Prototyping by Proxy

Master Thesis, Software Engineering
Aalborg University

Research Unit:
Information Systems

Authors:
Tommy Holm Jakobsen
Mikkel Normann Follin

Title:
Prototyping by Proxy

Project period:
Software Engineering
SW10, Spring 2012

Project group:
sw106f12

Participants:
Tommy Holm Jakobsen
Mikkel Normann Follin

Supervisor:
Ivan Aaen

Synopsis:

Robotically-assisted surgeries is a relatively new way of performing surgeries at the Department of Urology at Aalborg Hospital. While robotically assisted surgeries give better quality surgeries, they take longer time to conduct than traditional surgeries. Therefore the department would like to speed up the surgeries to diminish this issue. To help them attain this goal, we developed prototypes for collecting data from surgeries to be used in analyses in regards to time efficiency, quality of the surgery and for educational purposes. We developed these prototypes using a highly innovative software process based on the Essence framework. Through this case we have researched how prototyping supports the Essence framework by enhancing the way in which we, as designers, organically and evolutionarily learn, discover, generate and refine designs through the use of prototypes.

We introduce the concepts of a *Surrogate Challenger* and *Surrogate Users* as proxy users to support the development process when the customer is not able to supply neither an on-site customer nor tests subjects. The Surrogate Challenger act as on-site customer who is able to participate on-site in daily discussions and evaluations when the real on-site customer is unavailable.

The Surrogate Users acts as test subjects who are able to participate in frequents and simple tests of developed prototypes. To somewhat minimize the gap of knowledge between the Surrogate Users and real users, we introduce the concept of *mapping* problems of one domain to another. We show how we are able to map a problem of the surgeon during surgeries to a problem solvable by the Surrogate Users during tests.

This master thesis describes our process, research, and the ideas and prototypes developed and demonstrated to the Department of Urology.

Number of printed copies: 4

Number of pages: 142

Date of completion: June 15th 2012

Preface

This master thesis is made by group sw106f12 on 10th semester, Information Systems (IS), Department of Computer Science, Aalborg Unviersity.

We wish to thank the Department of Urology at Aalborg Hospital for entering a partnership, providing us with an interesting case and letting us follow their daily work during robotically-assisted surgeries.

More specifically, we wish to thank Mette Gjerløv, Special Consultant, "Sundhed - Plan og Kvalitet" (Health - Plan and Quality), Region Nordjylland, for establishing the partnership. Johan Poulsen, Chief Specialist Surgeon, FEBU (Fellow of the European Board of Urology), Consultant Urologist to King's College Hospital London and Aalborg University Hospital Denmark, and a MIUC consultant (Minimal Invasivt Udviklingscenter), for participating in meetings and sharing his knowledge. Jane Petersson, surgical nurse with speciality in robotically-assisted surgeries, for showing us around and sharing her knowledge. Furthermore we wish to thank Staff Specialist Grazvydaz Tuckus, Chief Surgeon Knud Fabrin and surgical nurse with speciality in robotically-assisted surgeries Lotte Juul Hansen. Finally, we wish to thank them all for showing great interest in our work, being helpful, and always answering our questions.

We would also like to thank Ivan Aaen, Associate Professor, Department of Computer Science, Aalborg University, for his highly appreciated supervision, for showing great interest in our project, and for sharing his knowledge about about Essence. We appreciate his dedication through his role as a *Surrogate Challenger*, for always being interested in participating in discussions, and for sharing his valuable ideas.

Finally, we would like to thank Morten Højfeldt Rasmussen, Ph.d. student, Department of Electronic Systems, Aalborg University, for his assistance with configuring and optimizing the speech recognition framework *CMU Sphinx - Open Source Toolkit For Speech Recognition*, and for sharing his and Morten Svendsen's Danish Language Model for CMU Sphinx.

Contents

1

Introduction

Information Systems are an important part of the society we live in, and the software underpins most of the significant technological advances in modern societies. However, researching is mainly focused on efficient development through method and engineering techniques, and less focused on being creative and innovative in the development, design and exploitation of information systems. This is known as Software Innovation, and we believe that this lack of focus in current research is a problem that deserves more attention.

Software innovation has become more important than ever for different reasons. Software is found everywhere and today most of us cannot work or communicate without it. This large scale of software development means that it has become mass-produced, and the routine kind of development can often be outsourced. Thus in order for software companies in highly developed countries who employs expensive, but highly-educated engineers, need to think carefully about their market position. This problem will only increase in the future, so in order to be able to compete in the market it is necessary to focus on development forms with higher value addition, such as software innovation.

Looking from the designers or developers (not necessarily two different persons) perspective is the desire for being challenged in order to flourish, develop skills and learn new techniques and technologies. It is important to have both creative freedom, and space and time to express that creativity. Developing innovative products is exciting and inspiring for the developer compared to routine tasks, but also very satisfying for both customers and users.

Usually, software development processes focuses mostly on efficiency in order to meet requirements and deadlines, be they traditional, agile or in-between [Aaen, 2012]. While we might have a known solution to a problem, we do not know if this is the best possible solution. In order to find a better solution, we need to be creative and innovative—think out of the box.

But how can we do that, and what techniques can we exploit to create a more creative process? Ivan Aaen is in the process of creating a development framework called Essence. A framework that supports creativity and innovation at the level of the software developing teams, and support teams in producing valuable solutions throughout a systems development project. Main ingredients in the strategy are incremental development, testing increments against real world challenges, and learning from collaboration and experimentation [Aaen, 2012].

Acknowledging the importance of software innovation, we have researched how we can further enhance creativity through the correct use of prototyping and incorporate it into the Essence framework. The idea of exploiting prototypes to support creativity is inspired by other industries such as the automobile or architecture industry, where prototypes are very common in the design process. For example, the automobile industry uses a full-size clay model before building a new car, and architects use 2D models to help determine the spatial relationship of the rooms.

While exploring a physical prototype (and thereby the problem), we say that we traverse the design space, in order to explore all possible design alternatives and rationales. Donals Schön describes this as *reflective conversations*, because it forces the designer to see the product; it "speaks back to him" and new ideas might come up that can be shared and evaluated with the team. Through this process, different ideas might come up, and these might be able to solve the problem in a better way than what was originally thought about.

The research was conducted while working with the Department of Urology at Aalborg Hospital. This partnership was established during our previous semester because they had a problem that they were not sure how to solve. The department had recently introduced the *da Vinci®* Surgical System to assist prosthetic removal surgeries. A robotic surgery system designed to facilitate complex surgeries using a minimally invasive approach. While greatly improving e.g. the patients' time to recover after such a surgery, the department had a problem of not being as efficient as traditional surgeries. Using the *da Vinci®* Surgical System they could operate two, maybe three patients on a day, but using traditional surgeries, it is possible to operate up to five patients on a day.

Meetings were held to discuss possibilities and features of some system that could help them improve their overall efficiency. We arrived at a solution that involved *marking up* each step in a prosthetic surgery removal process. Marking up when a steps starts and ends, makes it possible to create statistics showing what steps are most time intensive, as well as storing video feeds for each step in order to analyse the process even further. If it was possible to create a system able to do this without intervening the surgeries, chief surgeon, Johan Poulsen, saw many and great possibilities, that could really help improve the surgeries, both in terms of efficiency, but also quality and education.

During our collaboration with the Department of Urology, we discovered that they were not able to engage in meetings as often as we would have liked to. Ideally we would have an on-site customer available during prototype development, but the department did not have the resources to supply one. Not having an on-site customer available on the team is a common problem [Inayat et al., 2012] and thus we decided to research if it was possible to replace this role with what we call a *Surrogate Challenger*. A Surrogate Challenger is inspired by the *Challenger* role in Essence, which is the equivalent of an on-site customer. The Surrogate Challenger however, does not necessarily have the experience and knowledge of the Challenger, but he is able to support the project by "taking on the glasses" of the Challenger.

It is not only the on-site customer the Department of Urology was not able to supply. While developing the prototypes, it was necessary to test them, to see if they could function during surgeries without removing the surgical staffs focus from the surgery and especially the patient. Obviously, it is not possible to test a prototype during a real surgery, and setting up a similar testing environment

is expensive, both in terms of equipment required and salaries to the surgical staff.

Instead of ignoring the tests, we introduced the concept of *Surrogate Users* in order to be able to test the prototypes frequently and receive feedback, without requiring an operating theatre or surgical staff. Acknowledging that we could not perform "correct" tests (compared to real tests with the surgical staff during real surgeries), we researched how we could design the tests in order to get as close as possible to a real test.

By performing these frequent tests and exploiting the feedback to improve the prototypes, we believed that we were able to create a more complete prototype that could be demonstrated and tested with the Department of Urology. The Department of Urology could then use this prototype as a basis for decision for the department to help them decide whether or not they want to further develop the suggested solution, or we could use it to evaluate if there is a foundation for establishing a company to further develop the system.

1.1 Problem Statement

The aim of this thesis is to investigate how prototyping can be utilized in an innovation context. Prototyping enhances idea generation [Lim et al., 2008, Rudd et al., 1996] and by incorporating them into innovation processes such as Essence, it should be possible to enhance the possibilities of developing more innovative ideas on the team. Ideas can then be manifested into prototypes, tested and explored, with the ultimate goal of developing more novel solutions. Based on this, we form the first research question as:

> *RQ1: How can prototyping support the Essence innovation process with the goal of developing more innovative solutions?*

[Jakobsen and Follin, 2011] describes how certain users are not able to engage in the development process due to time constraints. This engagement could have been users acting as on-site customers or prototype testers, giving feedback and thereby generating, maturing or discarding ideas as new ones arise. Trying to find a way of simulating this interaction between users and developers, we will be introducing the concept of *Surrogate Users* to make it possible even when the actual users are unavailable.

> *RQ2: Is it possible to use a Surrogate Challenger and Surrogate Users instead of the actual on-site customer and users when they are unable to engage in the development process? If yes, in which situations and with what limitations?*

Surrogate Users most likely do not have the same capabilities of the actual users and will therefore have limitations when it comes to testing the prototypes. There will be a greater risk of giving incorrect results compared to what an actual user might get. To minimize this difference, we will investigate the possibilities of *mapping* the problems from the actual users domain to problems that the Surrogate Users are more capable of handling.

> *RQ3: Is it possible to map problems from one domain to another, still resolving the problems in the original domain through the use of Surrogate Users?*

In order to achieve this mapping, we will examine the domain of the actual users and propose solutions that are similar but easier for the Surrogate Users to handle. To test whether this mapping is adequate, we will be testing the final prototype on the actual users, validating whether the prototypes developed through the use of Surrogate Users results in a prototype that is useful for the actual users.

2

Case Study: The Department of Urology, Aalborg Hospital

This chapter describes our case study at the Department of Urology, Aalborg Hospital. It is partly taken from the document we wrote during our previous semester [Jakobsen and Follin, 2011], but with minor modifications.

A short description of Aalborg Hospital and the Department of Urology will be given, including their use of the robotically-assisted surgical (RAS) system, the *da Vinci*® Surgical System. Furthermore, a typical prostate removal surgery will be described in detail, including the operating room and staff involved.

Through Mette Gjerløv, Special Consultant, "Sundhed - Plan og Kvalitet" (Health - Plan and Quality), Region Nordjylland, we established a working relationship with the Department of Urology at Aalborg Hospital. The Department of Urology is far ahead in the use of RAS and they see a great future in it. Therefore they are in the process of establishing an educational course for teaching new and existing surgeons how to perform robotically-assisted surgeries. This development within the department, made them interested in a working relationship with us, hoping that we had some interesting ideas on how to improve their existing techniques within this new area of work.

Mette Gjerløv helped us establish contact to chief specialist surgeon Johan Poulsen, FEBU[1], Consultant Urologist to King's College Hospital London and Aalborg University Hospital Denmark, and a MICU consultant (Minimal Invasivt Udviklingscenter). As a second contact, we had staff specialist Grazvydas Tuckus, and when we visited the operating room, surgical nurse with speciality in robotically-assisted surgeries Jane Petersson was our supervisor. For the second meeting, in addition to Jane Petersson and Johan Poulsen, chief surgeon Knud Fabrin and surgical nurse with speciality in robotic surgery Lotte Juul Hansen were also present.

Aalborg Hospital, Aarhus University Hospital is the largest hospital in Northern Jutland, Denmark, and employs approximately 6,500 and attend highly specialized regional functions for approximately 640,000 inhabitants [Sygehus, e]. As the region's largest hospital, it plays a key role in the cooperative North Denmark healthcare system, but it also stands out both domestically and in-

[1] Fellow of the European Board of Urology, EU academic degree

ternationally, which is, among others, seen by its use of robotic surgery in the Department of Urology [Sygehus, c, Medicin,].

2.1 The Department of Urology

The Department of Urology at Aalborg Hospital provides clinical services for diseases of the male and female urinary tract and the male reproductive organs. Patients have access to various treatments, for example

- Treatment of urologic tumors in the prostate, bladder, kidney, etc., and among this larger surgical procedures whether replacement of the bladder or removal of the prostate
- Urinary problems in consequence of enlarged prostate
- Male genital disease and erectile dysfunction
- Advanced laparoscopic surgery

Aalborg Hospital has chosen to utilize robotically-assisted surgery, which the region sees lots of potential in. It is however an expensive investment, so the region expects high capacity and return in terms of patients health and recovery. This should be possible, given that robotically-assisted surgery allows for more precision, miniaturization, smaller incisions, decreased blood loss, less pain and quicker healing time.

Aalborg Hospital bought their first robot for robotically-assisted surgery in the beginning of year 2008, and the first operation took place October the 6^{th} 2008. In the first year of use, they operated 110 patients (56 for endometrial cancer and 54 for prostate cancer) [Sygehus, a]. In 2010, the Department of Urology operated 2,310 patients [Sygehus, d], and 75% of these were operated using robotically-assisted surgery. The success of using the robot has led to an increase in the demand of patients wanted to be operated using this form of surgery. The high demand have made the region allocate another 16 million Danish kroner in 2012 for purchasing a second robot, and a third used robot for training purpose [Sygehus, b].

2.1.1 The *da Vinci*® Surgical System

The robotically-assisted surgery system used at Aalborg Hospital is the *da Vinci*® Surgical System, by Intuitive Surgical Inc.[2] A picture of the surgical system can be seen on Figure 2.1, and a video of the system in use can be seen here [3].

da Vinci overcomes the limitations of traditional open surgery and minimally invasive surgery. Small incisions are used to "dock" the arms of robot on the patient, which introduce a 3D camera and different instruments. The surgeon are seated at a console, viewing the 3D video feed of the surgical site while controlling it and the other docked instruments (both can be seen on Figure 2.1). Computer technologies scale, filter and seamlessly translate the

[2] *da Vinci*® Surgical System homepage `http://www.davincisurgery.com/`
[3] *da Vinci*® Surgical System video: `http://www.viddler.com/player/6d715836/`

Figure 2.1: *da Vinci*® Surgical System [Inc.,]

surgeon's hand movements into precise micro-movements of the instruments. Nothing is programmed and the robot cannot make decisions on its own.

The *da Vinci*® Surgical System has multiple inputs and outputs for video and audio feeds. Multiple monitors and microphones can be attached to the system for assisting the staff or other uses, allowing them to follow the surgeon's actions and assist him in doing so. Other video output can be attached to the systems input channels, allowing for multiple video feeds to reach the console and monitors. The surgeon can then cycle between different views of the attached inputs, allowing him to see a picture-in-picture view of for example the video feed from the surgical site and ultrasonographic images of the patient.

2.1.2 Prostate Removal Surgery Using the *da Vinci*® Surgical System

Prostate cancer or enlarged prostate can make it necessary to remove the prostate, a gland in the male reproductive system. If the prostate is not removed, cancer cells may metastasize to other parts of the body, particularly the bones and lymph nodes. In older men, the part of the prostate around the urethra often keeps growing, making the tissue press on the urethra, leading to problems passing urine. This section describes the prostate removal surgery and the surroundings around this procedure.

2.1.2.1 Procedure

The prostate removal surgery procedure was described to us by the surgeon Grazvydas Tuckus. It is composed of the 17 steps described below and the procedure usually follows that order. The sequence of steps may vary, depending on who the surgeon is.

1. The patient is anaesthetized and catheter is inserted
2. Manual placement of the ports where the robot is to be docked
3. Robot docking and mounting of the instruments

4. Exposure of the seminal vesicle
5. Exposure of the prostates back
6. Bladder dismounting
7. Exposure of the prostates sides
8. Opening of the bladder outlet
9. Removal of the blood flow to the prostate
10. Judgment: To maintain or not to maintain the nerves
11. Prostate exposure from the urethra
12. Place the prostate in a bag
13. Stitch together the bladder outlet and urethra
14. Judgment: Removal of the lymph nodes
15. Undocking of the robot and removal of the ports
16. Removal of the prostate and catheter
17. The port openings are stitched together

2.1.2.2 Surgical Personnel

During the prostate removal surgery, there is always at minimum four nurses and one surgeon in the operating room. In the beginning of the surgery, the surgeon is responsible for port placement, and when the ports are placed, he takes place at the *da Vinci* console, operating the robot. When the prostate has been removed, the surgeon removes the ports and proceeds to stitch up the patient. The four nurses has the following areas of responsibility divided among them:

- One is a nurse anesthetist, responsible for administering anesthesia and keeping constant watch on the vital signs of the patient in surgery.
- Another one is an assisting nurse, responsible for helping the sterile nurses do unsterile work, such as collecting and unpacking new equipment.
- The final two nurses are both sterile:
  - One is the surgical assistant to the surgeon, responsible for assisting the surgeon inside the patient by keeping the area being operated on tidy, e.g. by removing blood and fluids using a suction device.
  - The second sterile nurse is responsible for preparing the instruments to be used on the robot and helping the surgical assistant with what ever she may need.

