary prototyping it is acknowledged that all the requirements are not well understood and therefore the focus is on developing those that are well understood (Davis, 1992, p. 71) and through user feedback gather more requirements.

Throwaway prototyping is done for the sole purpose of requirements elicitation which means that when it’s purpose has been reached, the prototype can be “thrown away” and the actual development process can start. This has an advantage in situations where the customer has a poor understanding of how the end product should be (Davis, 1992, p. 71). It is less likely that building something with a poor understanding of the requirements will result in something useful.
To summarize, we can say that, in Agile methodologies such as Scrum and XP, the main goal is to efficiently develop working software that meets the customers requirements. Therefore, an Agile project requires a relatively good understanding of what is to be developed in order to be successful.

In Evolutionary prototyping, the main goal is to develop parts of software that can be used in the end product and also to gather more requirements. Evolutionary prototyping acknowledges that all requirements are not well understood and therefore focuses on the parts that are well understood.

Throwaway prototyping is focused solely on gathering requirements and is used when there is a poor understanding of the end product.

2.8 The gap in the literature

To summarize the Literature review; we have seen that although Information infrastructures has theoretically been around since the 90’s, it is still a relatively new field of theory that still is in need of much more research. Even the name of the concept is still quite fuzzy and there are relatively few guidelines and principles for how to successfully build them. This view is supported by Hanseth and Lyytinen who writes “we currently have a dearth of knowledge how to effectively design information infrastructures” (Hanseth and Lyytinen, 2004, p. 208).

But now, at least, we are at a point where it has been fully realized that information infrastructures have to be planned separately to the information systems that it is constituted by and that this is a challenge that needs significant work to master.

We have also seen how the popularity and importance of truly integrated information infrastructures has drastically increased over the passed decade, which makes this a particularly important field to study.

Although we have seen that there are several design philosophies (IRM, VBS, PAKS, SOA) and also some design principles (Hanseth and Lyytinen, 2004) for the development of Information Infrastructures, it is also possible to identify clear gaps in the theory.

The strategies that Goldkuhl and Axelsson are talking about are mostly concerned with the overall structure of information infrastructures and what should determine that structure.

The guidelines that Hanseth and Lyytinen propose are general and are assumed to be applicable for any type of II. They propose a set of guidelines for how to successfully develop a new part of the information infrastructure with concern to the installed base by creating a boot-strapping effect (principles 1 and 3), where the user requirements are considered and guidelines for how to scale up the II by taking into account interoperability requirements and avoiding technological lock-ins (principles 4-5).

SOA is, indeed, a very important design philosophy that enables enterprise agility which is critical for the development and evolution of an information infrastructure that is able to meet the changing needs of an organization.

While these strategies, philosophies and principles may be useful in general design situations,
they do not give any concrete advice as to how the problems concerning the elicitation of interoperability requirements from a business perspective could be overcome. The strategies (IRM, VBS, PAKS) does not help answering, in detail, the questions “which systems should the new system interoperate with?” and “what information should be communicated between the systems?”.

Hanseth and Lyytinen only give advice to use SOA to meet interoperability requirements, not how they could be elicitated.

SOA is only concerned with how business requirements can be met by enabling adaptability to changing business requirements. It does not specifically aid in the process of eliciting interoperability requirements based on business needs before a system is to be integrated into the II.

The purpose of this thesis is therefore to explore the possibilities of using prototyping as a means of elucidating and to elicit requirements of both the to-be developed system and the installed base that this system needs to relate to in order to create successful integration and interoperability of the new system and it’s environment.

The next section will present the methodology used in this study in order to achieve this purpose.
3 Methodology

This thesis is following a mixed research approach consisting of a Design and Creation research strategy and a single Case study approach. The use of two or more research strategies is called strategy triangulation (Oates, 2006, p. 37). Triangulation is said to give the research “multiple modes of attack” on the research question (Oates, 2006, p. 38) which sometimes can give a more valid research result.

3.1 Case study

A case study is a research approach where the research is focusing on investigating a specific instance of a certain “thing” (Oates, 2006, p. 141). In contrast to many other research approaches (e.g. experiment, survey), case studies have the characteristics of giving a deeper understanding of a small part of a general problem but doing this in a very realistic manner. This enables careful consideration to “the messiness of the world” (Oates, 2006, p. 142).

In this paper, the “thing” under investigation is the Information Infrastructure along with the specific problem and the context that is present at Uppsala University. This is considered to be a suitable approach because, when dealing with IT systems, there are always multiple interrelated factors that are affecting and acting on one another (e.g. the installed base of IT capabilities, working routines, politics, “the messiness of the world”).

This is an exploratory case study which means that it is used to explore or define a topic “where there is little in the literature” (Oates, 2006, p. 143). The purpose of the thesis is to define a method to be used in a specific problem case and see what the outcome will be. This type stands in contrast to a descriptive case study which aims to analyse and describe an already existing phenomenon, and an explanatory case study where the purpose is instead to explain why events or outcomes occurred. In short you could say that an exploratory case study is used to define something, a descriptive case study is used to to describe something and the explanatory case study to explain something (Oates, 2006, p. 143).

The reasons for choosing to conduct a case study is both because of convenience and because of a unique and interesting opportunity.

The convenience is due to the easy accessibility to, and previous knowledge of, key individuals within the organization. Also, the fact that the author is a student and an employee at this organization makes this a convenient location.

This particular case is also an excellent and unique opportunity to explore and possibly build upon a particular field of theory that is currently in need of additional theories and research.

The data generation methods used in this case study are interviews and document analysis. This research is therefore also making use of method triangulation (Oates, 2006, p. 37).
3.2 Design and Creation

Design and Creation is a strategy that focuses on the development of new IT artefacts (Oates, 2006, p. 108).

Oates, identifies four different types of artefacts that can be developed in the Design and Creation strategy:

- **Constructs** - Concepts or vocabulary that are used in the IT domain.
- **Models** - Ways to illustrate a certain situation in order to help understanding them.
- **Methods** - Different kinds of guidance, for example steps to follow in order to solve a problem.
- **Instantiations** - Working systems that is developed in order to examine something for example a method or technique in order to make a scientific point.

(Oates, 2006, p. 108)

This thesis is concerned with the development of two of these artefacts.

Generally, the thesis is concerned with the development of a method that can be used to elucidate and elicit requirements of both the to-be IS and the II that it needs to be integrated into. However, it is also concerned with the development of an instantiation in the form of a prototype.

While the method is constituted by the specific way that prototyping is used to elucidate and elicit requirements of the installed base the instantiation merely serves as a component of the method.

The Design and Creation strategy is typically used in order to solve a problem (Oates, 2006, p. 111). The strategy is therefore based on five steps that are followed iteratively in order to solve that problem. These steps are:

- **Suggestion** - Is where an idea about the solution to that problem is defined.
- **Development** - Is the implementation of the solution. This is where the IT artefact is developed.
- **Evaluation** - Is where the developed artefact is assessed in terms of it’s worth and how well the outcome matched the expectations.
- **Conclusion** - The results are formulated and the new knowledge is identified.

(Oates, 2006, pp. 111-112)

The method under development has the purpose of solving the problem to successfully integrating a new IS into an existing II. It is this method that is the main product of this thesis.
The prototype that is developed is in the vertical dimension of prototyping which means that it is a more elaborate and complete prototype. This type of prototype was chosen because it is more useful when gathering technical as well as business requirements. The alternative was building it in the horizontal dimension which only illustrates an overall idea of the look and feel of a system and does not capture any technical requirements. This thesis emphasizes the use of prototyping to unravel such technical uncertainties as integration and interoperability between systems and therefore the vertical dimension of prototyping was found more appropriate.

3.3 Data generation and analysis

As mentioned earlier the data generation methods used in this thesis are document analysis and interviews in the form of demonstrations and meetings. These will here be described in more detail and a list of participants will also be presented.

3.3.1 Document analysis

The documents that was analysed was partly a previous thesis (Fiske and Eklund, 2013) but also various system documentations. The system documentation was for the systems Selma, TimeEdit and Ladok. The reason for the analysis of these documents was in order to investigate the possibilities for integration between the prototype and these systems and to understand the architecture and concepts they use.

3.3.2 Interviews

The interviews was made in the form of seven demonstration sessions and several meetings. The participants are here listed with their respective job role and will from here on be identified by the specific given participant code (P1-P9).

P1 - IT coordinator, Department of Engineering Sciences
P2 - Systems developer, IT Division, The Student Portal
P3 - Systems developer, IT Division, The Student Portal
P4 - Director of Studies, Department of Business Studies
P5 - Director of Studies, Department of Informatics and Media
P6 - Administrator and Teacher, Department of Informatics and Media
P7 - Senior Lecturer, Department of Informatics and Media
P8 - Junior Lecturer, Department of Informatics and Media
P9 - Junior Lecturer, Department of Informatics and Media
3.3.3 Demonstrations

The demonstrations lasted about an hour each. They were recorded and transcribed. The meetings also took around an hour each but only three of these meetings was recorded but were not transcribed.

The recordings was transcribed where the discussion was deemed important for the study. This was mainly when different functions and features was discussed but also when different problems was discussed that could help the author build an understanding of their nature and causes. Some positive and negative feedback on the prototype was also transcribed in order to get the general impression or level of enthusiasm towards the system. This resulted in 36 pages of transcribed material.