Figure 2.2: Operation room overview during port placement

Figure 2.3: Operation room overview after docking

2.1.2.3 Operating Room

The operating room has a size of approximately 50–60m^2, and contains all the necessary equipment for performing a prostate removal surgery using the *da Vinci*® Surgical System.

The surgeon is seated at the console in the back of the room, and the sterile surgical assistant is standing next to the patient, assisting the surgeon and the robot. The second sterile nurse is standing on the opposite side of the patient, and the assisting nurse walks around in the room. The nurse anesthetist is seated in front of the patient surrounded by monitors helping her monitoring the vital signs of the patient.

An overview of the operating room before and after the robot is docked, can be seen on Figure 2.2 and Figure 2.3. Figure 2.4 contains a floor plan of the operating room.

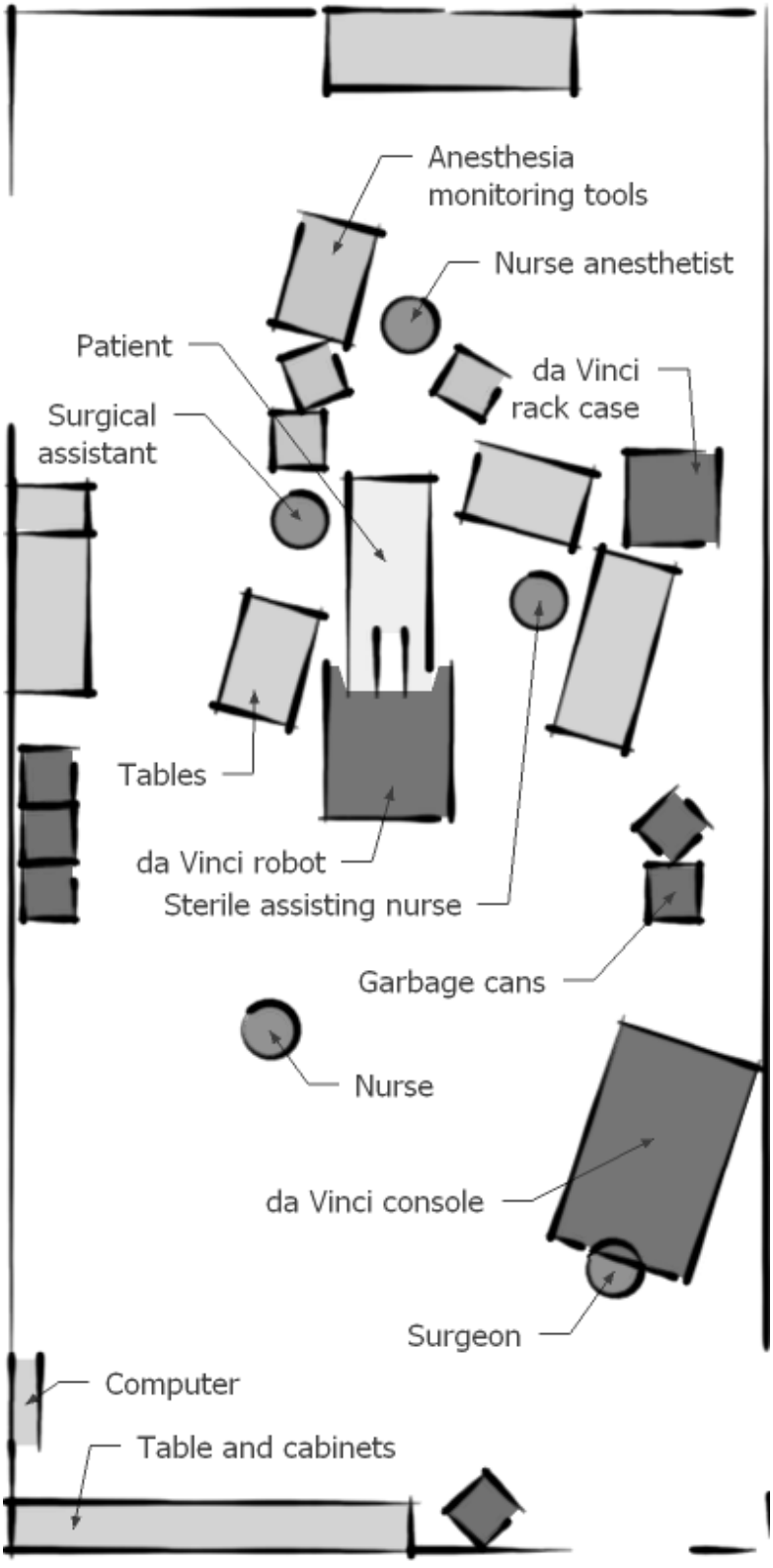


Figure 2.4: Sketch of the operating room floor

3

Previous Work

During the previous semester we conducted a literature study for use as foundation for our research in this semester and we developed prototypes for presenting ideas to our collaborator and "customer", the Department of Urology. During the previous semester we had several meetings with the Department of Urology. A timeline of these meetings and our research is shown on Figure 3.1.

We worked together with the Department of Urology in order to create a tool for collecting and analysing surgical data from robotically-assisted surgeries using the *da Vinci*® Surgical System. We quickly found that time is an issue in this project as the surgical staff of this department have very sparse time for meetings and communication with us, which required us to make the most of their available time. Therefore we studied several areas such as the concept of lead users, knowledge sharing, and prototyping along with literature on software innovation and the Essence framework.

3.1 Litterature Study

We studied the concept of lead users because we consider the Department of Urology to be lead users in the area of robotically-assisted surgeries. Lead users is a concept described by Hippel and Lettl in [von Hippel, 1986, Lettl et al., 2006, Lettl, 2007, Lettl and Hienerth, 2008], with the main properties being:

- They reap higher benefits, compared to regular users, from new products

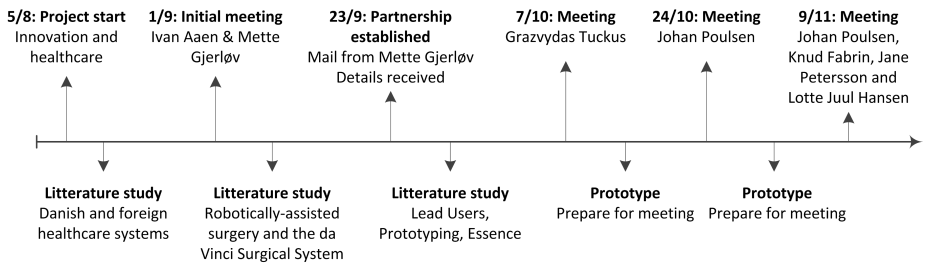


Figure 3.1: Timeline of our previous semester

and services that serve their needs

- They are at the leading edge of their area and therefore experience needs before regular users

These properties are often very beneficial for innovation because the higher need for improvement may motivate the lead user to go the distance needed to get this improvement, e.g. use time to study a completely different field (e.g. a surgeon studying software engineering) to build a solution.

Furthermore, rather than having to imagine what needs a user may experience in the future, the lead user can look at what needs he is currently experiencing. While it may still be hard for a lead user to determine his needs, it is a lot easier than trying to imagine future needs, and therefore a lead user has higher potential when it comes to determining needs and whether proposed solutions will satisfy that need.

We studied the creation and use of prototyping, especially when it comes to filtering and manifestation as described by Lim, and used this knowledge to create paper prototypes and simple software prototypes early in the development process. Prototyping is further studied in Chapter 5 and our use of prototyping is explained in Section 7.1.1

Essence were studied along with software innovation in general, and we gained knowledge on how creativity can be improved during development of software. Essence is described in detail in Chapter 4 and our use of Essence is described in Section 7.1

As the surgical staff at the Department of Urology has a lot of domain knowledge that we need to access, we studied knowledge sharing. Nonaka and Konno describes the concepts [Nonaka and Konno, 1998]:

- *Socialization*: Sharing of tacit knowledge through being together, e.g. working together
- *Externalization*: Externalizing tacit knowledge into forms that can be understood by others
- *Combination*: Explicit knowledge is combined to sets of explicit knowledge
- *Internalization*: Explicit knowledge is integrated in the organization and becomes tacit knowledge

As we wanted to access the domain knowledge of the surgical staff and externalize our technical knowledge to the staff, both being tacit knowledge, we focused on socialization and externalization to enhance knowledge sharing during our short meetings. By making prototypes, we became able to externalize our technical knowledge to the surgical staff team, and by having them comment our prototypes, we were able to take note and gain tacit knowledge, e.g. how they work in the operating theatre and what kind of information is important to collect with the markup solution.

3.2 Project Course

The collaboration with Aalborg Hospital started in the beginning of August where we were introduced to robotically-assisted surgery (RAS), and given information about the current status of RAS.

We had our first meeting with special consultant Mette Gjerløv on the 1st of September. About a month later, we had our first meeting with a surgeon, Grazwydas Tuckus, on the 7th of October, where we were introduced to the kind of surgeries they perform at the Department of Urology. It was at this meeting we were given the surgical step order of the surgery for use in our prototypes. From the information we gained through the meetings and emails from August to October, we began thinking about what kind of system we could design to help them improve efficiency, quality and education. We considered the following:

- Allocation of materials to be used during the surgery
- Enhance use of these materials such that as little materials are wasted as possible
- Improve surgery efficiency
- Improve surgery quality
- Enhance communication between surgeon and patient after the surgery

Most if not all of these considerations require data collection of some sort. Therefore we created a list of data that could be interesting to collect, created a paper prototype showing how the collected data could be analysed and presented this at our first meeting with the chief surgeon Johan Poulsen and special consultant Mette Gjerløv at the third meeting on the 24th of October. They both found it very interesting and saw the same prospects in regards to enhancing efficiency and quality of the surgeries. Because they both are involved in *Minimal Invasiv Udviklingscenter*[1] (MIUC), where new surgeons are educated in the use of RAS, the data collection and analytical tool could be used when educating the new surgeons.

A second prototype were developed to show our initial ideas of how to collect data about surgical step durations and how to use these durations for comparison of surgeries in regard to the time used. These step durations can also be used together with recordings from the endoscopic camera feed in the *da Vinci*® Surgical System, to store a video clip for each surgical step. This would allow the surgeons to review steps directly rather than having to navigate through the complete video clip of the surgery. Currently the surgeons are not recording surgeries on a daily basis either, so getting a tool for recording and viewing surgeries would be most welcome.

The new prototype was created as a touchscreen prototype to be used on a current monitor in the operating theatre as an overlay. The monitor in question can be seen to the right on Figure 2.3, page 15.

By allowing the surgeons to watch videos of previous surgeries, several possibilities unfold:

- If a coming surgery is unlike most surgeries, e.g. if the patient has previously had a surgery in the same area, and therefore has scars and the surgery therefore has to be done differently, the surgeon can watch videos from previous surgeries in preparation

[1] Minimal Invasivt Udviklingscenter: `http://http://www.miuc.dk/`

- If the surgeon encounter issues during a surgery that he does not know how to handle. He may need consultation with other surgeons, and using videos from similar surgeries may be helpful to show the surgeon how to handle the issue

- If a surgery had a bad outcome for the patient, the surgeons can watch the video of that surgery and determine if anything was done incorrectly; discuss how to better handle such surgeries and thereby become better surgeons

We showed the new prototype to chief surgeon Johan Poulsen, surgeon Knud Fabrin, surgeon assistant Jane Petersson, and nurse Lotte Juul Hansen at our meeting on the 9th of November, and received feedback on the prototype. The use of a overlay on their existing monitors was not useful as they are already crammed with information, and the surgical staff needs full overview of the monitor. Sterility becomes an issue as well, as there are high requirements when it comes to hygiene in the operating theatre, and therefore a touchscreen has to be kept sterile if it is to be used by the sterile personnel during the surgery.

The main idea however, collecting and analysing surgery data, is still very interesting, and therefore we arranged a visit to the operating theatre during a real surgery on the 16th of November. During this day we gained a lot of knowledge on how they conduct surgeries, and we got inspired for other kinds of interfaces that they can use during surgeries, such as pedals and speech recognition. We also found that a touchscreen may be of more use if it is not used as an overlay but instead as a separate monitor, thus making it easier to sterilize with plastic and place it near the user.

We noted these ideas and finished our report for delivery on the 5th of January, and thereafter began working on the touchscreen, speech recognition and Pedal Prototypes described in this report.

3.2.1 Toulmin Structure

During our previous semester we used a Toulmin Structure to explain the main vision of this project. Below is the Toulmin Structure as of the end of previous semester [Jakobsen and Follin, 2011]:

- *Challenge:* The surgical staff at the Department of Urology wants to improve their current processes in respect to efficiency, quality and education. By efficiency the main problem is time, as they want to be able to conduct more surgeries every day than they are currently capable of. They also want to increase the quality of their surgeries in respect to the patient. The Department of Urology has a centre for educating new surgeons, and improvements to surgery processes should be used by interns as well.

- *Idea:* Create a tool that allows for collecting data (e.g. durations of each step) from surgery. The tool must allow for collection and analysis of several kinds of data, e.g.:

1. Special criteria, e.g. if the patient have had previous operations on same and/or nearby organs
2. Complications during the surgery
3. Efficiency in regards to time used, e.g. step durations.
4. Compare critical steps of the surgery; what went wrong or well in terms of patient health?
5. Education of new surgeons, using the two points above.

- *Grounds:* Robotically Assisted Surgery (RAS) requires capital investment. In order to be cost-effective, RAS should lead to effective and high-quality surgery. For this reason, procedures must be optimal and staff must be educated to use these procedures appropriately. The surgeons do not have time for watching entire videos of previous surgeries, and only certain parts of the surgery are really important in regards to potency and continence, so video should be split up into one video clip for each step. The markup should be done on-the-fly and still with high precision, and therefore there is a need to find the most capable staff member to do the markup, in order to get a precise markup. Graphs, tables and other visualization methods may make it easier to compare a lot of surgeries.

- *Warrant:* Making RAS more efficient would make it more useful for other types of surgeries where RAS may be considered. It has the potential to become a completely new market for IT companies developing solutions for use with RAS.

- *Qualifier:* Assuming we can
  - get high-fidelity surgery data with appropriate granularity,
  - determine relevant units of analysis, and
  - fit our solution into the work context of an operating theatre.

- *Rebuttal:* The surgeons know when steps begin and end. They are thus able to determine when it happens and noting the time (markup) will give us the durations of each step. They know which information are relevant to them in order to analyse surgeries. The surgeon thinks the use of voice recognition would be useful to him as he feels comfortable talking while doing the surgery. Foot pedals could be used in combination with touchscreen, using the foot pedals to markup steps and touch screen to mark everything else. The markup task can then be shared among the staff, one using the foot pedals and others using the touch screen.

4

Essence

This chapter describes the Essence framework and the components used in this project. The theory described in this chapter is based on [Aaen, 2012, Aaen, 2008] and discussions with Ivan Aaen. Our own use of Essence is described in Chapter 7.

Essence is a framework with tools and structures made for increasing innovative thinking in software projects and building focus on creating innovative products. This is in contrast to usual software development methods, be they agile, traditional or in-between, where the focus is mostly on efficiency in order to meet the requirements and deadlines given by the customer, and rarely if ever on creativity within the team.

Aiming for efficiency is important, especially for projects with a tight deadline and/or budget. However, being able to create innovative products are better than what is currently on the market and offers higher value to the customers through new and better features than the competing products.

The probably most well-known example of this is the iPhone. While it is more expensive than most competing products, it offers value to the customer through features and platform that makes it a very popular product.

Agile methods aims for flexible development with ability to deal with e.g. requirements and market changes through iterative and incremental development, close customer relation through on-site customers and user-driven innovation. Essence is somewhat related to the agile paradigm in regards to these aspects [Aaen, 2008]. However, with the main goal of agile development still being efficiency, Essence diverge from the agile paradigm as innovation becomes the goal. Where agile and traditional approaches aim to give the customer exactly what was agreed on (e.g. comparing the end product to the contract between the customer and the developers), Essence aims towards giving the customer even better products than expected through creativity and innovation.

An advantage of this approach is that customers rarely if ever know what they actually want [Larman, 2004]. Furthermore, customers do not have the same technological knowledge as the developers and therefore cannot see the potential of those technologies. By sharing knowledge through socialization [Nonaka and Konno, 1998], the customers get to see some of the potential and the developers get a better idea of the design space, leading to more ideas from both parties.

Innovative ideas may be very spontaneous and seemingly impossible to create on demand, often experienced as a "aha" moment. One of the ideas of Essence is to employ tools to increase the chance that such moments of creativity happens more often, creating more ideas during the process. With more ideas at hand it becomes possible to choose between the ideas and weed out ideas that are not feasible.