3.3.4 Analysis of transcriptions

The analysis of the transcriptions was mainly qualitative data analysis but it also had some quantitative elements. The process of analysing the recordings was as follows:

1. From the transcribed material, business requirements and requirements concerning the information infrastructure was extracted and placed into different categories that was identified during this process.

2. The specific requirements was further categorized into more specific functions.

3. The requirements was then analysed to see patters such as functions that had been more frequently emphasized, and to find what people were most positive and negative about.

4. The data was summarized into requirements and their importance as well as overall feedback to this type of system.

3.4 Implementation/Approach

The research started with several meetings with the supervisor of the study, who over the years has developed a deep understanding of the information infrastructure at Uppsala University and identified several problems. In this study the supervisor should be seen as the client of the prototype and also in his role as a main teacher.

These conversations served as a starting point for the study and as an opportunity for the author to develop an initial understanding of the problems experienced at Uppsala University. In this stage the author also engaged in some document analysis related to these issues. Relating back to the steps in Design and Creation mentioned earlier this corresponds to “Awareness” which is the first step in this strategy.

The initial meetings with the client also included some analysis of excel spreadsheets that he had developed. These documents illustrated how it was possible to resolve parts of the problems with a new database structure. These illustrations took the form of a set of database tables.
and their relationships. This process corresponds to the “Suggestion” step in the Design and Creation strategy.

When sufficient understanding of the problem and of some possible solutions had been acquired, the development of a prototype began. In the Design and Creation strategy this step is called “Development” but this process also involved the step “Suggestion” because the development also resulted in new ideas and solutions.

The prototype was developed by the author in the ASP.NET MVC Framework which is a technology for building Web based applications. ASP.NET is based on Microsoft’s .Net platform and makes use of C# (an Object oriented programming language), Javascript, CSS, HTML and other mark-up and programming languages. It also makes use of a programming design pattern “MVC” (Model View Controller) which aims to separate concerns into three layers; Models, Views and Controllers. This is an old and proven design pattern which has been shown to increase flexibility and testability in software (W3Schools.com, 2014).

**Model** is a data layer for business logic, here you find classes that take care of various data operations and data access.

**View** is the presentation layer where the web pages exist and is therefore responsible for the GUI (Graphical User Interface) of the application.

**Controller** is a special layer for taking care of the flow in the application. Here you define the interaction between the Model and View layers and the user interaction (W3Schools.com, 2014).

The process of developing the Prototype was done in parallel with analysis of various documents and descriptions of the different systems in order to come to understand their API and how the interoperability between the prototype and the respective systems could be achieved. This process has an important role in the proposed method because in this process many requirements are made clear because the implementation makes visible various problems in terms of interoperability that arguably can be very difficult or even impossible to identify before any development has begun.

After the implementation of each feature of the prototype, the work was evaluated together with the client in order to ensure the correctness of these features.

The software development methodology that was used in the “Development” step was based on the authors previously acquired knowledge of software development methodologies. This development methodology can be classified as an agile development approach. In practice this meant an iterative approach consisting of a set of features (similar to “product backlog”) that was continuously implemented and closely followed by testing and meetings with the customer. The customer in this case being the supervisor of this study and occasionally other relevant stakeholders within Uppsala University.

This process continued until the prototype was at a stage where it was possible to demonstrate a complete flow of operation which was deemed enough to illustrate the usefulness of such a system.

The study then went into another phase which can be seen as both “Evaluation” and “Conclusion”. Namely, the demonstration of the prototype to various stakeholders including Directors of Studies and people responsible for various systems in the information infrastructure. This was
a key phase where many of the requirements were elicited.

The demonstrations followed a specific predefined case designed to simulate a task that one would normally want to carry out in a system like this. This case was also designed to elucidate some of the benefits of such a system and also showing some properties and functions not present in the existing systems.

The meetings where the demonstration took place was recorded with the consent of the present individuals. These recordings were then analysed in order to capture all the requirements that was elicited as a result of the demonstration.
4 The Context at Uppsala University

In order to fully understand the problems at Uppsala University, their underlying causes, and their practical implications, it is vital to first get an understanding of how the organization operates and in particular how the employees work (and cannot work) with the IT support today.

This section is to a great extent based on a previous study by Max Fiske and Michael Eklund (2013) who carried out an in depth case study of the information infrastructure at Uppsala University.

We will start by looking at the organizations business definition which is then followed by a description of the systems that make up the information infrastructure at Uppsala University. We then go through the organizations core processes or what is also referred to as the activity system and then we go through some important business concepts that is used when talking about education in the organization. The section is concluded with a brief summary.

4.1 The Organization

Uppsala University is a Swedish University established in 1477 in the city of Uppsala. This makes it the oldest University in the Nordic countries. The University’s goals are; to provide top quality education and research, be open minded and adaptable to change, have an active role in the global society and support development and innovation, strengthen it’s position as a world leading University and contribute to a better world (University, 2014).

With it’s 6,505 employees, Uppsala University is currently teaching around 40,000 students. The University has 51 faculties and departments and the annual turnover is 5,9 billion SEK (University, 2014).

The following table shows the University’s business definition on a local level.

<table>
<thead>
<tr>
<th>Norm setters</th>
<th>1. The State 2. Uppsala University 3. Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality norms</td>
<td>Regulations, Uppsala University’s Policies, Approach propositions</td>
</tr>
<tr>
<td>Producers</td>
<td>Teachers, Directors of Studies, Heads of Departments, Administrators</td>
</tr>
<tr>
<td>Actions</td>
<td>Budgeting, Accounting, Create course plans, Teaching</td>
</tr>
<tr>
<td>Tools</td>
<td>The available Information systems</td>
</tr>
<tr>
<td>Products</td>
<td>Education for students, Research, Academic knowledge contribution</td>
</tr>
<tr>
<td>Clients</td>
<td>Students, Researchers</td>
</tr>
<tr>
<td>Effect</td>
<td>A better educated public</td>
</tr>
<tr>
<td>Resource mission provider</td>
<td>Uppsala University</td>
</tr>
</tbody>
</table>

(Table 3 Fiske and Eklund, 2013, p. 35)
4.2 The Information Infrastructure

As mentioned by Fiske and Eklund (2013), the information infrastructure at Uppsala University can be described on two different levels. Referring back to Hanseth and Lyytinen’s classifications of IIs (Hanseth and Lyytinen, 2004) — these two levels correspond to a national “Business Sector Infrastructure” and a local “Corporate Infrastructure”.

The corporate infrastructure consists of the university’s own collection of information systems and their relationships with each other while the Business Sector Infrastructure consists of the collection of national information systems for Swedish Universities and how these two infrastructures relate to each other (Fiske and Eklund, 2013, p. 2).

More specifically, the information infrastructure at the local corporate level consists of a financial system, a student platform, an LMS-system (Learning management system), an education database, a local student directory, a planning and follow-up system, a decision support system (DSS), and a room booking system (Fiske and Eklund, 2013, pp. 25-28).

The national, business level of II consists of an admission system, an education registration website, and a database storing information about University educations in Sweden (Fiske and Eklund, 2013, p. 28).

This study is mostly concerned with the corporate level of information infrastructure since this is where the proposed planning system needs to fit in. The corporate level information infrastructure is illustrated in the systems infrastructure diagram below (figure 1).

![Diagram of Systems on a local level](image)

Figure 1: Systems on a local level (Fiske and Eklund, 2013, p. 31)
4.2.1 Corporate level information infrastructure

The local Information systems are listed and described below:

- **Selma** - Manages Uppsala University’s education database. It contains education plans, course plans, education offers, and course instances. This is also where education offers and course instances are created along with lists of course literature. Selma is providing information to Antagning.se, Studentportalen, and Uppdok (Fiske and Eklund, 2013, p. 25).

- **Uppdok** - Is Uppsala University’s own student directory which is a local version of the national equivalent Ladok. The database is keeping information about students and their achievements, education applicants, completed studies, grades and degrees. This information is used both for statistics and admission of students (Fiske and Eklund, 2013, p. 26-27).

- **Raindance** - This is the University’s financial system. It is developed by an external company but managed by the financial department at Uppsala University. This system is providing the University with functions for financial accounting, planning, management and monitoring in a cost-effective way (Fiske and Eklund, 2013, p. 27).

- **UU-Plus** - Is the university’s planning and follow-up system used for planning personnel budgeting and the availability of teachers. This system is going to be discontinued within an indefinite time frame so no efforts are currently being put into the development of this system (Fiske and Eklund, 2013, p. 27).

- **GLIS** - A web based Decision Support System that is using data concerning Uppsala University’s financial situation, personnel, rooms, students and publications in order to help the decision makers take the right decisions (Fiske and Eklund, 2013, p. 27).

- **Studentportalen** - IS an extensively used student platform as well as a teaching platform for teachers. As a teaching platform it offers a variety of tools such as the ability to track students’ progress based on different course components and get various statistics from previous course evaluations. This is also where students can find their course schedules and find the list of course literature. Apart from this, the platform is also used for examination registrations and occasionally also for students to register on courses (Fiske and Eklund, 2013, p. 25).

- **TimeEdit** - Is the local room booking system used to book different types of rooms owned by the University including lecture halls, classrooms and lab rooms. This system manages teachers and rooms as resources for the University’s teaching. TimeEdit also manages an on-line schedule where the bookings can be made visible given certain criteria, for example for a certain course or for a certain teacher (Fiske and Eklund, 2013, p. 28).

4.2.2 Business level information infrastructure

The national ISs are listed and described below:

- **Ladok** - Is the national student directory. It’s database is keeping names, civic numbers, eligibility, course registrations, admissions to programmes and courses, results from courses and examinations, and completed studies and degrees (Fiske and Eklund, 2013, p.27).
• **NyA** - Is the national admission system for universities in Sweden. This system is used in the selection process for education and stores information related to students achievements such as grades, completed studies, and results on the Swedish Scholastic Aptitude Test (“Högskoleprovet”) (Fiske and Eklund, 2013, p. 28).

• **UHR database** - a database that provides information about higher education as well as the Swedish Scholastic Aptitude Test (Högskoleprovet). UHR manages admissions and provides support for student administration. UHR also evaluates foreign qualifications (Fiske and Eklund, 2013, p. 28), (UHR.se, 2014)

• **Antagning.se** - Is a separate website where admissions for educations at Swedish universities are made. This website includes functions such as searching for educations and the students are able to prioritise the applied for courses. It is through this website also possible to get information about the different courses, this is accessed from systems owned by the respective university offering the course (Fiske and Eklund, 2013, p. 28).
4.3 The Activity System

The activity system of an organization is equivalent to its core processes. This section will look at the core processes at the Department of Informatics and Media and they are as follows:

- **Establishment of new educations** - This is the process of reacting to a student demand for a course or programme and consists of the development of course- or education plans. The plans can lead to an established course/programme or be rejected.

- **Student recruitment** - The process of recruiting students starts with a decision about what education should be offered to the students. The students then place their applications to their desired educations on a certain website and is followed by the student selection process.

- **Budgeting and resource planning** - This process is mainly the Director of Studies responsibility and consists of creating a rough budget on a subject basis as well as more finely grained budgeting on a course instance level. In this process the teachers are also planned on the different courses and rooms are booked by the main teachers.

- **Course implementation** - Is where the service is delivered to the students. This process consists of teaching, communicating various learning materials and other information to the students through Information systems, and grading various assignments and examinations.

- **Registering grades** - The grades are registered into Uppdok/Ladok in order to keep track of the students achievements and for use in various monitoring processes related to the University’s performance and budgeting.

- **Reporting** - At the end of each course instance, the Director of Studies needs to evaluate the budget as well as the outcome of the course instance. This is done with the help of a course report and a report about each teacher’s worked hours. It is the Main teachers responsibility to provide the Director of Studies with this information.

Next, we will go through these processes with the help of flowcharts and more detailed descriptions about how these processes are carried out at the Department of Informatics and Media.
4.3.1 Establishing of new education

An education offer is created as a reaction to a demand for such an education from potential students. When the demand is high enough an education plan or course plan needs to be created before this offer can be made available to the students. These plans are developed by main teachers and teachers and is then sent to the faculty board who decides whether the education should be established or not (Fiske and Eklund, 2013, pp. 35-36).

When new educations have been established they are entered into to the Selma database where course plans, education plans, course- and programme instances, and lists of course literature are stored (Fiske and Eklund, 2013, p. 36).

Figure 2: The process of establishing new educations.
4.3.2 Student recruitment

Before each application period a decision about which education offers should be made available to the students is made. When this decision has been made, all the educations that should be made available are sent from the Selma database to the UHR database and then to the system Uppdok (Fiske and Eklund, 2013, p. 36).

The students are then able to apply for courses and programmes they wish to participate in on the website Antagning.se which uses the offers stored in the UHR database. Antagning.se is a website hosted by a server owned by UHR (Fiske and Eklund, 2013, p. 36).

When the students have submitted their applications on Antagning.se the admission process is started. The selection process is done with the help of the IS NyA that is also using information from the UHR database. An admitted student is identified with his or her personal ID. When
the admission process is completed the information is sent to Uppdok (Fiske and Eklund, 2013, p. 36).

Uppdok stores all the course instances. When the information about the admitted students arrives to Uppdok, the education offers and admissions are related to their corresponding course instances. Before the students start their education they need to register to the course instances for their coming semester (Fiske and Eklund, 2013, p. 36).
4.3.3 Budgeting on a subject level

Budgets need to be created for course instances, room and personnel costs. The existing planning system at Uppsala University only allows for a rough budgeting on a subject basis. But the Directors of Studies need to have a budget on a more detailed, course instance, level too. In these cases the Directors of Studies need to rely on self-developed systems which in many cases means the use of simple Excel spreadsheets.

Every subject has a Director of Studies. One of their responsibilities is to plan courses for their specific subject. When a course is to be planned the Director of Studies first needs to check the number of full year places for the current year and subject. They do this in order to calculate the revenues that is available to use for coming educations.

This calculation is done by multiplying the annual cost for one student with the number of registered full year students. On the basis of the calculated revenue available for the subjects educations, the Director of Studies now calculate the available teaching hours.

The calculation of available teaching hours is done by dividing the available revenue by a rough estimation of the hourly cost for a teacher. When this is done, a budget for each course instance is created.

Figure 4: The process of budgeting on a subject level.
4.3.4 Budgeting on a course instance level

Figure 5: The process of budgeting on a course instance level.

The course budgets are based on all planned courses for the year, and are created manually in an excel spreadsheets. For each course instance the number of students and number of points the course constitutes are entered in, and based on this the number of teaching hours for each course is calculated.
4.3.5 Teacher planning

Together with the teachers the Director of Studies then decides which teachers should teach in which course instances and who will be the main teachers.

In the excel spreadsheet the planned hours for each course are then distributed to the respective teachers based on both the total number of planned hours for the course instance and the total number of hours that each teacher have an agreement to fulfil.

Before the course starts, the Director of Studies sends the respective course budget, via e-mail, to the main teacher of that course.

Figure 6: The process of planning teachers on course instances.
4.3.6 Room booking

It is then the Main teacher’s responsibility to create a schedule and to make a booking request for all the rooms that are needed during the course implementation.

The Booking Agent, who is responsible for and authorized to work in the room booking system TimeEdit, first sends an Excel form to the main teacher via e-mail. When the main teacher receives this form, the booking requests are entered in and the form is sent back to the booking agent via e-mail. The Booking Agent then makes a booking request in the system TimeEdit and after a while receives a response containing all the rooms and times that were successfully booked.

The Booking Agent then notify the main teacher about the successful bookings and the teacher can verify the booked rooms through TimeEdit’s web interface. On the basis of this the Main teacher can then start developing a schedule for the course instance. This schedule is today self developed in either Word or Excel. When the schedule is done it is uploaded to Studentportalen so that it can be accessed by the registered students.
4.3.7 Course implementation and follow-up

During the course implementation, some of the communication between students and teachers is done via the system Studentportalen. This communication is consisting of students accessing learning materials, schedules, various assignments and exercises or general information that the teachers have uploaded. In Studentportalen, it is also possible to set up forums for the students in a certain course to discuss the contents of the course. Normally, Studentportalen also keeps track of student groups such as project work groups or lab groups. For this communication to be possible, information about students and course instances needs to have been transferred from Selma and Uppdok to Studentportalen (Fiske and Eklund, 2013, p. 36).

The courses are examined in the form of various assignments, project works, and examinations that needs to be graded by the teachers. When the grading is done this is registered in Uppdok (Fiske and Eklund, 2013, p. 36).

After the completion of a course instance, a course evaluation form is set up for the students to fill in and based on the result from this and the students study results, the main teacher is creating a course report (Fiske and Eklund, 2013, p. 36).
4.3.8 Reporting of worked hours

![Diagram showing the process of reporting worked hours]

After course completion the main teacher summarizes and verifies the number of hours that teachers have worked on that course instance and compare it to the budget for the course instance. For this to be possible each teacher needs to keep track of how many hours they have worked and by request send an e-mail or arrange a meeting with the Main teacher to discuss the teachers hours. The Director of Studies sends the Main teacher an e-mail request with an Excel form where the reported hours should be filled in. The reported hours are then sent from the Main teacher to the Director of Studies who again updates the excel sheet and reflects on the difference between the budget and the actual outcome so that it can be adjusted for next years budget.

4.4 Important concepts and definitions

As in every organization, there are an array of concepts in use at the University. Of course there are too many concepts in use to describe all of them so this section will provide a list of the concepts that are most central for this study and describe them along with their original Swedish name, in order to avoid confusion.

- **Course instance** (Kurstillfälle) - Specification of certain admission and / or registration options for the course. A course instance is defined by course, the course instances time period, start semester, course pase, course location, course time, education form, financing form and course round (mp.uu.se, 2014).

- **Programme** (Program)- Undergraduate and graduate level where a number of courses have been combined, leading to a degree (mp.uu.se, 2014).

- **Programme Instance** (Programtillfälle) - Specification of certain admission and / or registration options for applications. A programme instance is defined by the programme,
starting semester, programme pase, programme location, education form, and financing
form (mp.uu.se, 2014).