However, a high quantity of ideas does not guarantee that good ideas will occur. To increase the chance of creating ideas of higher relevance and quality, Essence uses scenarios, where the customers and developers describe the design space. Scenarios allow the team to explore how usage scenarios unfold and thereby to penetrate deeper into the combined problem and design space. The team can benefit from diversity and interactions in the team while maintaining a shared focus and a shared understanding.

Essence is intended to be integrated into an agile approach to software development and function as a set of tools to be used when the development team reaches moments of creativity whether planned or not. Some tools are used to generate idea and/or store descriptions of the ideas while others are used to evaluate and/or mature the idea.

One of the important properties about Essence is that it is lightweight on process and heavy on structure. The goal of these properties is to make it easier to use Essence spontaneously as you do not have to follow a certain method. Instead, the structures can be used in the way that serves the team and the project best.

4.1 Roles

In a team it is important to be able to give arguments and counter arguments to ideas and decisions. In order to sustain a mind-set and allow the developers to view entities from different angles, the concept of Roles is introduced. By taking a role, each developer (or customer) can take a viewpoint and use it to e.g. give input and questions to an idea. The roles in Essence origin from existing roles in the workplace, which makes it easier to integrate Essence into an existing development method as no roles have to be introduced.

In Essence the roles are more pronounced, and ideas will origin from roles rather than from the persons taking the roles. A developer can take a role and through that role, propose ideas that would not normally come from a developer. This may be useful to teams who have worked together for long and have each found a role for themselves they rarely if ever shift from. This may hinder the developers in brining new and different ideas to the table in fear of judgment by the team.

There are four roles in Essence, and everyone in the development has a role at any time of the project.

4.1.1 Challenger

The Challenger is the customer (or someone representing the customer and their interests) and is therefore the one to supply the current challenges to be solved by the developers. The Challenger should determine if solutions given by the developers may be able to solve the given challenges.

The Challenger has the domain knowledge, the context for the Challenge, and is therefore able to use his competencies to explain the Challenge truthfully and answer relevant questions from the rest of the team. At the same time the Challenger has leadership capabilities and is able to focus on the main goals of the project and ensure the goals are reached in the end, even when severely outnumbered by developers in the project.

In agile development using an on-site customer is common, and Essence is designed to make use of the on-site customer as the Challenger. Whenever a on-site customer is not available, potential Surrogate Challengers include sales representatives and product line managers as they have regular contact with the domain of the project and thereby domain knowledge.

The Challenger has some responsibility for project management, such as:

- Track project progress
- Building a shared vision for the team and identifying the main goals for the project
- Manage priorities, e.g. which features to develop and when
- Coordination of teams and external activities
- Support project overview, being able to identify and act upon changes and issues in the project, e.g. delays

Rather than focusing on deadlines and requirement lists planning, the Challenger focuses on scenario-based planning. The Challenger also ensures that the scenarios in Paradigm View are relevant and represents actual scenarios. The Challengers main responsibilities reside in the Project View where the Challenger maintains the overall Challenge and vision for the project.

4.1.2 Responder

The Responders have the technical knowledge for the project and uses this knowledge and their creativity to devise potential solutions to the challenge given by the Challenger. This role is designed to be used by the developers, e.g. the *Programmer* in XP and the *Team Member* in Scrum.

The Responder role is related to the agile principle of "working software over comprehensive documentation" in the sense that the Responder creates working software based on user stories and shows the value of the software solutions by demonstrating them under realistic settings.

Whereas the Challenger supplies the Responder with the correct settings, scenarios and domain knowledge, the Responder is responsible for supplying and explaining the technological possibilities. This includes being able to determine which technologies are the most fitting for the project and adds the most value to the final product while still being realistic in regards to time and budget. The Responder also explores the application domain to find areas where technologies can be used to enhance these areas, e.g. increase efficiency in a process by automatizing tasks that are currently done manually.

To be able to determine whether a technology is useful for the project, the Responder may need to research the technology and conduct experiments with it. The Responder is able to explain his ideas in a way that the Challenger understands

- the values of the ideas (how the ideas match the main vision for the project),
- the advantages and disadvantages of the ideas, and
- the technologies needed for the ideas.

By supplying these informations about the ideas, the Challenger can then make a competent choice among the ideas and use these to change the scenarios so they make better use of them and solve the challenge in a better way.

While the Challenger has the final say when it comes to the main vision and choice of ideas, the Responder is very important as they know how much of the vision is actually feasible to develop solutions for, and therefore the Responder has a great influence in the Project View and in the management of the project.

4.1.3 Anchor

The Anchor manages the rest of team and ensures that the team is equipped with the necessary tools and techniques. This role is responsible for ensuring the project is always progressing, that is it iterating towards a solution for the challenge. Like the *Scrum Master* in Scrum and unlike a project manager in traditional development methods, the Anchor is not above the other actors in the team. The choice of tools and techniques is based on the team and their preferences rather than being chosen by the Anchor.

The Anchor also has Responder responsibilities, which is much alike the *Scrum Master*. Furthermore, the Anchor is responsible for ensuring the teamwork is functioning well and that issues between the team members are handled.

The Anchor facilitates and manages the discussions between the Challenger and Responders to ensure the discussion is kept on track and focused on solving the challenges at hand. When idea evaluations and research is conducted, the Anchor must ensure they are done properly. To do this, the Anchor can

- motivate for more creativity and idea generation or focus on a set of ideas and mature them,
- call for timeouts and recovery, and
- intervene when the discussion goes wrong and get the team back on track.

The Anchor has much of the responsibility for the idea evaluation because he is the one to manage discussions in the team.

When the team is well functioning, the Anchor role is not much different from being a Responder. However, when something goes wrong the Anchor is responsible for bringing order and finding the balance between when to intervene and when to let the team continue in its current direction.

4.1.4 Child

The Child is a free role that can be taken by Challengers, Responders and the Anchor at any moment in order to raise an issue or idea that may not be fitting for the current role of that person. The Child is allowed to stir up the discussion and look at issues from a different, even a naive, viewpoint. It may even suggest ideas that are contrary to earlier decisions.

The Child is very optimistic and uses a *let us try* approach rather than reject ideas and suggestions, which allows for greater exploration and learning. While the Child is allowed to take this approach, it may be ignored by the other roles, and in that aspect the Child can be compared to the *Chicken* in Scrum.

Whenever someone is working on developing ideas, they are considered to be in the Child role and outsiders, e.g. guests and external stakeholders, are considered to be in the Child role whenever they interact with the team. By taking on the Child role, the differences between other roles are minimized if not removed, and the team now work together in exploring and experimenting with the application domain and technology domain.

As a project progresses and move towards the end, ideas may often be related to previous ideas, limiting the amount of radical ideas in the end of the development. If a team member feels this is the case, the Child role can be used to address this issue and try to bring some radical ideas to the project by taking a new aspect compared to the usual role of that team member.

While working on idea generation and scenario exploration, the Child reflects on the discoveries made in the process and gain an understanding of the possibilities and the limits of the project. Because of the different viewpoint of the Child, these discoveries can potentially open up for new radical ideas.

4.2 Views

The amount of ideas and experiments can easily become overwhelming, and thus it is necessary to organize them in order to give an overview of all the ideas and the project as a whole. In order to do so, Essence introduces the concept of Views.

A View is a physical location, e.g. a smart board, where related project information can be stored for overview and communication between the actors. Summarizing Aaen, each view is used for [Aaen, 2012]:

- *Overview*: Getting the whole picture of the project
- *Separation*: Supporting a divide-and-conquer style of work without sacrificing coherence
- *Combination*: Being able to take diverse aspects of a set of related problems or solutions into consideration at the same time
- *Collaboration*: Allowing team members to contribute interactively and simultaneously as they see fit

By sorting related information in each View and having each View at a certain physical location the developers can employ a certain mind-set to each location and use each View to discuss certain aspects of the project.

Each view is described in the following sections.

4.2.1 Paradigm View

The Paradigm View is where most of the development begins as this View is used for idea generation, exploration and experimentations following these. This

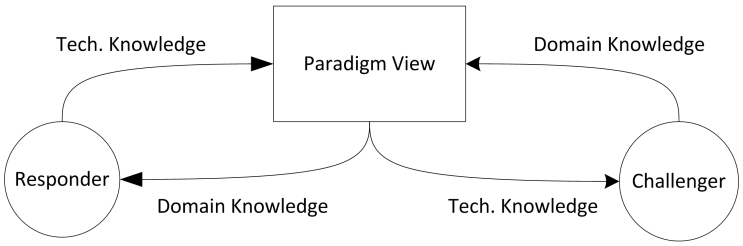


Figure 4.1: The sharing of technological and domain knowledge between the Challenger and Responder through the Paradigm View.

view is where the design space is explored and where the developers work on understanding the aspects of the problem they are building a solution for.

Understanding a problem can become very complex because of domain knowledge. Knowledge that the customers have used years to learn and considers a obvious part of their work may be completely new to the developers. This kind of knowledge is described as *tacit* knowledge, knowledge that is hard to explain and teach to others [Nonaka and Konno, 1998].

One way of exploring tacit knowledge is by describing scenarios together with the Challenger, thus making sure that the developers understands the situation that the developed system is to be used in. The Responders serve the Challenger with information about technologies to give the Challenge an idea of what is possible.

Understanding the design space, developers can begin to build their technical solutions and explain how certain technologies can complement the design space and create a solution for the challenges posed by the Challenger. Though ideas are explored and worked on in this view, they are not directly evaluated, chosen nor abandoned in this view. This takes place on the Process View.

The design space is to be explored in this View, allowing the Challenger to learn about the technological aspects while the Responders learn more about the application domain. Through the newly gained knowledge, the Challengers can get ideas regarding how to use these technologies on other parts of the application domain. By explaining these ideas, more of the application domain is externalized and transferred to the Responders. The Responders can thereafter apply their technological knowledge to the parts of the application domain they just learned about, externalizing more technological knowledge to be picked up by the Challenger.

This process of sharing knowledge through socialization and externalization around the Paradigm View is depicted on Figure 4.1.

Tools such as use cases and state diagrams can be useful for trying to grasp the domain knowledge and gain a overview of the challenge. Another tool that may be used in Paradigm View is simple prototypes, e.g. a GUI to show how an idea potentially could be used in the design space.

4.2.2 Process View

While exploring the design space and creating many ideas is important for creating innovative products, the ideas have to be evaluated and chosen/rejected for the actual product at some point. The Process View is where further ex-

amination and evaluation takes place in order to choose the best solutions for the challenge to be solved. The Process View is also used for research strategy evaluation, to decide which options and ideas should be researched further.

When an idea is hard to analyse and the team members are unsure about the advantages and disadvantages, e.g. new and unproven technology. Putting some effort in researching the idea/technology further may be necessary to determine whether the idea may be useful for the final product or the technology has the needed potential to be considered for the actual implementation.

When more than one person works on a project there may be ideas that one person grow fond of while others do not see much potential in it. Evaluating the ideas on their advantages and disadvantages and thereby discuss their potential is important to weed out some of the ideas. It may also help the team members to gain insight to what the other team members see in the idea in order to mature it.

Some of the tools that may be used in this view are *Plus-Minus-Interesting* (PMI) and *Strength-Weaknesses-Opportunities-Threats* (SWOT) analysis' to evaluate the generated ideas. By using these tools the team members get to express the advantages and disadvantages they see in the ideas and can thereafter argue whether the advantages and potentials outgrow the disadvantages and threats.

4.2.2.1 PMI

PMI is used to specify advantages (Plus), weaknesses (Minus) and interesting information about ideas. By noting all the known/expected advantages and weaknesses it is easier to get an overview of the idea and evaluate it. It is also useful for finding out if the team have the same conception of the idea. One team member may have a solution for one of the weaknesses and other team members might see disadvantages others do not. To store the information in a PMI format, a table is useful to easily list up the advantages and weakness and show them next to each other for comparing ideas to each other.

PMI is used in all of our prototypes, described in Chapter 8–13.

4.2.2.2 SWOT

The SWOT analysis is somewhat more thorough than PMI as it focuses on the Strengths and Weaknesses of the idea along with Opportunities and Threats. Strengths and Weakness are used to describe advantages and weaknesses of internal origin, e.g. evaluation of the features of the idea. Opportunities look at the advantages that are external to the idea, e.g. market advantages. Threats describe the external threats. An example could be competing companies and their products and/or upcoming products.

While SWOT is more thorough and considers external factors, it is not designed for easily comparing one idea to another as is the case with PMI. SWOT is used in all of our prototypes, described in Chapter 8–13.

4.2.3 Product View

The Product View focuses on how potential solutions might be implemented. This View consists of the design of the software product and the technical as-

pects that follow, e.g. software architecture. Class diagrams, system diagrams and even source code may be placed in this View as well for discussion on the architecture.

Along with design and implementation of the potential solution, the Product View is also used for finding out how much can be achieved with relevant technologies, both in terms of the chosen solution and for alternative solutions in order to help idea generation. In [Aaen, 2012], this is described as *technology affordance*. Technology affordance is also useful for finding out which additional features may be possible with the technology in question, thereby opening up for idea generation.

Technology affordance is especially useful in regards to impending technologies, where research may be able to determine whether new technologies may be more suitable as solution to the challenges given by the customer.

Some of the tools that use the Product View are SCAMPER and Six Serving Men, both described in the last section of this chapter.

4.2.4 Project View

While the other Views are very detailed with descriptions of many idea, design and implementation details, the Project View is used to get an overview of the project. This View is used to plan the project and describe the main goals of the project along with task lists, visions and other management tools.

The main vision of the project can be described in the Project View to show where the project is heading and what to focus on in the other views. The vision should then be reflected in the evaluation and choice of ideas from the Process and Product Views. The strategies for researching technologies and ideas is also described in the Project View, e.g. which tools and techniques to use.

Some of the tools that can be used on the Project View is the Toulmin Structure and the Elevator Pitch, each described in the following sections.

4.2.4.1 Toulmin Structure

When describing the main vision and idea of the project, the Toulmin Structure is useful as it gives a presentation of different aspects on the idea. On top of describing the challenge and idea of the project, it is important to describe how the solution is to solve the challenge. As solutions are rarely if ever perfect, issues and limitations to the solution will show up and they should be described as well.

Through the entities of the Toulmin Structure seen on Figure 4.2, these aspects can be described in one structure to give an overview of the project and vision.

The Toulmin Structure consists of the following parts:

- *Challenge* describes the challenge of the project, the main problem to be (partially) solved by the Idea.
- *Idea* describes an idea that may solve the Challenge. The Idea might not be able to solve all parts of the Challenge as well as it might be able to solve the Challenge and even more problems outside the scope of the project.

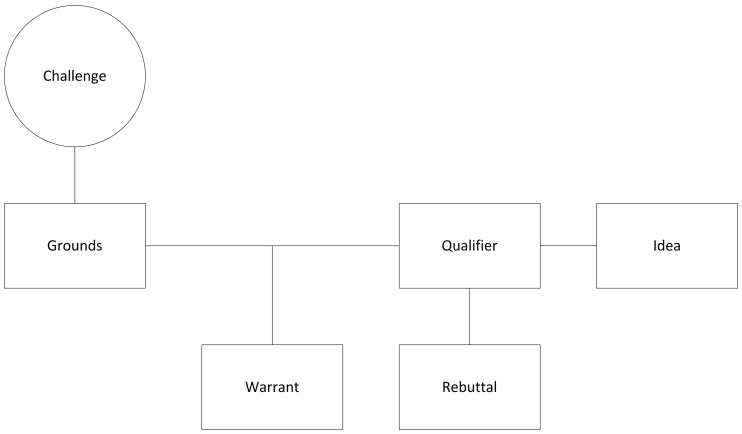


Figure 4.2: Overview of the Toulmin Structure

- *Grounds* describes the actual need for the solution and functions as a base for the idea, e.g. by describing the infrastructure the idea is to be based on and how the idea can improve the current situation.
- *Warrant* links Grounds with Idea. Warrant works from a more general aspect, e.g. argue if the solution could be useful on a broader scale and/or describe whether the technology needed for the Idea is available or will be within project time limits.
- *Qualifier* describes the issues bound to moving the specific issues in Grounds to the general aspect taken in Warrant.
- *Rebuttal* is used to describe solutions that help the issues pointed out in the Qualifier, thus which changes would be needed to raise the Idea to work on a broader scale.

The Toulmin Structure is primarily managed by the Challenger who has the final say over the Toulmin Structure.

4.2.4.2 Elevator Pitch

The Elevator Pitch is based on a "test", being able to describe the product vision in the time it takes to use an elevator (or within two minutes).

The Elevator Pitch is useful for describing the project vision in a lighter way than the Toulmin Structure. However, the Elevator Pitch is not concerned about idea maturation and describing the potential of the idea. This is opposite to the Toulmin Structure as it contains thoughts on the further development of the idea through Qualifier and Rebuttal.

The Elevator Pitch consists of:

- *For:* The target customer
- *Who:* The need and/or opportunity that this product fulfills
- *The:* The name and type of the product

- *That:* The main advantage and selling point of the product
- *Unlike:* The main alternative to your product
- *Our product:* The different between your product and the alternative

When an idea is matured, the lightweight Elevator Pitch is advantageous as it is quick and precise, short and still describes the product and how this product may benefit its users.