- **Course** (Kurs) - The largest unit within education on basic and advanced levels which are
  assigned a grade. All education at basic and advanced levels will be conducted in the form
  of courses. See also course-and curriculum. Course may also refer to a part of the training
  (test) at the graduate level (mp.uu.se, 2014).

- **Course Package** (Kurspaket) - Two or more course instances that have been joint into
  one registration option. A student who apply for a course package can be admitted for all
  courses included in the package simultaneously (mp.uu.se, 2014).

- **Course Plan** (Kursplan) - A required document that describe and contain regulations for
  a course (mp.uu.se, 2014).

- **Education Plan** (Utbildningsplan) - A required document that describe and contain
  regulations for an education programme (mp.uu.se, 2014).

- **Joint Study** (Samäsnings) - Two or more course instances that are scheduled so that
  they are coordinated during the semester can be defined as a merged course, a joint study
  instance. This means that two or more course instances use the same file area or the same
  discussion forum etc (Uadm.uu.se, 2014).

### 4.5 Context summary

In this section we have tried to provide a better understanding of the context in which this
case is taking place. First a brief description of the whole organization was presented followed
by a detailed look at the Information Infrastructure consisting of all Information systems that
make up the IT Support at Uppsala University. We also looked at the Activity System at the
Department of Informatics and Media and how these processes are carried out today and lastly
the important concepts and their definitions was presented.

Hopefully with a better understanding of the context at Uppsala University and how work is
being done there today, we will in the next section present some arguments for why this way of
working is not desirable.
5 Problem Description

In this section the problems that exists at Uppsala University concerning the Information Infrastructure that affects the Activity system is described followed by the practical implications these have for the employees.

5.1 Problems in the Information Infrastructure

In the Literature review, we learned that some common problems in Information infrastructures are different kinds of redundancy, inconsistencies, duplication, and isolated islands of systems which all leads to a bad fit between the IT support and the organizations activities. In this section we will look at what problems can be identified in the information infrastructure at Uppsala University. These problems are listed and then described next.

- Isolated island TimeEdit
- Lacking and misguided conceptual apparatus
- Lacking system documentation (Selma, UU-plus)
- Lacking API’s (Selma)
- Missing information systems
- Non user-centered development
- Incapability of affecting systems

5.1.1 The isolated island TimeEdit

Today, the system TimeEdit is not communicating with any other system. As we saw earlier, explained by Goldkuhl and Axelsson (Axelsson and Goldkuhl, 1998) and Beynon-Davies (Beynon-Davies et al., 2013), isolated islands of systems have a lot of negative effects. These negative effects include; duplication of data (or redundancy), unnecessary maintenance, unnecessary manual work and inconsistencies in the information stored. In this case we can see, judging from the way work is carried out (See figure 7), that this is causing the following problems:

- Manual work - Main teachers has to create their own schedules in Word or Excel
- Duplication of Data - Schedule information exists both in TimeEdit and are maintained by teachers in the form of Word or Excel documents.
- Inconsistencies - As a result of manually maintaining a separate schedule on the basis of information in TimeEdit, errors and inconsistencies are very easily introduced due to the human error factor.
5.1.2 A misguided conceptual apparatus

We also learned earlier, that the conceptual apparatus is a very important element of a successful information infrastructure (Axelsson and Goldkuhl, 1998). This was explained in section the section Common issues in Information Infrastructures (see section 2.2) and also emphasized in the strategies IRM, VBS, and PAKS (see section 2.3). The problem with the conceptual apparatus is that it is somewhat diffuse and according to the author, and others, misguided. This is mainly concerning the concepts “Course instance” and “Program instance” (Kurstillfälle, Programtillfälle) which is on a national scale defined, by the Ladok consortium, as follows:

**Course instance** (Kurstillfälle) - *Course instance (Kurstillfälle) - Specification of certain admission and / or registration options for the course. A course instance is defined by course, the course instances time period, start semester, course pase, course location, course time, education form, financing form and course round* (mp.uu.se, 2014).

**Program instance** (Programtillfälle) - *Specification of certain admission and / or registration options for applications. A programme instance is defined by the programme, starting semester, programme pase, programme location, education form, and financing form* (mp.uu.se, 2014).

The problem with these concepts is that the names and in particular the word “instance” (tillfälle) suggests that these concepts are related to the actual activities of the specific course, while the definitions show that they are just specifications of a certain admission made by the students, or a collection of courses that makes up an education at a specific time. But what if, for a certain course or programme at a specific time, there are too few applicants to even make the carrying through profitable for the University. In this case the programme or course will naturally be cancelled and there will be no actual carry through, there will be no “instance”.

This shows that the term “Course instance” and “Programme instance” is probably not suitable for what stands for today. This also elucidates the lack of an established concept, let alone a specific information object in the organization for the actual “service” or follow through of the education. This also creates problems because, in default of something better, “course instance” (kurstillfälle) is today used as if it represented the actual follow through of the course. This is true in, for example, Studentportalen where activities are tied directly to the “Course instances” which we have seen represents the admission of the course rather than it’s carrying through.

5.1.3 Lacking documentation

There are systems in the information infrastructure that have lacking documentation. In their thesis, Fiske and Eklund for example points out that the system UU-plus has a lacking documentation and in particular when it comes to how the system should be used (Fiske and Eklund, 2013, pp. 54).

During the development of the prototype in this study it also became apparent that the system Selma had a lacking documentation. This was concerning one of the methods called “Search Instance” (“Sök tillfälle”) which returned an object called “TillfälleDTO” which was not present in the Object descriptions document (uppdok.uadm.uu.se, 2014). This object turned out to have as many as 46 fields which is difficult to understand and get a good overview of without an object description.
5.1.4 Lacking API in Selma

During the development of the prototype in this study there was also a problem in retrieving what is now called “course instances” (Kurstillfällen) at one specific case. The reason behind this was partly because of the API but also because of the documentation.

When you search and retrieve multiple course instances through the method “Search Instance” (SökTillfalle) you get an array of objects called “TillfalleDTO” back. When these objects has been retrieved it is useful for a developer to, on the basis of one of those objects, be able to send a request and retrieve this object with the help of one of it’s unique identifiers.

In order to retrieve a specific course instance the appropriate method to use called “GetCourseInstance” which instead required the three attributes Course code (Kurskod), Education language (Undervisningsspråk), and Academic year (Läär).

Because the object “TillfalleDTO” did not exist in the Objects description as mentioned earlier and because all of these attributes did not directly exist in the TillfalleDTO object (Academic year did not exist directly), this request has been made unnecessarily difficult. The API does not support the option of retrieving this with the only existing unique identifier “ktillid”.

Apart from this what is called “Course Code” in the “GetCourseInstance” method was in the “TillfalleDTO” object instead called “Code” which made it difficult to understand that these two were the same.

5.1.5 Missing IT systems

The Information infrastructure is missing an activity planning system which is also the topic of this study. The lack of an activity planning system is causing a lot of unnecessary manual work, increases the workload for many employees, and decreases the efficiency of the organization as a whole. It is therefore important that such a system is developed in a near future.

5.1.6 Non user-centered development

As mentioned in a previous study (Fiske and Eklund, 2013, pp. 51), there is an overall problem in the information infrastructure concerning the lack of user-centeredness. This is of course a very important issue to resolve because it affects the working conditions for the employees at the University.

5.1.7 Incapability of affecting systems

Through out the organization, there is a feeling of powerlessness in affecting and having an impact on the evolution of the ISs constituting the Information Infrastructure. This results in unsatisfactory systems and working tools for the University.

This is a political issue and is also a side effect of buying systems from external consultancies. Another factor is large number of departments who use the systems which makes it difficult for
the developers to satisfy each set of unique needs. This is also of course one of the unavoidable problems of having a nation wide information infrastructure. It requires a great pressure to affect a nation wide system like Ladok.

5.2 Problems in the Activity system

In this section we look at the problems that exists in the Activity system which constitutes practical implications of the lacking Information infrastructure that we have just covered. These problems are here categorized after who the implications affects.

5.2.1 Problems for the Directors of Studies

- A lot of manual work in Excel is carried out in the budgeting of courses and programmes which means an increased risk of human errors which are tedious and time consuming to identify and correct.

- The Director of Studies is using previous years budget as a template but there are still many pieces of information that needs to be updated each year because the availability of teachers and the number or course instances varies from year to year.

- Spreadsheets are developed independently by each Directors of Studies for their subject which makes it difficult for someone else to take over this work if for any reason the current Director of studies would become unavailable to the organization. This is therefore hindering effective knowledge management in the organization.

- There is no automatic support for interoperability between Directors of Studies and Main Teachers which makes communication between Directors of Studies and Main Teachers difficult and ineffective because they have to rely on communication via e-mail or in person.

- The course budgets lacks clarity because it is many times difficult to get a good overview working in Excel spreadsheets, especially when the budgeting involves many teachers and course instances.

- There is no automatic support for reporting teachers worked hours. Today, each teacher needs to keep track of their own hours and send that information to the main teacher who has to verify all teachers hours for that course instance and then forward them via e-mail to the Director of Studies. These e-mails are not standardized so they come in many different formats and it is therefore very tedious for the Director of Studies to handle all these e-mails.

5.2.2 Problems for Main Teachers

- Schedules are today developed manually by Main teachers, often in Word or Excel, which means that there is no automatic connection between scheduling and the room booking system TimeEdit. This increases their workload and makes the communication between teachers, administrators, and room booking personnel more difficult because they have to rely on communication via e-mail.
• Because the schedules are developed manually there is an increased risk of making human errors in the schedules. It is also common that unanticipated changes in the schedules needs to be done during the course implementation. This means that the teachers has to create a new version of the schedules which creates increasing difficulties in the communication of schedules to students.

• The manually developed schedules also gives a lacking overview, especially when there are many teachers in a course.

• There is no automatic support for interoperability between the Main teachers and the Director of studies which makes this communication difficult.

5.2.3 Problems for Teachers

• There is no automatic support for reporting of worked hours for the teachers on the University which results in extra work in the form of individual book keeping of hours and that emails have to be exchanged between the teacher and the Main teacher which can make this process unnecessarily time consuming. These hours also have to be sent via non standardized e-mail communication.
6 Results

In the previous section we learned about the problems existing at Uppsala University concerning both the Activity system and Information Infrastructure. In this section we will present a possible solution to many of these problems, in the form of a prototype of an activity planning system.

On the basis of this prototype business and technical requirements of both the proposed system and on the installed base that was gathered through the use of the proposed method are presented. New activity processes, and a description of the interaction between the planning system and other surrounding systems as well as an updated conceptual apparatus that comes as a consequence of the implementation of a system like this are also presented.

First off, the prototype will be demonstrated with the help of screenshots and descriptions of these. After that, the user requirements gathered from the demonstrations of the prototype are presented, followed by the interaction requirements and finally the updated concept apparatus is presented.

6.1 The prototype

We will here examine the developed prototype with it’s various features and the GUI (graphical user interface).

In order to get a better understanding of it’s usage, a certain predefined case will be used to illustrate how one would go about using the system in this particular case. The case has been designed to display the (most important) features of the prototype and also to be a realistic and typical example of what people in this organization would want to do in their everyday work.
The first thing that happens when you are entering the website is that you are greeted as a Guest and you have the opportunity to Login to the system. When you have logged in, you are then instead greeted with your name and your role in the system (Figure 10).

Important to note here is that the prototype is available in two languages, represented by the two flags in the top right corner. It is very important for the University to have their systems available in both Swedish and English because there are many individuals working within the University who are not fully confident with the Swedish language. Also, it happens that the University takes in guest lecturers who do not speak or understand Swedish at all.

When external persons are invited as guest lecturers it might also be important for them to be able to use this type of system.

As we can see, there is a menu in the top right corner under the “Log off” button consisting of the links: “Home”, “Plan”, “Course Instances”, and “Users”. We will from now on simply call this “The menu”. We will now press the “Plan” link in order to get started.
Now we have pressed the “Plan” link in the menu and are presented with a sub-menu consisting of three different planning areas that each represent a set of different functions in the system. This menu consists of the links “Search course offers”, “Show course instances”, and “Personnel budget”. We will from here on refer to this as “the Plan menu”.

“Search course offers” represents functions concerning the process of finding available course offers and on the basis of them creating a planning object for our system. We have chosen to call the planning object for a course implementation, “Course instance”.

“Show course instances”, represents all the functions related to the planning of our “course instances”, and “Personnel budget” represents the budget for each employee.

The first thing we want to do in the system is that we want to create our planning object. That is, the information object that represents the collection of activities that a course instance is constituted by and that we need to plan. In this case we want to plan the implementation of a certain course and our planning object therefore needs to be based on one or more course offers.

We press “Search course offers” in order to find the course offer(s) whose implementation we want to plan.
We are now presented with a search form that will help us finding the course offers we are interested in (Figure 12). Note that what we in this prototype call “Course offer” represents the information object that is right now called “Course instance” in Ladok. But here we have chosen to call it Course offer because we believe it is a more appropriate name (See definition 4.4).

On this screen we find four drop-down lists, a filter text field, and two buttons. The drop-down lists fills the function of filtering the search by different criteria. And the “Filter” text box is used after the search in order to further filter the search results by more specific criteria, for example the Name or Registration code for the course offer.

The first drop down list “Department”, contains all departments at Uppsala University. We chose “Department of Informatics and Media” because that is were the course we want to plan is given.

The next drop-down list “Subject” contains all the subjects that exists in the specific department we just chose. We here chose “All” because we know there aren’t overwhelmingly many courses going on during an academic year at this department.

In the two last drop-down lists, “Start” and “End”, we can filter our search by the period the course offer we are looking for is offered in. The course we want to plan is in the period between the fall semester and spring semester. We then hit “Search”.

Figure 12: Search course offer screen.
When we have made our search with the search filters we set up, we will get a result and the number of results will also be presented (Figure 13). Here we can see that we got 54 hits on our search.

If we want further information about a certain course instance we can press “Show” on the right hand side of the course offer and a pop-up will show course offer details.

In this case we want to plan the course ASP.NET and we can see that there are two course offers called ASP.NET on the same period in the search results. In order to be more efficient and make a better use of the University’s resources we want to teach these two course offers together in a single course implementation. That is, we have two different student groups who can apply for the course with two different registration codes because they are for example given as a part of two different programmes. Therefore we also need to plan them as a single course instance so we simply select the two offers and press “Create course instance” (Figure 13).
We are now presented with a screen where we can review our course instance. The information displayed are; “Name”, “NameEng” (English name), “Available places”, “Start week”, “Number of weeks”, “Period”, “Semester”, and “Start date”.

We can for example change the name of our planning object (course instance) if we wish to do so (Figure 14).

The available places, the text field that is highlighted in the figure 13, has to be manually entered in the prototype but this could in a real implementation be taken from the Ladok database where this information is kept. In this case, since we are combining two course offers into one course instance, this would be the addition between the available places on both course offers.

When we are satisfied we press Save to save our changes to the planning object. We can now go on to view our course instances in the “Course Instances” link in the menu.
We then get a list of all the course instances we have created (Figure 15), some of which were already in the system. On each course instance there is a menu consisting of “Request bookings”, “Activity plan”, “Manage”, and “Show info”. Right now we are interested in getting some more information about the ASP.NET course instance we just created so we press the “Show info” link on the “ASP.NET” course instance.

We can now see that this course instance is connected to two different course offers and that there are 100 available places, the start week, period, semester and the start date.

Now we are ready to start planning the course instance. The first step is to press the “Manage” link on the course instance we want to plan. We press “Manage on the ASP.NET course instance.
On this screen (Figure 16) we can on the left hand side, see and modify some of the course instance’s information, and on the right hand side we can “Add teachers” and “Create course instance parts”.

We have here already added a teacher but we will now go ahead and add another teacher so we press “Add teacher”.

Figure 16: Manage course instance screen.
Now, we get a list of all employees at the department of informatics and media, on the left side. In this list, a check box, the employees name and job title are presented. In order to add a certain teacher on the course instance we simply check the check box to the left of that employee in the list and it will pop over to the right side as shown in 17.

Here we also need to assign the teacher with a role and plan the number of hours the teacher should work on this course instance.

When all desired teachers have been added we can now press the “Add” button. The teacher will then be added to the list of teachers we saw in figure 16.
If we now go back to the manage course instance screen (Figure 16) we can choose to press “Manage” next to this newly added teacher. Then we come to a screen that looks like this (Figure 18). Here we can change the teacher’s role and number of hours and we can also make a report after the course is done with the actual number of hours that the teacher has worked. The teacher might have exceeded the number of planned hours or fallen below in which case this need to be reported to the Director of Studies.
If we instead go to the link “Create course instance parts” that we saw on figure 15 we end up on this screen (Figure 19). This screen lets us divide a course instance into different course instance parts. This is particularly useful for departments who like to create very large course instances that spans over a whole semester. In such a case it is useful to be able to divide this course instance so that the parts can be planned separately. In this particular case we have a course instance of 10 weeks which we want to divide into two 5 week course instance parts. We here press “Create” to do this.

Now that we have created and managed our ASP.NET course instance we can start planning the activities for this course instance. We do this by going to “Course Instances” in the menu and then on the menu on our course instance (Figure 15) we press “Activity plan”.

Figure 19: Create course instance parts screen.
Figure 20: Activity planning for course instance screen.

This is the activity planning screen (Figure 20). It consists of something that resembles a schedule that is automatically covering the course instance’s period. Here, all the activities of the course instance can be planned into dates and times. For each activity that is added you need to specify certain information which we will now go through.

“Date” is the date that the activity is planned on. “Time” is the time the activity is planned on. “Name” is the name of the activity. “Type” is specifying what type of activity that is being planned, this could for example be “Lecture” or “Lab”. “Group” is which student group this activity is for, here you can also select “All” if it concerns all students. “Length” is specifying for how long the activity will be going on. “Teachers” specifies which teachers should be teaching in the particular activity. “Content” specifies what the contents of the activity will be, this could for example be “Introduction”. “Number of places” is the planned number of students that will attend this course instance. Number of places is important when the system creates a booking request because the rooms are among other things planned according to how big student group needs to fit into the room.