4.2.5 Tools

Some tools in Essence span over several if not all of the views.

4.2.5.1 SCAMPER

The purpose of SCAMPER is to mature, examine and extend the idea by changing parts of the idea. By asking questions to parts of the idea, the aim is to provoke new ideas or changes to the current idea.

The questions are set up in 7 groups based on the kind of question:

- *S - Substitute:* When substituting, the Product View is useful for picking amongst technologies that may be of use in a substitution. Substitution may apply to other parts of the idea, e.g. target customers.
- *C - Combine:* Combination may change the potential of an idea, and technologies can be the main target, e.g. combining two technologies to get the best of both technologies and have the technologies back up each other.
- *A - Adapt:* Adaption can e.g. be used to change the purpose of the idea/product; using the product in a new context. Looking at similar products for inspiration may be helpful as well to adapt the idea.
- *M - Magnify:* Magnifying parts of the idea, thus focusing on that part or even exaggerate parts of the idea may open up for new ideas as current barriers can be ignored for a moment.
- *P - Put to other uses:* Finding a new purpose for the idea/product may open up for new markets and make the idea much more useful.
- *E - Eliminate:* Elimination is useful for simplifying an idea, e.g. by looking if parts of the idea can be removed without altering the main functionality.
- *R - Rearrange/Reverse:* Rearrangement/Reversing can be used to change part of the idea and give thought to how the idea would work out if things are done in a different order, even in reverse order. This might for example be to change the idea to do the exact opposite of what it originally does.

4.2.5.2 Six Serving Men

The purpose of Six Serving Men is to describe the main features and properties of an idea and ensure the team have the same view on the idea.

Six Serving Men consists of six questions

- Who will use these features?
- What/Which components and architectures?
- Where are the features used?
- When should the components be available?
- Why are these main features needed?
- How is a feature used?

The answers to these questions are to be found in the different views. The scenarios in Paradigm View serve to explain *Who* is to use the features, *Where* to use the features and *How*. Product View supplies descriptions of *Which* architecture and components that are to supply the features. Finally the Project View supplies the answer to *When* and *Why* through project planning and the vision for the project.

5

Software Prototyping

In this chapter we look at how software prototyping and prototypes are defined in the literature, and how prototyping is typically used during the development lifecycle of a software project. Going beyond requirements, we look at how prototypes can help incorporate creativity in the generation of insights and solutions, and finally rationality to analyse and fit solutions to the context, into the development lifecycle. Chapter 7 builds upon this chapter, and describes how we have utilized this knowledge to incorporate prototyping into our development process.

We start by defining *prototype* and *prototyping*, but because there is no generally accepted definition we establish what we believe is the most generally accepted definition.

Bernhard Boar [Boar, 1984] has defined *prototyping* as a specific strategy for performing requirements definitions wherein user needs are extracted, presented and refined by building a working model of the ultimate system quickly and in its working context.

Connell and Shafer [Connell and Shafer, 1989] defines a *software prototype* as:

> *"A software prototype is a dynamic visual model providing a communication tool for customer and developer that is far more effective than either narrative prose or static visual models for portraying functionality. It has been described as:*
>
> - *Functional after a minimal amount of effort*
> - *A means for providing users of a proposed application with a physical representation of key parts of the system before system implementation*
> - *Flexible modifications require minimal effort*
> - *Not necessarily representative of a complete system."*

Boar's definition of software prototyping as a strategy for performing requirements definitions, is acknowledged by a range of authors such as Fred Brooks [Brooks, 1987], Naumann and Jenkins [Naumann and Jenkins, 1982], and Christiane Floyd [Floyd, 1984] etc.

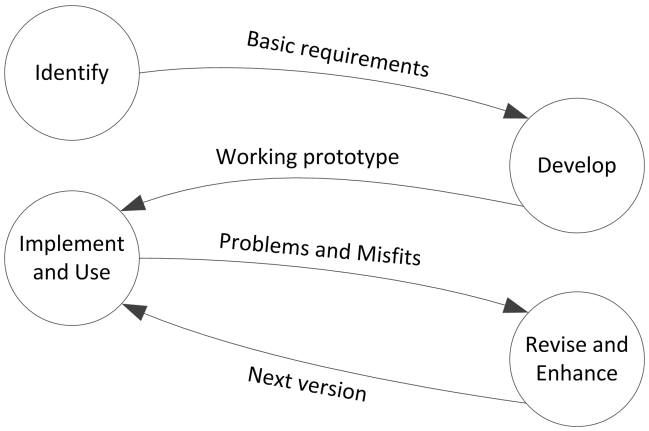


Figure 5.1: Prototyping is a four-step interactive process between User and Developer [Naumann and Jenkins, 1982]

It is important to note that when we use the term prototyping in connection with software development, it indicates that we are primarily interested in a process rather than in the prototype as a product. The focus of the process is on building a working system as quickly as possible instead of documenting the user's requirements and building the system afterwards.

Naumann and Jenkins describes a very general approach to prototyping, called the *prototype model* which is a four-step procedure as depicted on Figure 5.1 (or as Floyd describes it in [Floyd, 1984]; functional selection, construction, evaluation and further use). Each step is briefly described below:

1. *Identify:* First it is important to identify the most essential features of a user's requirements and based upon these, develop a working prototype.

2. *Develop:* To get the process started, it is important that this initial prototype is implemented in a very short time, ideally not more than a day or two (*rapid prototyping* [Floyd, 1984]). This time requirement serves both the user and the developer, because the user gets a tangible system quickly to experience and criticize, and the developer gets responses based upon that experience.

3. *Implement and Use:* The "hands-on" use of the system provides experience, understanding and evaluation of the system, and allows for better feedback to identitfy problems and misfits that are later to be revised and enhanced. Note that by *Implement*, Naumann and Jenkins refers to how the prototype is implemented in the users' everyday work, and not the process of building the prototype (see *Develop*).

   The fact that we present a prototype to the users and make them use it, we meet a fundamental goal of design. According to Christopher Alexander [Alexander, 1964]:

> *... the process of achieving good fit between two entities is a negative process of neutralizing the incongruities, or irritants, or forces, which cause misfit.*
>
> *... the experiment of putting a prototype in the context itself is the real criterion of fit.*

In this case, the two entities are the solution to the problem and the context that the solution is to work in. The process is negative in the way that it removes misfits and incongruity, however the absence of certain negative qualities does not necessarily make it a good fit.

4. *Revise and Enhance:* Undesirable, missing features or misunderstandings identified by the users must be corrected. Thus we can say that the prototype model exploits this *negative process* that Alexander talks about, rather than deplores it.

   The user is very unlikely to identify all the remaining problems so several iterations will most likely be required. Thus step 3 and 4 must be repeated until the user accepts the system as a good fit.

5.1 Approaches to Prototyping

Having established the most fundamental understanding of software prototyping, we now turn to how one should choose to approach prototyping depending on the goals to achieve. [Floyd, 1984] distinguishes three broad classes of prototyping:

- Exploratory prototyping where the emphasis is on clarifying requirements and discussing alternative solutions.
- Experimental prototyping where the emphasis is on determining the adequacy of a proposed solution before implementation in the final system.
- Evolutionary prototyping where the emphasis is on adapting the system gradually to changing requirements (not determinable in the early phases).

One does not strictly choose a single approach, and the borderline between them (especially exploratory and experimental) is deliberately unclear, but distinguishing between them is useful for clarifying the relation between prototyping and the development lifecycle as a whole. Each class is described in the subsequent sections.

5.1.1 Exploratory Prototyping

This approach focuses on the communication between software developers and users, particularly in the early stages of software development. Normally, developers have little knowledge about the problem area and the users might not have a clear image of what the system to be developed might do for them. In this situation, a prototype as a practical demonstration of possible features can serve as a catalyst to elicit good ideas and promote a creative cooperation between all parties involved.

Exploratory prototyping is very informal by nature, and there are no strict rules on how to utilize these prototypes, however the steps: functional selection, construction, and evaluation take place as needed within the overall communication.

[Floyd, 1984] says that for prototyping to be successful, the developers must have a strategy that pertains to the choice of features to be included in the prototype, because users' expectations will be deeply influenced by the exposure to the prototype. It is important to control these expectations, not making the prototype too "complete" but instead keeping it simple, displaying only a few set of features that the users can evaluate. The advantages of incompleteness in prototypes is described in [Lim et al., 2008] and is further explained in Section 5.2.1. At the same time it should be clear that there is no commitment to reproducing the prototype in the final system, but rather incorporate the good ideas derived from the exploration.

These kinds of prototypes are expected to be messy and unstructured, and they are normally thrown away (the process is often referred to as *throwaway prototyping*), and so it is also important not to put too much time into producing the prototypes.

5.1.2 Experimental Prototyping

In this approach a proposed solution to the problem is evaluated by experimental use. There are several different strategies to take into account when taking this approach, some of them being (described in [Floyd, 1984]):

- *Full functional simulation* where the prototype exhibits all the functions of the target system intended to be available to the users for normal use. The prototype may be constructed using techniques which offer ease of implementation and modification rather than efficiency of the prototype. Such a system may be impossible to use as a production system, because it might lack implementation behind the parts visible to the user, or lack efficiency, error handling or special cases not taken into account for the prototype.

- *Partial functional simulation* where the prototype is used to test a hypothesis about the system. For example, to see whether a proposed algorithm will produce acceptable results efficiently.

- *HCI simulation* where the user is presented with the proposed interface in its intended final form, but uses mock-ups for other parts of the system.

- *Skeleton programming* exposes the users to the overall structure of the system on the basis of a few system functions selected as being relevant. This involves the design of the whole system and a drastic reduction of its implemented functional scope. The functions implemented is to be used by the users to perform work tasks and demonstrates how the system will be embedded into the users' overall work process. It is also very useful for demonstrating the intended system's efficiency and thus get a feeling of what would be acceptable to the users.

Exactly which strategy to use should be discussed by taking into account the particular communication needs of the situation in hand as well as the available

resources, techniques and tools. The strategies is not an either/or choice, they can be combined or several strategies can be utilized depending on the current state of the project and what the goal is.

Where exploratory prototyping is primarily about defining software requirements, experimental prototyping is appropriate through the whole development lifecycle, not only for communication with the user, but also between developers (such as partial functional simulation and skeleton programming).

Depending on the strategy chosen, the prototype might be thrown away. But strategies such as full functional simulation may be expensive to create and in some cases these can serve as part of the final product or incorporated in a revised form on the basis of new requirements which were clarified during evaluation.

5.1.3 Evolutionary Prototyping

Evolutionary prototyping [Floyd, 1984] is remarkably different than the other two approaches to prototyping, however they still have things in common and exploratory and experimental prototyping are appropriate early steps of an evolutionary strategy. Evolutionary prototyping is based on the experience that

- the organization surrounding the system to be developed evolves, and therefore new requirements emerge, and
- the system itself, once it is used, transforms its usage context and thus itself gives rise to new requirements.

The goal of this approach is to provide a dynamic strategy which views the product itself as a sequence of versions, so that each version can be evaluated and serves as a prototype for its successor. Thus evolutionary prototyping breaks down the linear approach to software development and promotes a more agile approach with successive development cycles. Depending on the degree to which this takes place, we can distinguish between the following forms of development:

- *Incremental system development* deals with complex problems stepwise, and the design and implementation of the system is accomplished gradually in a process of learning and growth. In order to be successful, this stepwise process should be geared to the users' work tasks to be supported by the system, so it becomes possible to gradually train and involve users in the development process with benefits to the communication between users and developers.

  Incremental system development primarily affects the implementation phase and is based on the overall design, so it is still fairly compatible with a linear approach to software development.

- *Evolutionary system development* views software development as a sequence of cycles: design, implementation and evaluation, and emphasis is put on software development within a dynamic and changing environment. Instead of trying to capture a complete set of requirements in advance, the system is built to accommodate subsequent, even unpredictable, changes.

  The number of cycles involved can be tailored to the needs of the situation. For example there can be one or two cycles of exploratory prototyping

> during requirements analysis, one or more experimental prototypes during design, and incremental system development during implementation.

Common for the different approaches to software prototyping, is the fact that they are all concerned about prototyping as a tool for

- identifying system requirements,
- evaluating implemented functionality and
- evaluating the efficiency of the system (overall performance, algorithms, etc.)

This is well agreed upon in the field of Software Engineering, but so far we have not look at how the field of HCI views prototypes, which are somewhat similar, but less concerned about the process and with more focus on being creative and, interestingly, more innovative.

5.2 Prototyping as Design Experimentation

So far this chapter has concerned prototyping and prototypes from an engineering perspective and ignored the design perspective. Where prototyping in an engineering context is a process to support requirements engineering and evaluation, prototyping from a design perspective views prototypes as a tool to support the entire design process.

There has been a tendency within the field of software engineering to completely ignore the design phase in the development lifecycle [Buxton, 2007]. Citing Buxton, he gives an example from the chapter "What Is the Design Phase?" in Nokes' book *The Definitive Guide to Project Management*:

> ... on software projects, for example, the design and build phase are synonymous. [Nokes, 2003]

Looking at other industries where the design phase is more commonly used, we find architecture, film-making and the automobile industry. In the automobile industry, the design phase for a new car, for example, involves the construction of a full-size clay model. A clay model can take more than a month to build and cost over a quarter of a million dollars [Buxton, 2007]. Film-making has a preproduction phase where, for example, ideas are created, rights are ensured and the script is completed and evaluated. Like the automobile industry, the architecture industry makes heavy use of prototyping to explore the problem area and traverse the design space where all possible design alternatives and their rationales can be explored (what Schön describes as *reflective conversations* [Schön, 1992]).

The purpose of the design phase or preproduction in film-making shares the same purpose; the purpose of evaluating whether or not the project can enter a "green-light" state and continue development. This up-front process is of course expensive and time consuming, but it costs nothing compared to the costs that will likely be incurred if one does not make that investment [Buxton, 2007].

Moving the focus away from the engineering view on software development (planning and requirements), Buxton introduces the design phase, as depicted on Figure 5.2. What the figure tries to depict, is that the business, technical and

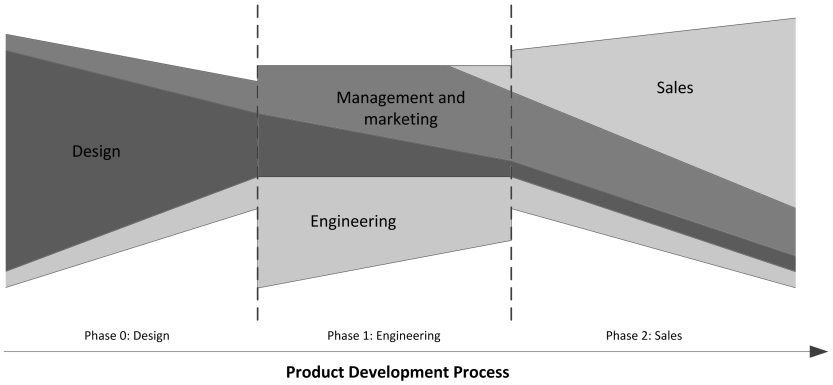


Figure 5.2: Buxton's suggestion of a design phase in the development process [Buxton, 2007]

creative elements must be active in all three phases, but that the focus changes through the process. It is important to note that the "design phase", as Buxton defines it in [Buxton, 2007], is not just the design of the final product. It also includes the design of the engineering process, marketing plan and business model etc. Also, the three phases are not strictly linear as the figure suggests, but can easily be adopted in an iterative process.

But how does prototyping fit into this design phase? Like prototypes in the engineering context described earlier, designers also exploits them for evaluation purpose (for example usability testing). However according to Lim et al. this is a relatively small part of the entire design process, and that prototypes are the means by which designers organically and evolutionarily learn, discover, generate and refine designs [Lim et al., 2008]. This is in contrast to Buxton, where he looks at design in a commercial context in his book *Sketching User Experiences*. Lim et al. views design more generally, focusing on learning and problem-setting through the use of prototypes.

It is important to acknowledge the fact that prototypes are not solely usable for evaluation or proving success or failures of design outcomes, but that they enable *design-thinking* in the process. As Schön was one of the first to point out, product development demands attention to both problem setting and solving, and this is fundamental to the design process, and especially the design phase that Buxton talks about. Looking at prototyping as tools of traversing a design space where all possible design alternatives and their rationales can be explored (like in architecture), allows for a much more creative and innovative design phase [Lim et al., 2008].

This view differs markedly from current approaches in software engineering contexts where engineers use prototypes to identify and satisfy requirements [Floyd, 1984]. Requirement-oriented approaches have their limitations, especially since design activities are *flexible rather than rigid, reflective rather than prescriptive, and problem-setting rather than problem-solving* [Schön, 1982]. A very important fact is that a design idea that satisfies all the identified requirements does not guarantee that it is the best design since a number of ways can meet each requirements [Lim et al., 2008].

So when using prototypes as a means of exploring the design space, what

matters is not identifying or satisfying requirement, but finding the manifestation that in its simplest form, filters the qualities in which designers are interested, without distorting the understanding of the whole. Lim et al. calls this the *fundamental prototyping principle* [Lim et al., 2008].