While planning teachers on these activities it might also be useful for the main teacher who is in charge of this work to know how many hours each teachers are already planned on for this course instance. In this case the main teacher can press on the teachers initials and a pop-up will open (Figure 21).
In the image above we can see the pop-up that opens when pressing a teacher’s initial in the activity planning (to the right of the pop-up in figure 21).

It shows planned, reported, distributed, and remaining hours for this particular course instance and teacher, as well as some more general information about how many hours in total the teacher is planned on over the whole year.

This could be useful for the main teacher when planning activities so that a teacher’s hours are not exceeded to any greater extent.

When the activities are added in the activity plan we press “Create” at the bottom of the page to save our planned activities.
Figure 22: Requested booking in the activity plan (activity turns blue).

The planned and saved activities show up in the activity plan in blue (Figure 22), once a booking request is sent it will turn yellow and once it has been booked it turns green.

The next step is to create a booking request for all activities that needs a room. In the top left of the activity planning “schedule” there is a link with the text “Create booking request”.

Now we want to press this link to book our activity.
On this screen (Figure 23), we can select which of our planned activities we wish to create a booking request for. We can for example have planned activities which does not require the booking of a room, such as a deadline for the submission of a PM or a grading activity for the teachers.

When we have chosen which activities we want to request rooms for we press the button “Request Chosen Activities”. The idea is here that the booking request will be sent to the room booking system TimeEdit where the rooms will be booked and sent back to the system and be places in the schedule where they actually was successfully booked. The requested time and the booked time can differ since the room booking in TimeEdit is a coordination of both teacher and room resources.

In the prototype this integration with TimeEdit is not yet in place because of a system upgrade in TimeEdit which meant that they did not yet have the new testing environment set up but this integration should, according to the IT coordinator (P1) be possible.

Now we have gone through almost the whole process of planning a course instance but we have one more function namely the “Personnel budget”. We can find this in the Plan menu on the Plan screen (Figure 11).

When pressing this link, we are presented with the Personnel budget screen (Figure 24).
On the Personnel budget screen (Figure 24) we can see various information about how each teacher is planned for the current academic year.

The information displayed are enumerated and explained next.

“Name” is the name of the employee. “EstExtent” is the estimated total percentage of workload at the University. “EduProp” is the proportion of the total workload that is distributed to teaching in educations. “EduHours” is the number of hours that the teacher is spending on teaching. “ResearchProp” is the proportion of the total workload that is distributed on research activities. “ResearchHours” is the number of hours that the teacher is spending on research for the University. “PlanHours” is the number of hours the teacher is currently planned on. “RepHours” is the total number of hours the teacher has already reported this academic year. “Activity hours” is the total number of hours the teacher has been planned on specific activities in the system. “RemHours” is the total remaining hours that the teacher has to work this academic year. “EduYear” is the academic year which is expressed by the start- and end semester.

These pieces of information are connected to the systems database and is therefore always up to date so that the Director of Studies can monitor the personnel budgets during the course of an academic year and make sure that the teachers get a good distribution of hours.

If a teacher exceeds the personnel budget in any way, that teacher is highlighted in red to indicate that something is deviating from the budget. The purple text just under the filter text box in figure 24 explains that:

“Red fields indicate one of the following: 1: planned activity hours > planned hours, 2: reported hours > planned hours, or 3: remaining hours < 0”.

59
6.2 User requirements

During the demonstrations of the prototype, the participants was encouraged to give feedback, requirements and general ideas about the activity planning system. In this section these ideas and feedback are presented in the form of users requirements on this system. The categories are listed and the respective requirements presented next.

6.2.1 Course instances

- A possibility to auto-generate what we call “Course instances” (the planning object) would be good since about 70% of the time, the course instance is not a so called “Joint Study” (Samläsnings) that includes multiple registration codes (P4).
- The course code needs to be included in the course instance (P6).
- Important to be able to create more loosely held “projects” as an alternative to the course instance form. This could for example be used for research or a side project in the basic education (P7).

6.2.2 User roles

- There should be user roles to control who can do what in the system because different people have different responsibilities (P4,P8,P9).
- A personal view where a teacher can see what is expected of them for a certain year in terms of what course instances and what roles the teacher has and how many students each course instance has (P7).
- Logins should not be based on employer logins because this creates problems for guest lecturers (P6).

6.2.3 Budgeting

- It has to be possible to budget course instances. That is, both how many hours the course instance should have as a whole and also a rough plan of how many hours should be spent on specific activities such as labs, lectures, administration and so on (P4).
- The connection between activity plan and budget is important (P4).
- Good with the ability to put comments in the budget (P5).
- Good with the ability to make teacher budgets on a monthly basis because the level of employment can vary very much for example due to parental leave (P5).
- Ability to see a continuous budget between subject and departmental level when it comes to the purchase of equipment and so on so that these decisions can be better informed (P5).
- Information such as level of employment should not be public in the system since it could be sensitive (P5).
• The possibility to manually change planned number of students is important because it is a balance and strategic decision (P5).

• Good if the Directors of Studies can get notifications or subscribe to deviations if any of the course budgets are exceeded in the planning by the main teacher (P6).

• More exact personnel budgeting that is based on teachers individual salaries (P7).

6.2.4 Teacher planning

• Important to get a good overview of the course instances for a specific academic year (P5).

• Be able to juggle hours between teachers, when planning teachers on course instances, and instantly see the outcome (P5).

• Manageable overview even for departments with many teachers and course instances (P5).

• Good to be able to plan both on the basis of a teacher and on the basis of a course instance (P4).

• Teachers that are added in the planning system and their assigned role on the course instance should be the basis of the teacher list in Studentportalen (P6).

• Teachers should be seen as resources rather than university employees so that external people, effortlessly, can be added in the system as for example a guest lecturer (P6).

6.2.5 Course instance parts

• This might already exist in Selma and in that case it should not be in this system too (P4).

6.2.6 Reporting of hours

• It should be possible for the main teacher to send an hour report for the whole course instance (P5).

• The process of reporting hours should be made in two steps. First the teachers report their hours to the main teacher and the hours can be discussed internally in the course. Second the Main teacher sends the report for the whole course instance to the Director of Studies and then it can be discussed between them (P4,P8).

• The report should be based on the activity plan so that the Director of Studies can see what activities the teachers spent the reported hours on (P4,P9).

• Main teachers should be able to comment hour reports if they for example exceeded the budget (P5).

• It should be possible to report hours on a monthly basis, in addition to the end of the course instance, because teaching assistants who work on an hourly basis should get paid each month (P5).
• The system should have the ability to send reminders about submitting the hour report (P5).

• There should be confirmations from the system, for example the hour report for the course instance was accepted by the Director of Studies (P8).

• There has to be a transparency so that Teachers can see what hours the Main teacher has reported for various teachers (P6).

• Teachers should get a list from the system of the activities that the teacher has participated in so that this can be either confirmed or modified with a comment (P6).

6.2.7 Activity planning

• Activity planning should be the responsibility of Main teachers (P4).

• Important to be able to add administrative work and planning without having to put them on a specific time or book a room and that this also is connected to the hour report (P8,P9).

• It should be possible to plan activities on a higher level, for example the number of hours it takes to reply to emails over the whole course instance (P8).

• A teachers overall budget for the whole year should not be visible in this phase because it is not the main teachers responsibility to take into account a teachers other commitments. This has already been taken care of by the Director of Studies when planning the teacher on the course instance (P8).

• The activity planning should be connected to the hour reports so that it is possible to see what a teacher has spent their hours on (P8,P9,P5,P6).

• Transparency is important so that teachers can see what activities the main teacher has planned for different teachers (P6).

6.2.8 Groups

• If the course instance has different registration codes connected to it, it would be good if the system could create and keep track of the student groups that have registered on a specific registration code because sometimes their activities are slightly different.

6.2.9 Room booking / Scheduling

• Schedules should be publicly available for the students before they have started and before they have various permissions to access systems (P5).

• It has to be possible to show schedules in an easily readable format with a good overview, something that can be difficult in TimeEdit today (P5).

• It should be possible to put reading instructions in the schedule (P5).
• If the system TimeEdit is going to be used for schedules it should not require too much administration work for booking agents and other personnel, this should to a great extent be computerized (P8). This is also because responsibility should not be put on other people in creating a schedule.

• There should be a possibility to export subscriptions to schedules through the system (P8,P9).

• It should be possible to change bookings directly through the planning system, without involvement of administrators, like activity type and so called “null bookings” which means bookings that does not have a room (deadlines etc.) (P6).

6.2.10 Other

• Confirmations from the system must be built in because this means a much safer system than we have today (P8).

• It should be possible to add activities and examinations during the course instance (P8).

• Positive to be able to extract statistics about certain course instances which could be a help when developing the course report (P9).

• Good to be able to export everything (P9).

• Keyboard shortcuts are positive (P9).

• Color codes should not be used in case of color blindness (referring to activity plan) (P6).

6.2.11 General feedback

The overall feedback to the demonstrated prototype was very positive. All the participants were enthusiastic and saw a potential in this type of system which lead to many creative ideas. One functionality that existed in the prototype but was not discussed so much and did not receive any particular enthusiasm was the ability to create course instance parts. It was also mentioned that this function might already exist. This might mean that this function should be less prioritized as a requirement but it might also have been because it was not a relevant feature for the particular participants. This feature is probably mostly relevant for the Department of Government because they create very large semester long course instances and no one from this Department participated in the demonstrations.
6.3 Interaction requirements

In this section all the requirements that was gathered concerning the interoperability and interaction between the planning system and the rest of the Information infrastructure will be presented. These interactions will be described in different levels of specificity depending on how far the results of this study reached. The integration with TimeEdit has the highest level detailed because this became one of the focuses in the study. This integration is shown with a sequence diagram and more technical details are presented.

Next the systems that the activity planning system needs to interoperate with is presented followed by a description of the required interactions.

6.3.1 The integration with Selma

The system needs to get course plans from Selma. These course plans are as we saw in section 4.5 “A required document that describe and contain regulations for a course” (See 4.4). This is useful for Directors of Studies and Main teachers when planning and developing the course because the course plan describe the course goals, content, education forms and needs to be followed accordingly.

6.3.2 The integration with Ladok

It would be best if the information objects that represent the implementation of a course (the planning object), that we in the prototype referred to as “Course instance”, and which could include multiple “course offers” (Kurstillfällen), could be created already in Ladok (P2). This is for example convenient for Studentportalen because this is where they get them from today. This would remove many problems that exists in Studentportalen today with Joint Study (Samlåsning).

Ladok needs to provide the Activity planning system with these course instance objects so that they can be used to plan that course instance. However, it is doubtful that this will actually happen, due to the fact that “there is no one at Uppsala University who can control Ladok” (Demonstration with P2,P3). If this is not happening then some other solution needs to be figured out.

6.3.3 The integration with Studentportalen

It would be best if the Activity planning system could be responsible for the creation of groups and teacher lists on course instances (course implementation) because the planning of courses is made in an early stage of the process. In this case, Studentportalen need to be able to get groups and teachers with their specific teacher role from the Activity planning system. So in this case, Studentportalen would be a consumer of the Activity planning system.
6.3.4 The integration with budget system

The Activity planning system should be able to import personnel budgets from some system. This could for example be the planning system UU-Plus but it is unclear for how long this system will still be in use at the University. In this study we have not examined this system or the integration so there is unfortunately no more specific requirement for this integration here.

6.3.5 The integration with TimeEdit

During the course of this study, the author and the client had several meetings with P1 the IT coordinator at the Department of Engineering Sciences who has a good insight in what is going on with TimeEdit. We were from these meetings able to agree on certain requirements that would be necessary for a conveniently working Activity planning system and also some aspects of how this integration could be solved in practice.

First of all, the Activity planning system should be able to send booking requests on three different levels of detail. These are presented next.

1. **Activity level** - It should be possible to make a booking request of specific activities, you should be able to specify the booking for each activity session separately. This is the level that is illustrated in the Prototype where the number of hours, activity name, activity type, date, time, teachers and so on is specified for a specific room and a specific time for one specific group of students.

2. **Batch level** - It should be possible to make a booking request in batches, that is if we have a specific activity that should be held in 4 different sessions. In this case it should be sufficient to specify the length for each session, the number of sessions, time period, the activity name, content and so on which is then resolved later by booking professionals.

3. **Course instance level** - It should be possible to make a booking of a whole course instance. This would only require the specification of number of hours per type of activity and the length of the activities, and so on. This could for example be used to get a rough plan that the Main teacher can change certain things without any confirmation from TimeEdit such as activity name, teachers, groups and so on.

If these levels of requests would be possible, it would allow for sufficient flexibility without too much administrative work for anyone involved.

The information that was required and that should be possible to specify was also discussed in these meetings and the result is presented below

**Required information** - Request Id, Date(period), Length(hours), Activity type, Number of places, Course instance Id, Customer Id (Department code), Contact person.

**Optional information** - Room type, Facility, Programme group, Course instance group, Teachers, Activity name, Content, Internal comment.

Finally a sequence diagram is illustrating the interactions between the Main teacher, Activity planning system, and TimeEdit.
As we can see, the Main teacher first plans the activities for the course instance. The planned activities are displayed to the Main teacher through the Planning system. The Main teacher then submits the booking request. The Activity planning system then calls TimeEdit's method importOrders with login details and an XML string containing the order details (See Appendix B for XML structure). TimeEdit then returns an array of order IDs which is then stored in the Activity planning system. The Activity system can then display for the user which activities are booked. When the main teacher then accesses the activity plan, an automatic request is sent from the Planning system via the method exportOrders with all the orderNumbers for the requested bookings to check if they have been booked. If so, TimeEdit returns XML strings containing the successful bookings which is then parsed in the planning system and displayed in the Activity plan for the user.

### 6.4 Information infrastructure Requirements

Here, all the requirements that can be categorized as Information infrastructure requirements are presented. In the problem description we saw that there were some problems in the Information infrastructure, for example, isolated islands of systems, misguided conceptual apparatus, lacking system documentation and API's and Missing information systems.

In order to correct these problems Uppsala University needs to put some requirements on the information infrastructure. The requirements on the information infrastructure that resulted from this study are presented next.
6.4.1 Lacking system documentation

During the development of the prototype, the author could not find the object “TillfalleDTO”, which was returned from a method “SokTillfallen”, in the object description for Selma Web Service (uppdok.uadm.uu.se, 2014). A requirement is therefore that this is added in this Object description (uppdok.uadm.uu.se, 2014).

6.4.2 Lacking and misguided conceptual apparatus

This study has also pointed out that the term “Course instance” (“Kurstillfälle”) that is today used for the specification of a certain admission and/or registration options for a course (see 4.4), is not a very suitable term. Therefore, Ladok should consider the use of a different name for what they now call “course instance” (Kurstillfälle), because it is not semantically correct to use the Swedish word “tillfälle”, meaning something similar to: session, instance, round, moment, occasion, for representing the specification of an admission. This is also causing further problems when applying this term to the specification of a course package admission. It then becomes “course package instance” (“kurspaketstillfälle”) which makes this mistake even more apparent.

6.4.3 Isolated islands of systems

Uppsala University’s Room booking system, TimeEdit, is today an isolated island which means that the integration of this system into the information infrastructure has failed. This is partly because of the lack of a planning system that makes this integration more natural. This system needs to be integrated into the information infrastructure through the interaction with an activity planning system.

6.4.4 Inaccessible testing environments

During the course of the study, repeated efforts were made to get access to a testing environment for Ladok’s web service but without luck. Information infrastructures evolve and in order for Uppsala University to be able to expand and maintain a functioning and useful information infrastructure, it is important that new systems can be built. This in turn requires the ability to test different solutions and interactions between systems. Therefore there should be available and accessible testing environments for systems in the information infrastructure.

6.4.5 Missing system

The information infrastructure at Uppsala University is missing a activity planning system and the most important and critical requirement of this thesis and for Uppsala University is the development of this system.
6.5 Consequences of implementation

This section attempts to show how this proposed system would affect various parts of the organization. This is here done by means models and descriptions. The consequences for different areas will be presented starting with the Activity system, followed by the conceptual apparatus and finally for the information infrastructure.

6.5.1 Activity system

The section showing how the activity systems looks today shows the manual nature of how work is carried out in the organization today.

The planning system greatly changes this way of working. The processes that this system would have the greatest affect on are Budgeting, Personnel planning, Room booking, Creating schedules, and Reporting of hours.

In general this system would eliminate a great deal of the manual communication of important information which will greatly improve the safety and efficiency of these processes. It also would eliminate much of the administrative work, especially concerning the booking of rooms which is today in many ways manual and demanding for administrators.

This system would make monitoring of course instances, follow up and course reports easier. This would effectively mean a more streamlined, efficient and safe activity system.

6.5.2 Conceptual apparatus

The implementation of this system would affect the conceptual apparatus because with it, new concepts must also be established. We also saw a problem with the conceptual apparatus in use today concerning the term “Course instance” (Kurstillfälle) which has a misleading name (see section 5.1.2). If possible this concept should be replaced with the term “Course offer” (Kurserbjudande) instead.

The concept “Course instance” should instead refer to the actual activities of the implementation of a course and be the basis of the planning of these activities. This is how the term i used in the prototype.

6.5.3 Information infrastructure

This system would if implemented, become a new part of the existing information infrastructure that we saw in section 4.2 (See figure 1). A result of this study has been to figure out where this puzzle piece fits. The figure below illustrates how this could be arranged.
Figure 26: The Activity planning system as a part of the existing information infrastructure (Edited, original image: 1).

This information model is showing an ideal picture of how this information infrastructure ought to be. This model assumes that Ladok will be creating the planning objects that we call “Course instances”. This illustration is making use of the terminology used in the prototype so what is now called “Course instance” in Ladok is here instead called “Course offer”.
7 Discussion

This study has explored the possibilities of using Prototyping as a means of solving complex information infrastructural problems, and is constituting a proposal for a possible method that can be used in similar problem situations. This method specifically concerns system integration in information infrastructures and involves prototyping for multiple purposes.

These purposes are firstly, to unravel detail planning of the integration of a system. This detail planning can for example be how to interoperate with various systems in the existing “installed base”.