5.2.1 Anatomy of Prototypes

To support the perspective on prototypes as a means of exploring a design space, Lim et al. defines a framework as an *anatomy of prototypes*:

> *The framework is an attempt to create an understanding of the nature of prototypes in general and to provide a language for articulating the characteristics of a particular prototype. Such a framework will enable designers to specify more effectively the goals and questions to explore when planning and making their prototypes. It will also better guide designers in thinking critically about their approach to prototyping.*
>
> *Two fundamental aspects of prototypes form the basis of our framework:*
>
> 1. *prototypes are for traversing a design space, leading to the creation of meaningful knowledge about the final design as envisioned in the process of design, and*
> 2. *prototypes are purposefully formed manifestations of design ideas.* [Lim et al., 2008]

This section summarises Lim et al. description of their framework. For a more detailed description, please refer to their article *The Anatomy of Prototypes: Prototypes as Filters, Prototypes as Manifestations of Design Ideas* [Lim et al., 2008].

Part of the framework, they identify an initial set of design aspects that a prototype might exhibit. These aspects are called *filtering dimensions*. *Filter* because by selecting aspects of a design idea, the designer focuses on particular regions within an imagined or possible design space. Designers can purposefully avoid certain aspects in order to extract knowledge more precisely and effectively. Through filters, designers can exploit the fact that the primary strength of a prototype is in its incompleteness.

For example in architecture, like mentioned earlier, a two-dimensional prototype of a three-dimensional building can help determine the spatial relationship of the rooms, without placing any constraints on the materials used for walls and floors. This incompleteness structures the designer's traversal of a design space by allowing decisions along certain dimensions (appearances of walls and floors) to be deferred until decisions along other dimensions (spatial relationship of rooms) have been made.

When incomplete, a prototype reveals certain aspects of a design idea—that is, it filters certain qualities. Lim et al. gives an example of a designer who needs to evaluate her ideas about the ergonomics of one-thumb interactions with a mobile device. Testing her ideas with three-dimensional form prototypes, she not only evaluates which ideas work better than the others, but also, more importantly, she discovers what factors of the forms make the ergonomics

Filtering Dim.	**Example Variables**
Appearance	Size; color; shape; margin; form; weight; texture; proportion; hardness; transparency; gradation; haptic; sound
Data	Data size; data type (e.g., number, string, media); data use; privacy type; hierarchy; organization
Functionality	System function; users' functionality need
Interactivity	input behaviour; output behaviour; feedback behaviour; information behaviour
Spatial structure	Arrangement of interface or information elements; relationship among interface or information elements—which can be either two- or three-dimensional, intangible or tangible, or mixed

Table 5.1: Example variables of each filtering dimension

better, leading her to generate new design ideas. So by focusing on these three-dimensional prototypes, a new design space opens up to be explored. A space that may offer possibilities and better choices of the forms of the mobile device that are more effective ergonomically.

The competence involved in prototyping is therefore the skill of designing a prototype so that it filters the qualities of interest to the designer. Thus the most efficient prototype is the most incomplete one that still filters the qualities the designer wants to examine and explore.

Table 5.1 shows four filtering dimensions which corresponds to the various aspects of a design idea that a designer tries to represent in a prototype. They also refer to the aspects of a design idea that the designer must consider in the exploration and refinement of the design.

Besides filtering the qualities of the prototype, designers need to make careful choices about the prototype's *material*, the *resolution* of its details, and the *scope* of what the prototype covers, that is, whether the prototype covers only one aspect of the design idea or several aspects of the design idea. These considerations are called the *manifestation dimensions*. Based on manifestations and filters dimensions, prototypes are intended to traverse and sift through a design space and as manifestations of design ideas that concretize and externalize conceptual ideas.

Design is a continuous coupling of internal mental activities and external realization activities that are constituted through iterated interaction with external design manifestations. Confirmed by Clark [Beynon et al., 2001], this externalization of thought gives rise to new perceptual and cognitive operations that allow for reflection, critique, and iteration. Or like Schön describes it when he states that we have to externalize our ideas, he says that the "world can speak back to us" (also strongly related to his definition of *seeing-moving-seeing* pattern [Schön, 1992]).

Manifestations can take almost any form, shape and appearance, based on the choice of material. From the simplest form of a rough sketch on a piece of paper, or the clay models from the automobile industry mentioned earlier in this chapter. By looking at our own or a colleague's manifestation, we can get a sense of eventual possibilities or limitations inherent in the idea. As an idea evolves

and is refined, the need for more complex prototypes or manifestations increases. Table 5.2 shows three core aspects of the manifested forms of prototypes.

Manifestation Dimension	**Definition**	**Example Variables**
Material	Medium (either visible or invisible) used to form a prototype	Physical media, e.g., paper, wood, and plastic; tools for manipulating physical matters, e.g., knife, scissors, and pen; computational prototyping tools, e.g., Adobe Flash and Visual Basic; physical computing tools, e.g., Phidgets and Basic Stamps; available existing artifacts, e.g., a beeper to simulate an heart attack
Resolution	Level of detail or sophistication of what is manifested (Corresponding to fidelity)	Accuracy of performance, e.g., feedback time responding to an input by a user—giving user feedback in a paper prototype is slower than in a computer-based one); appearance details; interactivity details; realistic versus faked data
Scope	Range of what is covered to be manifested	Level of contextualization, e.g., website color scheme testing with only color scheme charts or color schemes placed in a website layout structure; book search navigation usability testing with only the book search related interface or the whole navigation interface

Table 5.2: Definition and example variables of each manifestation dimension

6

Research Method

To summarize the research questions given in Section 1.1, the goal of this master thesis is to investigate

1. how prototyping can support the Essence innovation process with the goal of developing more innovative solutions;
2. if it is possible to use Surrogate Users instead of the actual users; and
3. whether or not it is possible to map problems from the users domain to the Surrogate Users, still resolving the problems in the users domain.

This chapter describes how we intend to investigate these questions within our case at the Department of Urology on Aalborg Hospital (described in Chapter 2) as an experiment. The goal of this case is to develop a surgery markup tool to be used by the surgery staff during robotically-assisted surgeries at the Department of Urology. Thus the goal of the experiment is to develop an innovative solution that solves the markup problem, and proves to the Department of Urology, that this solution will be worth investing in.

6.1 Prototyping

In order to enhance the possibilities of developing more innovative ideas, we intend to use elements from Aaen's Essence framework described in Chapter 4. We will try to enhance this framework by adding prototyping as a strategy for supporting the overall innovativeness (*RQ1*) [Lim et al., 2008, Rudd et al., 1996].

While looking at prototypes in the context of innovation, we step away from the more traditional view on prototypes where they are understood as a tool for evaluation as in the *prototype model* (see Chapter 5). Prototypes are not only useful for evaluation, but they also enable *design-thinking* in the process, and can be used for traversing the design space where all possible design alternatives and their rationales can be explored (see Section 5.2).

This also complements Christopher Alexanders negative process of achieving good fit (see Chapter 5), by bringing in the positive part where instead of only neutralizing the e.g. incongruities which cause misfits, we bring new or alternative ideas that might ensure a better fit. Obviously, this allows for a

much more creative and innovative design process which we will try to exploit in the experiment.

6.2 Surrogate Challenger

The agile manifesto states *customer collaboration over contract negotiation* and suggest the use of an on-site customer that is involved throughout the process. This role is generally perceived as being very important for agile processes such as XP and Scrum, and statements suggests that it has been utilized with success. Ilieva et al. states that the customer had constant control over the development process, which was "highly praised by the customer at the project sign-off". In addition, Mann and Maurer found in a study on the impact of Scrum and customer satisfaction, that customers believed the daily meetings kept them up to date and that planning meetings were helpful to "reduce the confusion about what should be developed" [Dybå and Dingsøyr, 2008].

However, looking at Martin et al. study of the XP customer role, statements from the customer such as

> *"I think we needed some extra roles basically. We probably needed about three of me ... it is been my life for about a year ... look at these gray hairs ... "* [Martin et al., 2004]

indicates that it can be a lot of work for the on-site customer and the article also concludes that

> *The existing XP customer practice appears to be achieving excellent results, but they also appeared to be unsustainable, and so constitute a great risk to XP projects, especially in long or high pressure projects ... the customer role is difficult and requires serious consideration.*

The fact that the on-site customer role can be stressful and unsustainable for long periods, is also confirmed by Dybå and Dingsøyr [Dyba and Dingsoyr, 2009].

Furthermore Inayat et al. finds that having a full time on-site customer is a rarity instead of common practice in present day software industry. Studies also show that it is not absolutely essential to have an on-site customer 100% of the time as he is needed at most 21% of the software development time, and that customer absence is compensated in several cases by user representative or a Project Manager acting as a *proxy customer* [Inayat et al., 2012].

It is clear from these finding, that the on-site customer (the Challenger role in Essence) requires a lot from the customer and that the project can be very dependent on this role in order to succeed. In our case the chance of having an on-site customer is complicated by the fact that our potential subjects to fulfill this role (the surgery staff and especially the chief surgeon) are lead users within their field, as described in [Jakobsen and Follin, 2011]. The fact that they are lead users, showing a huge amount of domain knowledge within their field, actually makes them a perfect fit for the role. But in this case, people with such comprehensive domain knowledge are very scarce, and the few people who possess it are occupied and therefore unable to act as on-site customers.

To solve this problem we introduce the concept of a *Surrogate Challenger* in Essence (*RQ2*). The Surrogate Challenger is intended to replace the challenger (on-site customer in Essence), but not entirely as the challenger will still be

involved in the process but to a much lesser extent than the on-site equivalent. After all, we still need the challenger to understand the situation we are in and the challenges in the problem domain.

The Surrogate Challenger differs from the customer in a way that he does not have the same knowledge and experience, thus he might see the problem area differently. However, we believe that we can use this Surrogate Challenger to explore a problem domain as close as possible to the original domain, or at least a domain that emit the same challenges. The scenarios and ideas developed with the Surrogate Challenger can then be evaluated together with the customer whenever possible. In order to minimize the lack of knowledge between the Surrogate Challenger and on-site customer, and thus the differences in the problem domain, it is important to find a Surrogate Challenger with at least some domain knowledge, making it as easy as possible for him to see the problem area from the real challenger's viewpoint.

The Surrogate Challenger should of course engage in meetings with the rest of the team and the customer, in order to gain as much knowledge about the problem domain as possible, and in order to understand their challenges. While working on solving these problems through idea generation and by exploring the problem domain and design space, prototypes can be exploited to help the Surrogate Challenger (and the rest of the team) to better explore the ideas and relate them to the real context where they are to be applied. This can further be enhanced by creating an environment where the team can test the prototypes in a context that matches the challenges in the real context.

6.3 Surrogate Users

While exploring the problem domain, generating and evaluating ideas, solutions evolves—often manifested in forms of prototypes. These solutions needs to be evaluated in order to verify whether or not they are usable and solves the problems in the problem domain.

In our case, like with the on-site customer, the customer does not have the resources to establish a testing environment with surgical staff available to test all built prototypes during the process. Instead we will try to test the prototypes with *Surrogate Users* (*RQ2*), users who does not necessarily have the same knowledge, experience and relation to the problem domain. We believe that this should be possible by mapping the problems from the problem domain, that the prototype to be tested tries to solve, to a domain that is more accessible for the Surrogate Users (*RQ3*).

For example when training astronauts, the problem of training in space is solved by training them on earth but in water tanks. This simulates how it is to maneuver in a large bulky suit in an environment that has them floating around instead of standing on firm ground. Thus the original problem is mapped to a problem that is easier to comprehend and less resource demanding, yet still preparing the astronauts for the original environment.

7

Development Method

Chapter 6 described how we intended to investigate our research questions through an experiment together with the Department of Urology at Aalborg Hospital. In this chapter we describe the development method that we used throughout this experiment, and how it focused on enhancing innovative thinking through the use of the Essence framework and prototyping. Furthermore, Section 7.2 gives an overview of the experiment, describing when the prototypes were built and how they were tested using Surrogate Users.

The overall goal of this experiment was to develop an idea that solves the Department of Urology's problem of being able to markup tasks during surgeries (described in Chapter 2), and to prove this idea to the department through a prototype. The prototype should then act as a basis for decision for *a*) the department to help them decide whether or not they want to further develop the suggested solution; or *b*) evaluate if there is a foundation for establishing a company.

While a single idea might solve the problem in terms of identified requirements, it is not necessarily the best possible solution because requirements can often be met in different ways. To accommodate this, the development method focuses on exploring the problem domain and the use of prototypes in order to generate, evaluate and mature as many ideas as possible, hopefully finding the best possible solution.

This is in contrast to typical software development processes where the focus is mostly on efficiency in order to meet requirements and deadlines, be they traditional, agile or in-between.

7.1 Essence

As mentioned, we incorporated elements from the Essence framework to encourage innovative thinking on the team. To further improve this, we added prototyping to our process together with the use of a Surrogate Challenger and Surrogate Users. This was necessary because the Department of Urology was not able to provide the resources required to have a real challenger and a number of users available for testing, whenever we had a prototype that needed to

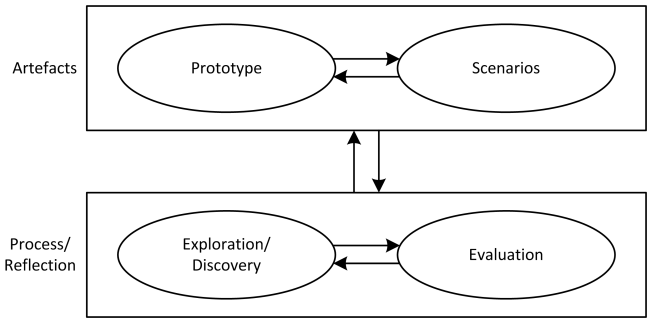


Figure 7.1: Our development method showing interaction between the artefacts and the process. The process creates artefacts and these artefacts inspire reflection-activities. Figure is based on discussions with Ivan Aaen.

be tested. The Surrogate Challenger role is further described in Section 7.1.2, and the Surrogate User role is described in Section 7.2.2.

To some extent we acknowledges Buxton's desire for a *design phase* in the early stages of the process (see Section 5.2), but we do not strictly distinguish between phases. Generally, there will be more focus on design early in the project than later, and given the scope of this project, we were primarily working in the design phase as Buxton describes it. In the design phase, there will be focus on generating, exploring, maturing and evaluating ideas, in order to, first, find out whether or not it is possible to solve the problem, and second, to find the best possible solution that fits into their daily work routine. While Buxton also talks about the design phase in terms of designing/planning the engineering process, marketing plan and business model etc., we strictly focused on designing the best possible solution.

Our development method is illustrated on Figure 7.1 and is to be understood as a process with constant interaction between the artefacts and the process, and between the elements in each. The process creates artefacts and these artefacts inspire reflection-activities.

For example, during exploration (e.g. discussions between the Responders or with the customer), scenarios were described and we gained insight into the problem domain. Ideas came up and were further evaluated in order to decide whether or not to continue working with them and manifest them as a prototype or part of one. The manifested idea is then evaluated by e.g. the Responders, the Surrogate Challenger, or through tests with the Surrogate Users. With the new prototype, exploration continues and new ideas or modifications might be discovered.

While this sounds like a iterative process, it is important to note that it is not. It is not iterative in the way that a cycle is completed and then the next cycle builds upon the previous cycle. It should be thought of as a dynamic and flexible process that acknowledges changes in the context as long as they are relevant to the project. Whether building the prototype, describing a scenario or evaluating ideas, new ideas can arise that might be superior to an existing idea (evaluation), scenarios can come up that initially was not thought about or an existing scenario can be changed etc. A new idea might add a prototype to the artefacts and modify or discard an existing.

How does this prototyping oriented approach fit into Essence's concepts of views? We find that it fits very well because in Essence use the Views to organize ideas and the project as a whole. Recall from Section 4.2 that a view is a physical location, e.g. a smart board, where information is stored for overview and communication between the actors. For example, during the process in Figure 7.1, ideas are noted on the Paradigm view and are further evaluated on the Process view, and at some point chosen or rejected (this process it not necessarily in this order, it is more non-sequential and random). It might also be concluded that an idea needs to be researched further, and depending on the case, a quick throwaway prototype can be created to test a implementation of some technology to get an idea of whether or not it is realistic to continue with the idea. Details about researched technologies and what they afford are stored on the Product View, which again might generate new ideas. The Product View is also used for how potential solutions might be implemented and, again, inspiring the responders for alternative ideas on how a product could be implemented differently etc.

7.1.1 Prototyping

While the views organize the ideas and the project as a whole, prototypes manifest these ideas supporting idea generation (see Section 5.2) together with the views.

However, it is very important to carefully consider how the prototypes are manifested. When building a prototype, it should be taken into account what the purpose is. If the purpose is to support exploration of the design space and idea generation, it is important that the prototype is not complete as this leaves little for imagination and thus hinders potential ideas that could have come up.

To enhance the prototypes' usefulness in design space exploration and idea generation, we used Lim et al. way of thinking (and describing) about prototypes as filters and manifestations (see Section 5.2.1). In our design process we used the prototypes, not solely for proving solutions, but primarily for discovering problems and exploring new solution directions. We did this when we created a prototype to perform the markup with pedals and a monitor for tracking the process (to see which step is currently being recorded, etc.). Using Lim et al. manifestation dimensions (see Table 5.2 on page 44) to describe the prototype, we chose to create a C# application displayed on a monitor (*Material*), with the means of testing the feedback (*Resolution*) given on the monitor when pressing the pedals (*Scope*). Using their filtering dimensions (see Table 5.1 on page 43, we designed the interface to be easy to see and recognize (*Appearance*), also from a distance (*Spatial structure*), and navigation and actions was performed by pressing different pedals (*Functionality*).

Recall from Section 5.1 that we, traditionally, have different ways of using prototypes through the process, in terms of identifying, clarifying, and adapting changed requirements. Through our process, we distinguished between exploratory/experimental prototypes (throwaway prototypes) and evolutionary prototypes.

Throwaway prototypes were used to either *a*) manifest an idea or multiple ideas in order to explore, evaluate or mature them; or *b*) to research technologies, implementation techniques and details to explore affordance, performance, etc. While the throwaway prototypes most often were thrown away, some of

them, especially the ones used for researching, evolved into a more evolutionary prototype either directly or by copying source code. This evolutionary prototype was then used throughout the process as ideas came up and were added to the prototype. Meanwhile, other prototypes were created to test and explore ideas or technologies before being added to the more complete and evolutionary prototype.

For example, two throwaway prototypes were used to investigate which speech recognition framework we should use to develop the Speech-to-Text Prototype. One prototype explored the *Microsoft Speech API* (SAPI), while another was used to explore *CMU Sphinx—Open Source Toolkit For Speech Recognition*. The CMU Sphinx prototype evolved, more or less, into an evolutionary prototype (after a few prototypes had been thrown away) and ended up as the "final" prototype that was used to prove our idea to the Department of Urology.

7.1.2 Roles

Because of the lack on an on-site customer, the Challenger role in Essence could not be fully filled out, and so we decided to adjust the responsibilities of the Challenger role and make use of a *Surrogate Challenger*. We chose our supervisor, Ivan Aaen, as our Surrogate Challenger, because he has some medico-technical background combined with knowledge on relevant technologies, process improvement, and system development. It is important to be aware about the fact that the Surrogate Challenger does not have the same capabilities as the Challenger.

The limitation of the Challenger role was mostly seen in the Project View, where the project planning and main vision (Toulmin Structure) was accomplished by the Responders instead of the Challenger. Scenario development was primarily done by the Responders as well, but to some extent in cooperation with the surgical staff at the Department of Urology (during short meetings). Unlike a real Challenger, the Surrogate Challenger was not present on a daily basis but rather on demand, as he could be contacted whenever the Responders needed input for some issue.

Because of the small team (two Responders and one Surrogate Challenger), the Anchor role was not used as intended during the process, and the responsibilities of the role was divided between the two Responders. As many of the discussions were between the two Responders, having one of them acting as an Anchor would limit the discussion possibilities severely as the Anchor is supposed to motivate and moderate the discussion more than engage in it. Thus the Responder role is extended to take all of the responsibilities of the Anchor, along with some of the responsibilities of the Challenger role.

Without a real Challenger to take on the role of Child once in a while, some of the discussions in the Paradigm View are somewhat limited due to the lack of domain knowledge that a real Challenger could have provided. However, because we engaged in meetings with the Department of Urology and visited the operating theater, we gained some knowledge that enhanced the Child role in the Responders and Surrogate Challenger. But because of the small team, it did not make sense to distinguish the child role from the other roles. We were however able reflect and relate ideas and prototypes to the operating environment where the solution was to be installed. Knowledge that only a real Challenger could provide if it was not for our observations.

Figure 7.2: Software Innovation Research Laboratorium (SIRL) during problem exploration and prototype development

7.1.3 Software Innovation Research Laboratorium

The Software Innovation Research Laboratorium (SIRL) is located at the Department of Computer Science, Aalborg University. The lab is designed to support Essence's idea of having the four Views as a physical location that are easily accessible. Using a specific physical location for each View is important because it will be easier for the actors to single out a particular viewpoint during discussions, to keep their focus, and to complete the objectives inherent to the View in question [Aaen, 2012]. Furthermore the views supports *Overview*, *Separation*, *Combination* and *Collaboration* as described in Section 4.2.

During the project, we were situated in the SIRL and we used its facilities throughout the project, both when exploring the problem domain, generating and evaluating ideas and when building the prototypes. Each View is displayed on a smart board which we used whenever we discovered and discussed ideas, planned the project or while developing the prototypes (noting state and class diagrams etc.)

Our tests were also conducted in the SIRL but in this situation, the smart boards were used for introducing the problem to the Surrogate Users (by a PowerPoint slideshow), while the Essence Views and or details were hidden.

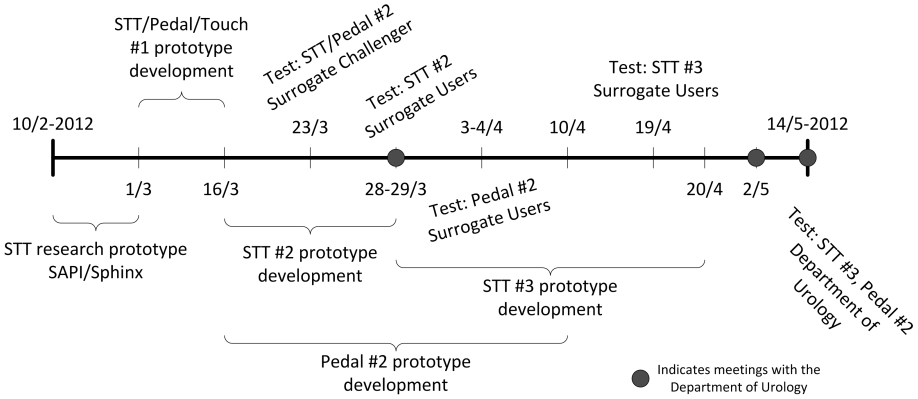


Figure 7.3: Timeline showing major milestones during our project course.

7.2 Project Course

This section describes the major milestones during our project; what and when the prototypes were created and tested with both Surrogate Challenger and Surrogate Users and finally shown to the Department of Urology. While this chapter gives an overview of the project course, Chapter 8–13 gives a more thorough description of each prototype, such as the ideas generated, how they were manifested in the prototype, evaluated, matured and finally how the prototype were tested. Testing the prototypes was different than on most project because the was performed with Surrogate Users instead of the actual users. This method of testing is further described in Section 7.2.2.

- *10/2–1/3 2012*: Speech recognition frameworks were researched and prototypes developed to further test and explore the technologies *Microsoft Speech API* (SAPI) and *CMU Sphinx—Open Source Toolkit For Speech Recognition*. SAPI is discarded and the CMU Sphinx prototype evolves into the prototype *Speech-to-Text Prototype 1* (STT). Because this research is strongly connected to STT Prototype 1, it is described together with this prototype in Chapter 8.

- *1/3–16/3 2012*: Development of the three prototypes STT Prototype 1, Pedal Prototype 1 and Touchscreen Prototype 1 was done simultaneously by the two Responders. While building the prototypes, the problem domain was explored, ideas were generated, evaluated and matured, and meetings took place with the Surrogate Challenger.

  - *STT Prototype 1*: After choosing CMU Sphinx as the speech recognition framework, voice commands were designed and discussed, and accuracy was tested and improved. See Chapter 8.
  - *Pedal Prototype 1*: Initial functionality implemented with focus on the visual feedback. See Chapter 9.
  - *Touchscreen Prototype 1*: Design based on Pedal Prototype, revised for touchscreen input. See Chapter 10

- *16–28/3 2012*: Development of STT Prototype 2. Implementation of states and complete redesign of commands. See Chapter 11
- *16/3–10/4 2012*: Development of Pedal Prototype 2. Implementation of states and revised visual feedback. See Chapter 12.
- *23/3 2012*: Testing of STT Prototype 2 and Pedal Prototype 2 with the Surrogate Challenger:
  - *STT Prototype 2*: Test of the prototype. Test of test setup in preparation for test with Surrogate Users. See Section 11.2
  - *Pedal Prototype 2*:
- *28–29/3 2012*: Meeting with the Department of Urology and testing of STT Prototype 2 with Surrogate Users
  - *Meeting*: Johan Poulsen was generally positive about the current status of the prototypes and the scenarios we set up. A short description of the meeting can be seen in Section 7.2.1.
  - *Test*: Test of the prototype. Successful test, only minor mistakes with one command. See Chapter 12.
- *29/3–20/4 2012*: Development of STT Prototype 3. Revised commands and audio feedback. See Chapter 13.
- *3–4/4 2012*: Testing of Pedal Prototype 2 with Surrogate Users. Test of the prototype. Several missed recognitions, otherwise a very successful test. See Section 11.2.
- *19/4 2012*: Testing of STT Prototype 3 with Surrogate Users. Successful test, only few missed recognitions. See Section 13.2.
- *2/5 2012*: Short meeting with the Department of Urology in preparations of the demonstration and tests of the prototypes on the 14th of May 2012. See Section 7.2.1.
- *14/5 2012*: Demonstration and testing of STT Prototype 3 and Pedal Prototype 2 with the Department of Urology. The tests were a great success and they were very impressed. However, there was some noise issues in the operating theatre. See Chapter 14.

7.2.1 Meetings with the Department of Urology

At the meeting on the 28th of March with Johan Poulsen and Jane Petersson, we discussed several aspects about the prototypes. First of all, they both agreed that the touchscreen near the surgeon assistant is most likely a solution with too many issues; especially placement of the touchscreen to make it easily reachable without being in the way. For the STT Prototype we discussed the idea of using a "step x" command, and Johan found it to be a very good idea, and that he believed this command would be used a lot more than Next and Previous. We played some of the audio feedback for him to hear his opinion on them, and he found that most of them are too long and too slow, and that they should be made shorter or removed, especially as they get used to the system.

For the Pedal Prototype we discussed the prospect of using pedals as a backup solution for the STT solution, such that the surgeon assistant can take over the markup task when the surgeon is too busy. We discussed the current functionality and they suggested that when marking the stop of a step, the next step should be selected automatically

Lastly we discussed further meetings this semestre, especially a meeting where the prototypes can be tested by the surgical staff. We agreed to have another meeting on the 2nd of May and bring the prototypes for test on the 14th of May where Johan Poulsen, Grazvydas Tuckus and Jane Petersson teached other surgeons how to perform RAS.

At the meeting on the 2nd of May, only Jane Petersson was present. We visited her in the operating theatre during a surgery with Grazvydas Tuckus as surgeon. The foundation of this meeting was to prepare for the demonstration and tests on the 14th of May and get answers for our questions in regards to this preparation. Our first question regarded the surgeries performed on the 14th of May in order to get a surgical step order, this was however not possible as the test surgeries are often made up just before starting, and depends on what the surgeons wants to train.

We were shown the operating theatre at the animal facilities of Aalborg Sygehus and thereby gained an idea of where to place our prototypes and how to conduct the tests.

7.2.2 Testing with Surrogate Users

The Department of Urology's limit in resources was not only seen in the lack of the Challenger role and the few meetings (only two during the project course as seen on Figure 7.3), but also in the ability to test the prototypes and solutions in an environment as close as possible to the operating theatre.

It is unrealistic to test every prototype during a real surgery because the focus needs to be on the patient, and errors in the prototype would most likely remove this focus. A more realistic scenario would be if we could have used the operating theatre used for training surgeons in robotically-assisted surgeries, where surgeries are performed on pigs instead of humans (like in the final test, described in Chapter 13. However, these test surgeries performed on pigs are still expensive, and furthermore in order to perform the best possible test, we still need surgery personnel who again are too expensive not to use for real surgeries.

We believe that not being able to perform frequent tests is a common problem, and that the feedback the tests give is very important in order to deliver the best possible solution. Thus instead of ignoring tests before the final test with the Department of Urology, we performed frequent tests with Surrogate Users. Surrogate Users do not necessarily share any experience or knowledge with the real users of the system. Using Surrogate Users obviously would not give as precise results as if it was the real users, but we believed that we could minimize the difference by mapping the problems from the real domain to problems that are easier to handle by the Surrogate Users.

But first we considered what part of the solution that we wanted to test with the Surrogate Users. We tested the following aspects of ideas manifested as prototypes, to figure out whether or not the prototype needed to be discarded, modified or extended with further functionality:

- *Technical*: Performance; speech recognition accuracy;
- *Usability*: Ergonomic pedals; voice commands; feedback; easy/intuitive to use

While being able to test some technical and usability aspects of the prototypes, the following lists some of the problems that makes the tests less accurate:

- Environment: Given the nature of surgeries, the focus of the surgical staff needs to be 100% on the patient, and not on the software to markup the surgery. This is also hard to simulate on surgeries performed on pigs, because it will never be as critical as if it was on a human patient.

  It is even harder to simulate stressed situations emerged through a surgery if something does not go as planned. In this case the surgical staff needs to be even more focused on the patient, and will most likely ignore everything else.

- Integration: Existing systems, such as the *da Vinci*® Surgical System is not available outside the operating theater because they are very expensive. Thus it is impossible to test the features needed from the system, such as video feed and microphone.

In order to minimize the difference between the real environment in the operating theatre and the test environment with the Surrogate Users, we used the concept of mapping the mental state of being concentrated and focused on the patient, to problems that made the Surrogate User focus on something very specific while performing a markup. For example, in case of the STT Prototype the Surrogate User was supposed to be focused on drawing dot-to-dot drawings with high precision while using the prototype to markup when a drawing was started and finished (see Section 11.2.2 for a more thorough description).

Thus we were able to test the prototype while the user was focused on solving a task that required high attention. It is obviously not as attention-demanding as a real surgery, but it moves some attention from the prototype to the task to be solved, thus lessening the difference in the environment to some extent.

The problem that we did not have access to the *da Vinci*® Surgical System with such an essential feature as the video feed was solved by abstracting away from the video feed in these prototypes, and simulating that we "recorded" the video by storing a timestamp of each markup. We did however ensure us that it was possible to get access to the video feed from the robot, by talking to one of the engineers at Aalborg Hospital.

The microphone however, was not as accessible as the video feed so this remains uncertain. It was however discussed with the chief surgeon (who operates the *da Vinci*® Surgical System), that it would not be a problem to wear a small headset while seated at the console, or to install an external microphone in the console if this was technically feasible.

7.3 Documenting the Process

The usual process of designing software appears quite irrational, where we start without a clear statement of desired behaviour and implementation constraints. A long sequence of design decisions is made with no clear statements of why

it is done. However, when presenting systems to others, it is often described as a process pretending that the design and development was carried out as a rational process. Parnas and Clements say:

> *"... the picture of the software designer deriving his design in a rational, error-free, way from a statement of requirements is quite unrealistic. No system has ever been developed in that way, and probaly none ever will. Even the small program developments shown in textbooks and papers are unreal. They have been revised and polished until the author has shown us what he wishes he had done, not what actually did happen."* [Parnas and Clements, 1985]

We acknowledge this way of describing the process as being rational, however it is not suitable for this project where we wish to describe the actual process; the thoughts and actions behind the decisions made, and how Essence supported this process.

Recall from Section 5.2 that design activities are *flexible rather than rigid* and *reflective rather than prescriptive* [Schön, 1982], and Schöns description of design as *reflective conversations* [Schön, 1992]. These kinds of processes are difficult to describe because they seems to be, not random, but chaotic at times. There are lots of interleaving in the process, which can also be seen in the Views, where things are noted as they are discovered by the designers.

Such a concurrent and highly interactive process is difficult to describe clearly in a linear text. To offset this issue somewhat, we decided to document our process in a chronological order where the first version of each prototype are described before the revised versions. Although each prototype is very different in terms of interface, they build upon the same principals, which the chapters reflects in their chronological order. For example, in version two of the STT Prototype we introduced the concept of states. This was also introduced in version two of the Pedal Prototype, right after implementing and testing it in the version two STT Prototype.

Chapter 8–13, in that order, gives a better overview of the design process, while reading the chapters in a sequential order with respect to prototype and version might give a better overview of each prototype and how it evolved during our process. We suggest reading the chapters in the order 8–13 because it is the process that we are primarily interested in documenting. Each chapter describes a prototype and its design process, in the following way:

- *Functionality*: Overall functionality of the resulting prototype, giving the reader an overview
- *Test*: Setup, description and results of how the prototype was tested using the Surrogate Challenger and/or Surrogate Users
- *Essence*: Description of the design process, documenting how the prototype evolved into the one described under Functionality. This includes exploration, evaluation and discussions between developers, with the Surrogate Challenger and meetings with the real Challenger. The Paradigm View acts as the underlying basis of the description because it is here exploration and discussion takes place. In order to indicate that a discussion affects another View, we use the following "callout":

Process View: Example

Content added on the process view, or a summary with a reference to the actual content added.

Hopefully, this way of describing each prototype gives the reader some insight into how the design process progressed through the project.

8

Speech to Text Prototype 1

During surgeries the surgeon is the one with the most experience and knowledge about the operation and he obviously knows the current status of the operation. This makes him an ideal candidate for performing the markup during the operation because he will always be the first to know when the operation moves from one step in the process to another, thus making the markup more precise than if someone else was to conduct the markup. However, during robotically-assisted surgeries, the surgeon is seated in a console (see Figure 8.1), operating the robot with both hands and feet, making the surgeon unable to interact with pedals or touch screens to perform the markup.