Secondly, prototyping is used to identify problems that needs to be sorted out before the implementation of the system begins. Probably the best way of finding unforeseen problems that might occur during the development and integration of a system into an information infrastructure is to actually start building something. This forces developers as well as decision makers and other involved people to start thinking about practical details and it is better for problems to occur while still in the prototyping phase.

Thirdly, the resulting prototype is used in it’s traditional way to have something to show the users, because as we found in the literature review, this is practically the only way to give the users an understanding of the system and only then do they have a real opportunity to give valuable feedback and impose requirements. This can also make the users understand and see the potential of the system leading to enthusiasm and creative ideas that can be used in the system. This effect was achieved in this study but it can of course also have the exact opposite effect. But even that is useful because as we saw in the literature review, the first of Hanseth and Lyytinen’s guidelines for the development of Information infrastructures, was “Design initially for usefulness” (see section 2.3.4). If the users does not find particular function useful then it should probably not be there and there is no point in wasting time on it.

Doing these three things, means that developers can be better prepared when the actual implementation of the system starts. It means that more requirements will be gathered before the implementation and thus decreasing the risk of late delivery. It seams reasonable that this could lead to fewer encounters of problems, leading to an overall smoother development of the actual system.

The idea of the prototype is that’s only purpose is to elicit requirements and like throwaway prototyping it is not a prototype that is meant to become the actual system.

This might seem like old news but the fact is that there is currently not much knowledge about how to design information infrastructures today (Hanseth and Lyytinen, 2004). And this is a practice which this study illustrates, can be extremely complex.

This study is a good example of how many requirements an approach like this can generate and how much more clarity it can bring, both in terms of the design but also the political circumstances that are affecting the design. It also gives clarity when it comes to the evaluation of current processes and their quality compared to alternative redesigned processes.
8 Conclusions

This section aims to summarize the study and to provide an evaluation of the research results through an examination of both the results and the research approach. This will lead to a better idea of the usefulness, limitations and generalizability of the research. This section will also connect back to the original purpose and goals of the research, to what extent, and how, they were reached.

First of all, this study is constituting a Design and Creation research (aka “Design Science”). The criteria for Design Science Research are; the research involves the creation of some kind of new IT artefact, it shows not only technical skills but academic qualities such as analysis, explanation, argument, justification and critical evaluation, and most importantly it offers a contribution to knowledge (Oates, 2006, pp. 108-109).

This study consisted of the creation of two different artefacts. These were a method and an instantiation. The instantiation was in the form of a working prototype, and the method was one to guide the development and integration of a system that needs to relate to an existing information infrastructure.

The reason for developing this method was that there is a dearth of knowledge in the design of information infrastructures (Hanseth and Lyytinen, 2004, p. 208), and none of the examined existing methods is specifically providing any practical guidance to how the process of requirements elicitation and detail planning can actually commence. Therefore this study has been an attempt to build upon the current knowledge in this domain.

This study provided an insight in the working practices in the specific research case at Uppsala University and identified an array of problems concerning the activity system as well as the information infrastructure. This also illustrated the need for changed working processes and in particular a need for a better support for the employees when it comes to the planning and administration of the delivery of the organization’s services (course implementation).

This information along with continuous feedback and insights from the client, which is this case the supervisor of the study, as well as with the IT-coordinator constituted the basis of the development of the prototype.

The development of the prototype was a process that in itself functioned as a way to elucidate requirements and identify problems. Through the development of the prototype the study identified several problems including lacking documentation, lacking system API’s, and inaccessible testing environments. Also the prototype helped to unravel certain technical details, such as the possibilities for interaction with the room booking system TimeEdit. In addition, the prototype was used to demonstrate how problems could be solved and to elicit user-requirements. This generated important requirements, creative ideas and an overall better understanding of the design of the activity planning system.

The study also gave a clarity about the problems existing in the organization today. These problems includes the problem with inappropriate concepts such as “course instance” (Kurstillfälle) that turned out to be a source of some problems, which led to the realization that there is no information object representing the implementation of a course. This is valuable information because some of these problems might have to be solved before it is even possible to implement a
system like this. This new information object needs to be established and the question is where it should be created which is now something the organization needs to look into.

The purpose of the study has been to explore the possibilities of using prototyping as a means of elucidating and to elicit requirements of both the to-be developed system and the installed base that this system needs to relate to in order to create successful integration and interoperability of the new system in its environment.

This thesis is constituting a “proof of concept” because it shows how a method like this can be applied to a real world situation and result in an array of requirements and an overall improved understanding of how a system can fit into an existing information infrastructure.

The methodology of the study was a mixed research approach consisting of Design science and a single case study approach. It made use of document analysis and interviews that took the form of demonstrations and meetings. These analysis processes were discussed and justified. However, this study was not using any formal systems development methodology so a proper description of this process was not given. This makes the research less transparent.

Generally when it comes to case studies, and in particular single case studies, cautiousness in the generalizability of the study has to be taken. However there are elements of this study which can be generalized to other situations. For example, this study gave a rich insight of how problems and requirements can be gathered through the presented approach. As mentioned in the introduction, the problems presented in this thesis also exists at University Dalarna which is another Swedish University. Since the systems on a national level are connected to all Swedish Universities information infrastructures, it is also likely that similar problems exists in even more Universities across Sweden. And this activity planning system might be useful for many other Universities in Sweden to illustrate how such a system can help the organization to become more effective. Therefore there this knowledge can most likely be of general use for most Universities in Sweden.

The implications of this study is that there is now a proposed method for the integration of a system into an existing information infrastructure that is able to give guidance as to how similar problem situations can be approached and which can help bring clarity and generate requirements as well as creative ideas. This lays a foundation for further research and development of ideas in this specific area.

8.1 Limitations

The participants of this study that was used to gather user requirements was mostly from the Department of Informatics and Media. This was due to convenience and this means that the collection of user requirements are biased towards the work at this department.

This was a particularly complex and internal case, which might be difficult to understand for people external to Uppsala University. Therefore there might be a barrier in fully grasping the problems presented in this study for people outside the company.

Since this organization operates to a great extent in Swedish, most of the important terms in the organizations concept apparatus is originally in Swedish. This can sometimes cause difficulties in the translation of Swedish terms a fair way and at the same time explain the problems with
those terms.

Another limitation with this study was that it did not make use of any formal systems development methodology but only stated that it was an agile approach. This leads to a limitation in terms of the transparency of the study.

A possible limitation, that needs to be considered, with prototyping and in particular throwaway prototyping might be the extra overhead and effort it requires to develop something that is not going to be used other than for gathering requirements. This is especially the case when the prototype is in the vertical dimension (high fidelity) where actual functionality is developed which usually takes considerably more time and effort than making low fidelity drawings which is usually the case when dealing with throwaway prototyping. However, as was mentioned earlier, this approach is often the only good choice when there is a very poor understanding of the requirements to begin with.

Because this was a single-case study, only certain elements of this study can be generalized, any further generalization requires more research.

8.2 Further research

We saw in this study that the area of information infrastructure design is currently in need of new methods and guidelines. This study explored one approach of how the requirements elicitation process of the development of a new system that needs to fit into an existing information infrastructure can practically be carried out in order to gather interoperability requirements. The issue with the elicitation of interoperability requirements highlighted in this study needs to get more attention and more methods needs to be explored in order to provide better guidance in these types of situations which are becoming more and more important.

The proposed method in that this thesis presents, needs to be further explored, evaluated and tested in other cases and the results be compared to the ones in this study.
References


UHR.se. Welcome to the swedish council for higher education, May 2014. URL http://www.uhr.se/sv/Information-in-English/.


A  Activity Planning Prototype ERD (without user tables)
B TimeEdit Actual Booking Request XML Structure

Listing 1: TimeEdit OrderXML

```xml
<tns:orders>
  <order>
    <extid/>
    <newextid/>
    <organizations>
      <organization/>
    </organizations>
    <customerobject>
      <extid/>
      <type/>
      <fields>
        <field>
          <extid/>
          <value/>
        </field>
      </fields>
    </customerobject>
    <ownerobject>
      <extid/>
      <type/>
      <fields>
        <field>
          <extid/>
          <value/>
        </field>
      </fields>
    </ownerobject>
    <modifiedby>
      <loginnamename/>
      <authserver/>
      <modified/>
      <createdby/>
      <created/>
      <info/>
      <title/>
      <description/>
      <status/>
      <begin/>
      <end/>
      <requiredobject>
        <extid/>
        <type/>
        <fields>
          <field>
            <extid/>
            <value/>
          </field>
        </fields>
      </requiredobject>
      <rows>
        <extid/>
        <ordertitle/>
        <orderbegin/>
      </rows>
  </order>
</tns:orders>
```
<orderend/>
</orderrequiredobject>
<object/>
</orderrequiredobject>
<description/>
</status>
<object/>
</active>
<possible/>
</ordercount>
</orderlength>
</deliveredcount>
</deliveredlength>
<objects>
<object>
<extid/>
</type>
<fields>
<field>
<extid/>
</value>
</field>
</fields>
<objects>
</objects>
</extraobjects>
<object>
<extid/>
</type>
<fields>
<field>
<extid/>
</value>
</field>
</fields>
<objects>
</extraobjects>
</rows>
</order>
</tns:orders>