We tried to solve this problem by building and testing a voice recognition prototype that translates speech from the surgeon (or other possible candidates if needed) into text (thus the name, Speech to Text (STT) Prototype), which are then inspected and acted upon in a Java program, marking up the process changes.

This chapter describes version one of our Speech to Text Prototype, where we primarily focused on exploring whether voice recognition is usable in terms of accuracy and thereby usability, and thus the following:

- Explored and tested different frameworks for recognizing voice commands given by the user
- Improved accuracy of the chosen voice recognition framework
- How to design the voice commands for better accuracy
- How to give the user proper audio feedback making sure that he always know what the software is doing

The *da Vinci*® Surgical System provides a video feed of the cameras connected to the robot (controlled by the surgeon from the console), and when a step has been marked up, the video recordings from that step should be stored. Thus when the surgeon marks the start of a step, this could be seen as a start of a *recording*, and subsequently a stop of the recording when he marks the end of the step. The terms markup and recording of a step will be used interchangeably through the rest of this chapter.

Figure 8.1: The surgeon seated at the robot console during a surgery. Note that both hands and feet are occupied.

8.1 Functionality

The overall goal of the prototype is to make the surgeon able to markup when he moves from one step in the surgery process to another (see Section 2.1.2.1 for a description of a surgery process). Thus the most basic functionality that the surgeon needs are:

- Navigate between steps in the process
- Start and stop a recording of the step
- Cancel a start of the current recording
- Cancel a stop of the last recording

The functionality above was translated into voice commands, which could then be spoken to the computer through a microphone. Each step in the process was given a name corresponding to what was to be performed in that step, so by saying the name the computer would *select* the step. When a step is selected, it is possible to start a recording of that step. The recording is automatically stopped when the user navigates to another step and starts a new recording. To ensure that the user does not speak a command by accident while talking freely to other people around him, it was decided to prefix each command with the word *computer*.

The available voice commands can be seen on Table 8.1. The commands does not map to the actual steps in the surgery process, but to the steps that would be performed while testing the prototype. The test and steps to be performed are described in Section 8.2.

The prototype did not actually record anything, but saved a timestamp when the user started and stopped a recording. These timestamps could be used to split up the recorded video from a video feed, but this was not considered for the prototype. The focus was solely on exploring whether or not voice recognition was a possible solution in terms of accuracy and usability.

Functionality	**Command**
Select step	Sy et og to sammen (stitch together 1 and 2)
	Bind blodknude (tie a barrel knot)
	Sy tre og fire sammen (stitch together 3 and 4)
	Bind pælestik (tie a bowline)
	Sy fem og seks sammen (stitck together 5 and 6)
Start and stop a recording of the selected step	Start
Cancels the start and/or stop the recording	Annullér (cancel)
Terminates the markup of the surgery	Afslut (exit)

Table 8.1: Voice commands in danish with translations, without the *computer* prefix

8.2 Test

No systematic external tests were conducted for this prototype, but we developed the test setup that was used, with modifications, in the later STT Prototypes. This section describes the test setup and the tasks that were to be completed by the test subjects in order to create a process that needs to be marked up using the prototype. Finally we describe how we tested the prototype ourselves and discussed it with the Surrogate Challenger.

8.2.1 Setup and markup tasks

As mentioned above, the target user of the prototype is the surgeon, who should be able to perform the markup during surgeries while seated in the console of the *da Vinci*® Surgical System. During prosthetic removal surgeries, the surgeon needs to perform a series of tasks, such as exposure of the prostates back and removal of the lymph nodes (see the full list in Section 2.1.2.1).

We tried to map the tasks that the surgeon is performing to tasks that are solvable by Surrogate Users (the test subjects), in an environment that are much more available and where we can easily conduct tests of the prototypes. There are several important factors to take into account when mapping these problems, some of them being:

- The test setup should approach the environment of the surgeon while performing surgeries
  - The test subject should be seated in a position similar to that of the surgeon
  - The test subject should be occupied with both hands, seeing his actions only through a camera
  - The tasks should be somewhat similar to that of a surgeon performing surgeries

  - The tasks should be challenging in respect to spatial awareness, meaning that the user needs to focus on where the tools are in respect to each other (this is especially hard for the surgeon because he does not have the tactile feedback in his hands when operating using the robot compared to traditional manual surgery)

- The focus of the test subject should be on solving the task with the uttermost precision
  - The test subject must not be distracted in solving the task while performing the markup

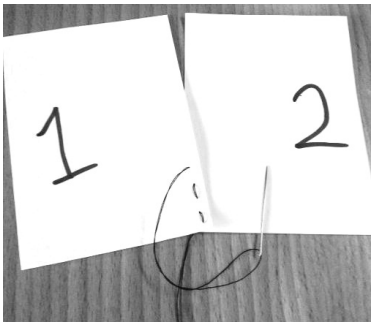
To create a test setup where the test subject should solve tasks using both hands, without being able to see his hands directly, we used a closed cardboard box mounted with a camera. The cardboard box was mounted on a table and a hole was cut to the test subjects hands. A monitor was placed behind the cardboard box, showing the image of the camera, thus displaying the hands of the test subject during tests. The setup can be seen on Figure 8.2.

The camera was mounted at the top back center inside the cardboard box thus skewing the image. The spatial awareness that human beings are normally very good at, became significantly more difficult because of the skewed image of the test subjects hands. This made it more difficult to locate and interact with objects, forcing the test subject to be more focused on solving the task with high precision.

The tasks to be solved and marked up by the test subjects, should make sure that; *a*) both of their hands are occupied; *b*) focus and concentration is needed to obtain high precision; and *c*) tasks are similar to the ones performed by a surgeon during surgeries. To solve these requirements we came up with the task of stitching together two pieces of paper; a task that is typically used to train surgeons[1]. To solve this task the test subject would need to hold the paper in

[1] It was confirmed by staff specialist Grazvydas Tuckus during the first meeting at Aalborg

Figure 8.2: The STT Prototype test setup



(a) Stitching two pieces of paper together

(b) Test subject stitching and marking up the task using the STT Prototype

Figure 8.3: The stitching task to be solved and marked up by the test subjects

one hand and the sewing needle in the other. The task should be solved inside the cardboard box showed on Figure 8.2, where the camera displays a skewed image of the test subjects hands solving the task. This makes it significantly more difficult to stitch the two pieces of paper together with a decent precision, and requires the test subject to be very focused on the camera. The stitching task can be seen on Figure 8.3.

Trying to create a little diversity in the process, we created a second task to be solved by the test subject. The idea was that the test subject should tie different knots inside the cardboard box as well. Like stitching, both hands are required and because of the skewed image this becomes a more difficult task that requires more concentration from the test subject.

The test process consisted of five different tasks which were to be solved and marked up by the test subject. When the test subject started on solving a task, he should mark that using the STT Prototype by navigating to the task and start the recording. When the task was solved, he should navigate to the next task and start the recording of that (thus stopping the previous recording). This continues until all tasks has been solved, and thereby marked up. The process was as follows:

1. Stitch together paper 1 and 2
2. Tie a barrel knot
3. Stitch together paper 3 and 4
4. Tie a bowline

Hospital, that they used the task of stitching paper together in a surgeon training program.

5. Stitch together paper 5 and 6

Figure 8.3b displays a user solving task 1, stitching together paper 1 and 2 and marking up the task using the STT Prototype through the headset. The tasks of tying a knot are similar to that on the figure, just with a single thick cord.

8.2.2 Results

As mentioned earlier, no tests were actually conducted for this version of the prototype. Instead we tested it ourselves and discussed it with our Surrogate Challenger, by which we found weaknesses in both the test setup and the prototype itself, and so it was decided not to test it on actual test subjects before version two of the prototype. We performed some tests of the test setup in order to make sure that it actually met the requirements described in Section 8.2.1, and that we could use it for testing version 2 and 3 of the prototype with our test subjects.

We found that the setup itself, displayed on Figure 8.2, fulfilled the requirements set. The closed cardboard box made sure that the user could not see his hands directly, except through the camera. And because the camera was skewed, it became significantly more difficult to manipulate objects within the box, forcing the user to concentrate more than he would normally do.

However, we discovered some problems with the tasks that the test subjects were to solve during the tests. The task of stitching together two pieces of paper was too easy to perform because it could be done almost exclusively by feeling one's way, thus removing some concentration and focus from the camera. The surgeon does not have this feeling when using the robot as there is no such feedback. The same problem occurred with the task of tying knots, but to a little lesser extent because the user still had to focus on the camera to make sure that the knots were tied the correct way. Besides looking at the camera, the user also had to look at a paper displaying how to tie the knot, where the surgeon is exclusively focused on the video feed from the robots cameras. We decided to abandon these tests as they were not as similar to the surgeon's tasks as we expected them to be.

To solve this problem we introduced the task of drawing dot-to-dot drawings, which are described in version two of the prototype in Chapter 11.

8.3 Essence

In this section we will describe our process and how we explored and matured ideas for version one of the STT Prototype. The description will be based on the paradigm view and as ideas are discovered and evaluated, the related views will be updated accordingly. Most of the ideas will originate from usage scenarios that we have discussed with the surgical staff, Surrogate Challenger and ourselves as new ideas have arised.

Because this chapter focuses on the STT Prototype, we start out with the most basic usage scenario seen from the surgeon's point of view:

> **Scenario: Surgeon performing markup during operation**
> *The surgeon wants to markup every time he starts and stops a step in the surgery process while performing the surgery using the da Vinci® Surgical System.*

Based on this scenario, several ideas were developed and matured, each described below:

The markup tool to be used to create the markup during operations could be either *a*) touchscreen; *b*) pedals; or *c*) microphone. Touchscreens would allow for displaying information to the surgeon while performing the markup to ensure that the correct step is being marked. This also applies to solutions with pedals or microphone if they have an accompanying monitor for feedback.

However, given the scenario and what we saw when we visited the operating room during surgeries, it is clear that both the touchscreen and pedals are not a viable markup tool for the surgeon, because both hands and feet are occupied operating the *da Vinci®* Surgical System from the console. Monitors for feedback were generally discarded because the surgeon needs to be 100% focused on the video feed from the robot, displaying his actions while using the robot, and feedback from the *da Vinci®* Surgical System. It was therefore decided to work with the idea of performing the markup using voice recognition through a microphone placed inside the console, without any visual feedback.

Process View: Idea Evaluation

Idea:	**Touchscreen or pedals as the markup tool**
Pros:	High/descriptive feedback, text describing each step in the process.
Cons:	The surgeon would have to remove himself from the console in order to operate the touchscreen or pedals.
Idea:	**Microphone as the markup tool**
Pros:	Surgeon can remain seated at the console. Not occupying hands, feet or taking any vision.
Cons:	Immature technology. No visual feedback.

Product View: Speech recognition framework

We created two prototypes while researching the speech recognition frameworks Microsoft SAPI and CMU Sphinx. We chose to discard Microsoft SAPI and continue working with CMU Sphinx for future prototypes. See Section 8.3.1.1 for a complete description of the evaluation.

Prefix voice commands to make sure that the voice recognition software does not recognize voices that was not intended for the computer. We decided for our prototypes to use the word *computer* when addressing commands to the computer. This word can then later be changed if the surgeon wants it to be different.

Another scenario concerns the order of the steps in the surgery process, which has been discussed with the surgeon:

> **Scenario: Surgeons perform surgical steps in different orders**
> *Depending on the surgeon performing the operation, the step order might differ from other surgeons. The step order depends on surgeon experience and the patient, and it happens that during the operation the surgeon needs to take a different step than what he would normally do (sequentially).*

Selecting the step to markup is necessary because the surgeon does not necessarily perform the steps sequentially. We first tried to map the description of each step to a voice command, so when the surgeon wants to select step 7, he would say *Exposure of the prostates sides*. We quickly discovered that it was significantly more difficult for the voice recognition software to recognize longer and more complex commands, while it was easier for it to recognize the shorter commands with just a few words.

This was confirmed through our tests where we had mapped the problem to the tasks of stitching and tieing knots, as described in Section 8.2.1 (the voice commands can be seen on Table 8.1). Furthermore when using the longer commands for selecting tasks, it often misunderstood or confused the commands because they were very similar. For example the command *computer sy tre og fire sammen* (stitch three and four together) would often be recognized as *computer sy fem og seks sammen* (stitch five and six together).

Process View: Idea Evaluation

Idea:	**Selecting the step to markup using the description of each step**
Pros:	Easy to remember the commands. No doubt about what the selected step covers in the procedure, and what is to be done in that step.
Cons:	Higher chance of misrecognitions and misunderstandings compared to shorter commands.

Cancellation of commands in case the surgeon misspeaks or if the system misrecognizes an intended command. We worked with two ideas; one where the user simply says the word *annullér* (cancel) to cancel the last spoken command and, one where the last spoken command is cancelled if a new command is spoken within x seconds (*cancellation time*). The time based approach was not considered for this version of the prototype.

While exploring the command based idea, we discussed what should happen when a start-markup is to be cancelled. Three cases were to consider;

- The time passed from stop-markup to start-markup (the command to be cancelled) should be discarded;
- added to the previous step; or
- added to the next step.

If the system recognizes a start-markup not intended by the surgeon while he is still in the middle of a step, the time should be added to the previous step

(the one that the surgeon is currently working on). If the surgeon proceeds to the next surgery step, and marks up the start of a different step than the intended one, the time passed should be added to the correct step, assuming that the surgeon continues working while selecting the correct step. Whether or not we should make the software add the time to the previous or next step will be determined during tests to see if the system often misrecognizes commands and if such a feature is needed, or if it is okay to just discard the time. Thus version 1 of the prototype will just discard the time.

Process View: Idea Evaluation

Idea:	**Cancellation as a command**
Pros:	Always possible to cancel the last spoken command.
Cons:	Risk of canceling a command unententionally, which could lead to a possible loss of markup.
Idea:	**Time based cancellation**
Pros:	No ekstra commands needed, the system determines if the last spoken command is to be cancelled.
Cons:	Impossible to cancel the last spoken command if cancellation time has passed. Impossible to perform the markup of a step which lifetime is within cancellation time.

One major problem with this version of the prototype was that it was not possible to stop a recording. Once a recording was started, there was no way of stopping besides starting the recording of the next step. This works only if the surgeon does not interrupt (or gets interrupted) the surgery, but in a normal working environment we cannot assume the surgeon to never get interrupted.

We made a SWOT analysis of speech recognition in the Process View:

Process View: SWOT Analysis

Strengths:	Can be operated without occupying hands nor feet. Does not require visual focus. Precision can be improved through use of noise cancelling techniques.
Weaknesses:	Hard to implement. Speech recognition is far from fully matured. Lack of precision. No visual feedback.
Opportunities:	May be useful for other types of surgery. Lots of focus on speech recognition, the technology will surely mature over time.
Threats:	Dialects and noisy facilities may be problematic in regards to precision.

8.3.1 Product View

In the product view we explored *how* given features could be built; features that we discovered, matured and evaluated in the paradigm and process views,

Filtering	**Description**
Interactivity	Sound Input - User gives spoken commands. Output Command - system outputs a line describing which spoken command it recognized.
Manifestation	**Description**
Material	Java console application.
Resolution	Performance - Speech recognition time. Accuracy - Precision of recognition.
Scope	Speech recognition only.

Table 8.2: Filtering and Manifestation dimensions for STT Prototype 1

described in Section 8.3. The filtering and manifestation dimensions for this prototype is shown on Table 8.2.

Version 1 of the STT Prototype was primarily dedicated to exploring speech recognition frameworks, their usability and accuracy, and how we could build a solution around them that realized our ideas.

The following section describes our choice of a speech recognition framework and why we chose CMU Sphinx over SAPI.

8.3.1.1 Speech Recognition Frameworks

To support our ideas, the most basic requirements for the speech recognition frameworks are the ability to; *a*) recognize speech and translate it into a command (*command and control*); and *b*) define the words to recognize (*grammar*) and react upon.

Furthermore, we defined the following requirements which we used to evaluate the frameworks against in order to find the best suited framework:

- *Accuracy:* The higher speech recognition accuracy the better.
- *Documentation and ease of use:* Important because our focus is on building a functional prototype quickly that allows us to test the prototypes on external users.
- *Customization:* The ability to customize the framework in order to provide the best possible integration in the environment.
- *Multi-language support:* Surgical staff (surgeons and surgeon assistants) are of different nationalities, and we would like to support at least Danish and English speech recognition for better integration.

The description below of the evaluation will not go into the more technical details of the frameworks, as it is not the scope of this document. We did however evaluate and test it (including the technical feasibility) to ensure that we can recommend the chosen framework for the actual implementation.

Two prototypes were created to research the frameworks *CMU Sphinx—Open Source Toolkit For Speech Recognition*[2] and *Microsoft Speech API*[3] (SAPI)

We chose to research SAPI because we were both very familiar with the Microsoft Windows platform, the .NET Framework and the C# programming language. SAPI 5.4 ships with *Microsoft Windows 7* (all versions).

[2]CMU Sphinx: `http://cmusphinx.sourceforge.net/`
[3]SAPI: `http://msdn.microsoft.com/en-us/library/ee125663`

CMU Sphinx was chosen because it is a recognized open source speech recognition framework, used in software such as the Linux desktop manager *GNOME Desktop*. CMU Sphinx 4 is written entirely in the Java programming language, and created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP), with contributions from the University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT).

Microsoft SAPI

First we created a prototype to research SAPI, and we quickly discovered a lack of documentation to get developers easily started. The available documentation is located on Microsoft's Developer Network (MSDN), and is very sparse. It gives an overview of the API as a high-level interface between an application and the underlying speech engines (one for text-to-speech systems and one for speech recognizers). The following documentation then documents these C++ interfaces (no documentation for the C# equivalent) and their functions sparsely and with very few C++ examples of their use. Generally, we missed some more detailed examples of the API in-use.

Microsoft provides a range of language models that needs to be installed into the operating system by executing a packed installer executable file. One does not have full access to these models, and we did not find any information about how to, for example include phonemes in the acoustic model that was not included by default. Generally, the customizability of the framework was *very* limited, according to the documentation.

However, we found that we could customize the speech recognizers confidence levels for each word (to be recognized) in the grammar. The speech recognizer engine returns a confidence level for each word in the recognized phrase. The higher the confidence number, the better the match between what the speech recognizer engine heard and the engine's stored pronunciation. By adjusting these confidence levels one can increase the overall speech recognition precision. For example if the grammar contains the word "hello" as the only word, the engine would recognize words such as "fellow" and "yellow" as "hello". By adjusting the confidence level of "hello", the pronunciation needs to be close to "hello" in order for the word to be recognized.

CMU Sphinx

The second prototype was created to research CMU Sphinx. Being a open source framework, CMU Sphinx has some obvious advantages over Microsoft SAPI (e.g. source code and a engaged community). CMU Sphinx was originally build for research purpose as a framework where different parts could easily be modified in order to, for example, test the efficiency of a newly developed algorithm.

Compared to SAPI, CMU Sphinx is much better documented. It contains a complete API documentation (*Javadoc*), and lots of examples on how to use the framework. For example the source code downloaded contains a *doc* folder that includes different examples on how to use the framework. Furthermore their home page contains detailed information about research and theory on speech, speech recognition, language models and dictionaries (in the context of speech

recognition). They describe how to adapt existing acoustic models, how to train your own acoustic model and building a complete language model from scratch.

CMU Sphinx comes with an English/American language model by default, but others have been created by the community. However, there was no Danish language model available, but we became aware that such a model had been created on Aalborg University by Morten Højfeldt Rasmussen and Morten Thunberg Svendsen as their part of master thesis [Rasmussen and Svendsen, 2005]. With assistance from Morten, we adapted this model and used it through our prototypes.

Given that CMU Sphinx was built as a framework to support research, it is *very* customizable. Almost any property one can think of, can be tweaked through a XML configuration file. For example, we tweaked *Threshold* property of the *SpeechClassifier* which makes it possible to control when the recognizer begins recognizing by monitoring the volume level of the received input. This is used to filter out background noise. Like SAPI, it is also possible to tweak the confidence level of words in the grammar.

Because CMU Sphinx was so customizable, it was also more difficult to use and thus we spent more time on researching CMU Sphinx before we were able recognize and translate the commands into text that could be acted upon in a Java application. However, the extensive documentation and an active community helped us during this process, and so we decided to discard SAPI and continue using CMU Sphinx in future prototypes.

9

Pedal Prototype 1

As described in [Jakobsen and Follin, 2011], we followed the staff at Aalborg Hospital for a day and found that the surgeon assistant uses both hands very often to change robot equipment when the surgeon requests it. Furthermore, the assistant is also occupied rinsing and soaking up liquids during the surgery, giving the surgeon a better view of the organs. This task is shown on Figure 9.1 from the view of the surgeon assistant.

The assistant uses a nearby monitor to navigate inside the patient, showing the same view as the one given to the surgeon by the endoscopic cameras.

The surgeon assistant is often a surgeon or a nurse who has completed extra education on this type of surgeries, and therefore knows a lot about the surgery. This, along with the fact that the surgeon is in close communication with the surgeon assistant during the surgery, makes the surgeon assistant a good choice for handling the markup of the surgery.

Given the fact that the surgeon assistant is very occupied with both hands, we got the idea of performing the mark–up using foot pedals. The elements we focused on for this prototype was:

- General test of pedals as tool for marking up steps
- The use of colors to give a clear visual feedback when a pedal is pressed
- The ability to cancel a start or stop of a step in case a pedal is pressed by accident

Like with the STT Prototype, it is important that the prototype draws as little attention as possible. Thus the foot pedals needs to be placed in such a way that the surgeon assistant can find them using her feet without drawing too much attention to where they are located on the floor. Generally the assistant does not have much free space around her, however as seen on Figure 9.2, the surgeon assistant has some free space available where the foot pedals could be located.

Like the foot pedals should not draw attention, it is important that the visual feedback of the prototype does not either. To avoid this, we will try exploiting humans peripheral vision together with the use of color codes for clear indication of what the prototype is doing.

Figure 9.1: The surgeon assistant using equipment to soak up liquids during the surgery

Figure 9.2: The available space for pedals near the surgeon assistant

The staff in the operating theatre has to focus primarily on the patient to avoid errors while conducting the surgery. This makes visual feedback, e.g. using a touchscreen, quite problematic as it usually requires focus from the user to be seen. To minimize the amount of focus needed, we will try to exploit peripheral vision rather than requiring focus from the user. Peripheral vision has several limitations compared to fovea vision [Johnson, 2010]:

- The resolution is high in the fovea sight and very low in the peripheral sight
- Fovea sight is better at discriminating colors than peripheral sight
- Peripheral sight is good at registering movement and changes

In order to exploit the peripheral vision for e.g. a touch screen, the visual feedback must be large and clear so it is easy to see even in peripheral vision. Furthermore, we should be using very distinct colors to mark changes so they are easy to see. Lastly, applying some kind of movement on the screen when an action is taken should make it easy for the user to register the feedback given by the monitor.

9.1 Functionality

As with the Speech to Text Prototype, we have a list of functionality needed by the user:

- Navigation between steps in the process
- Mark start and stop of each step, one at a time
- Cancel the start markup of the current step
- Cancel the stop mark of the previous step

In order to present the steps and features to the user we chose to create one button for each step, each corresponding to a step performed by the surgeon during the surgery. Besides that we created a button for marking that the surgery has ended and a button for canceling the last start/stop command. The text of the button will describe which step/feature it represents and the border of the buttons will be colored as described above. In order to make the recording status even clearer to the user, the step being recorded is showing an animation to show that the tool is currently working. It is showing as a rotating arrow. The buttons were given a simple design with black text and white background for readability through high contrast. A screenshot of the first prototype can be seen on Figure 9.3.

The prototype uses a "selector", which is the step marked with a black border. This is used to signal which step will be started if the start pedal is pressed. When a pedal is pressed, a keyboard event is fired (each pedal is mapped to a key). The buttons in the prototype are activated through these key events.

The system was designed to be used with three pedals. The left pedal moves the "selector" to the previous button, the middle pedal activates the selected button and the right pedal moves the "selector" to the next button.

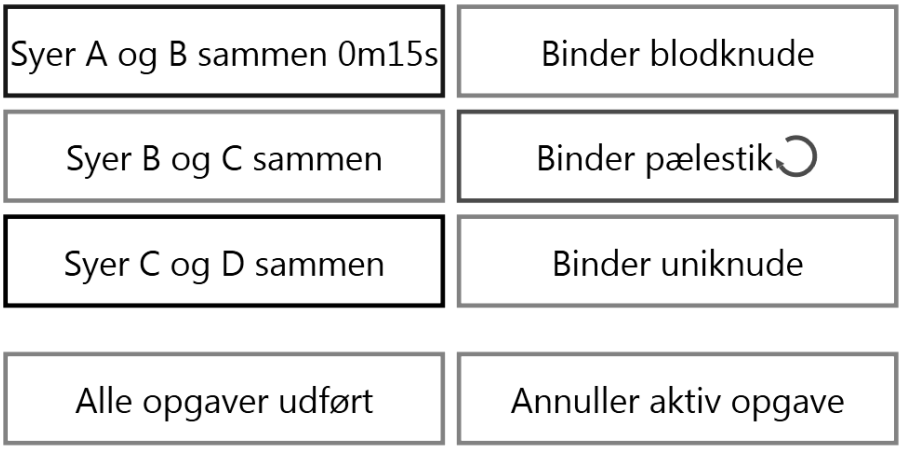


Figure 9.3: The first Pedal Prototype implementation, showing the visual feedback for the surgeon assistant. In this picture, the first button ("Syer A og B sammen 0m15s") is blue, the third button ("Syer C og D sammen") is black and the fifth button ("Binder pælestik") is red.

Starting a new step automatically stops the step currently being recorded. Furthermore, starting a step automatically moves the "selector" to the next step, e.g. starting step 1 will move the selector to step 2. If the order of surgery steps given to the prototype is followed completely, the user will be able to record the surgery by only using the middle pedal because of this simple interface.

Steps that have already been recorded cannot be chosen again and when the "selector" is moved to one such step, it will automatically move another step until it reaches a step that is either currently recording or has not been recorded yet. While this allows for faster navigation it also limits the recording of steps as it will not be possible to complete part of a step, stop it and then resume it at a later time.

9.1.1 Color codes

In order to see which steps have been recorded and which step is currently being recorded, we chose colors with inspiration from e.g. video recorders. The colors and their meaning are as follows:

- *Black*: The currently selected step/order
- *Red*: The step that is currently being marked up
- *Blue*: Steps that have been marked up
- *Gray*: The steps that have not yet been marked up

By marking the steps with these colors, we will exploit peripheral vision as described previously. An example of this is the selected step as it becomes red when the recording starts. The user should then be able to register that the recording has started without removing focus from the patient.

9.2 Test

No actual test was conducted for this prototype, however, our Surrogate Challenger tried out the prototype and quickly found several issues with the use of this prototype. While color codes allow the users to easily see if a step is being recorded, the colors were hard to see from distance because only the border of each button was colored. Using buttons to cancel and end the surgery was easy to use when there are 6 steps in the surgery but will most likely not scale well. Furthermore, the user will have to read the text on each button to find out which button to press for ending the surgery or cancelling the last start/stop. This became an issue as the button texts were not large enough to easily read from distance.

During the day where we followed the surgeons in the operating theater we also found that the surgeon may have to take a break, e.g. to consult another surgeon, and since this prototype does not support starting a step again after the first stop this can be problematic for the use of the prototype.

9.3 Essence

While developing and using this prototype we used the Views in Essence to contain the ideas, design and implementation of this prototype. This section describes the ideas we got through the process and therefore Paradigm View is the main view for this section.

The surgeon assistant is the only focus for this prototype and we have the following scenarios for the surgeon assistant during the surgery:

> **Scenario: Surgeon assistant performing markup during surgeries**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery.*

Our three ideas for performing the markup; microphone, touchscreen and pedals, are described in Chapter 8. However, where the microphone solution is the only solution for the surgeon because of the interface on the *da Vinci*® Surgical System robot, the surgeon assistant could potentially use all three. The surgeon assistant is required to wear a surgery mask for sterility issues which might be a problem for use of the microphone.

Process View: Idea Evaluation

Idea:	**Touchscreen or pedals as the markup tool**
Pros:	High/descriptive feedback, text describing each step in the process.
Cons:	The pedals and touchscreen have to be placed in a way such that they do not annoy the surgeon assistant while still being easy to reach when needed.
Idea:	**Microphone as the markup tool**
Pros:	Surgeon assistant can move around freely and always reach the markup tool interface if a wireless headset is used.
Cons:	Immature technology. No visual feedback. Surgery mask may cause sound problems.

As the microphone idea is already being evaluated for the surgeon, the main focus for our scenarios with the surgeon assistant will be on the pedal and touchscreen prototype.

Project View: Project Planning

Focus on touch screen and Pedal Prototypes while STT Prototype is being analysed and tested in the prototype for the surgeon.

On top of performing markup, the surgeon assistant might have to markup steps in a different order than originally expected.

> **Scenario: Surgeon assistant performing markup during operation, surgical step order is different from the expected order**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery. The surgeon chooses to do the surgical steps in a different order than usually, and therefore the surgeon assistant has to navigate between the steps to markup the correct steps at the correct time.*

We developed a two ideas that will make the staff able to cope with changing surgical step orders.

Use one pedal to mark start/stop of steps and the other two steps for selecting next and previous step in the list. This allows the surgical assistant to perform the markup during surgeries. Show the currently selected step using a visual effect.

Use a computer to change the order of steps. When the surgeon decides to abandon the normal order of steps, he can go to a nearby computer, edit the original step order to the order he is going to execute. The new list then shows up on the monitor, and the surgeon assistant can markup as usual.

The different ideas introduced was evaluated on the process view.

Process View: Idea Evaluation

Idea:	**Using pedals to select next/previous step in case the surgeons takes a non-linear order of steps**
Pros:	Possible to perform using only the pedals. Does not require a dedicated system.
Cons:	Increases the complexion of the Pedal Prototype.
Idea:	**Use a dedicated computer to change the order of steps**
Pros:	Makes the other systems less complex. Allows for great overview over the process.
Cons:	The surgeon or surgeon assistant needs to leave his work in order to operate the computer, or ask and wait for an assistant to do it.
Idea:	**Color codes to show status for steps**
Pros:	Easily shows the status of each step. Exploitation of peripheral vision.
Cons:	Users have to remember the meaning of each color.

Idea:	**Cancel start/stop**
Pros:	Allows the user to cancel a start or stop, giving more precise data if start/stop was pressed by mistake.
Cons:	Time consuming to navigate to the Cancel button if there are a lot of surgery steps.
Idea:	**One recording per step per surgery**
Pros:	Expecting a step to be finished after the first recording allows for quicker navigation between the remaining steps that are yet to be recorded.
Cons:	The surgeon will be unable to stop and resume the recording of a step, which may prove problematic for some types of surgery or if the surgeon needs a break during the surgery.

The implementation of the Pedal Prototype focused on the basic functionality needed for the final product, having a list of surgery steps and be able to mark the beginning and end of each step.

Product View: Feature List

- Start recording a step
- Stop recording a step
- Choose which step to start next
- Cancel a start of the current recording
- Cancel a stop of the last recording

In the Product View we described the filtering and manifestation dimensions for the prototype. A description of those dimensions is given below:

Product View: Filtering and Manifestation Dimensions

Filtering	**Description**
Appearance	Large buttons Clear colors, red, black, gray and blue
Interactivity	Input - Input given through pedal system and is mapped to keyboard keys. Output Command - System gives output by refreshing the visual feedback.
Manifestation	**Description**
Material	WPF 4.0 C# application.
Resolution	Performance - Feedback time. Interactivity - Usefulness of feedback.
Scope	Navigation using pedals. User feedback.

A SWOT analysis of the Pedal Prototype is described in the Process View.

Process View: SWOT Analysis

Strengths:	Can be operated without using hands. Easy to implement through key events. Somewhat easy to place in the operating theatre.
Weaknesses:	Users may press the wrong pedal by accident. Limitation between amount of commands and pedals needed. Must remember what each pedal does.
Opportunities:	Pedals may be useful for other types of surgery. Change size/feel of some pedals to distinct pedals without looking them.
Threats:	Might be impossible to place in some operating theatres. Stepping on a pedal can give balance and/or precision issues.

We found that this prototype lacks on several features, especially the visual feedback, therefore a new Pedal Prototype is to be made:

Project View: Project Planning

A new Pedal Prototype with improved visual feedback planned for development and testing.

10

Touchscreen Prototype

This chapter describes the development of the touchscreen prototype. As this prototype is much alike the Pedal Prototype in look and functionality, some information in this chapter will be a recurrence of Chapter 9. This prototype has the surgeon assistant as user.

Even though the surgeon assistant uses both hands a lot while conducting the surgery, the assistant is mostly using one hand when handling tools. Because of this observation, we considered the idea of using a touchscreen to markup the surgical steps.

The goals for this prototype were:

- Use touchscreen to markup the surgical steps by the assistant
- Use colors to clarify the current status of the markup
- Consider different positions to place the touchscreen to ensure it is easy to reach while not standing in the way of the surgery personel staff.

One factor that must be considered about the use of a touchscreen is whether it is feasible in regards to efficiency, attention from the user and hygiene. If it is not possible to make a solution that is fast to use, requires low attention from the user while being able to function in a sterile environment, the touchscreen will become an issue for the surgeon assistant.

One of the primary effects we wanted to make use of is colours to clarify the current state of the markup. The goal here is to make the visual feedback so clear that the user can use peripheral vision and quickly determine if the tool is in the correct state without losing focus on the patient.

Most of the hygiene issues can be solved by using plastic wrapping on the touchscreen, however, this puts some requirements to the touchscreen, as it cannot be a capacitive touchscreen that uses the finger as an electrical conductor. Instead, the touchscreen has to be an resistive touchscreen.

During our observations at the operating theater we found some uses for touchscreens to streamline some of the tasks, and this prototype served as a introduction to the use of touchscreens and getting an idea of their potential.

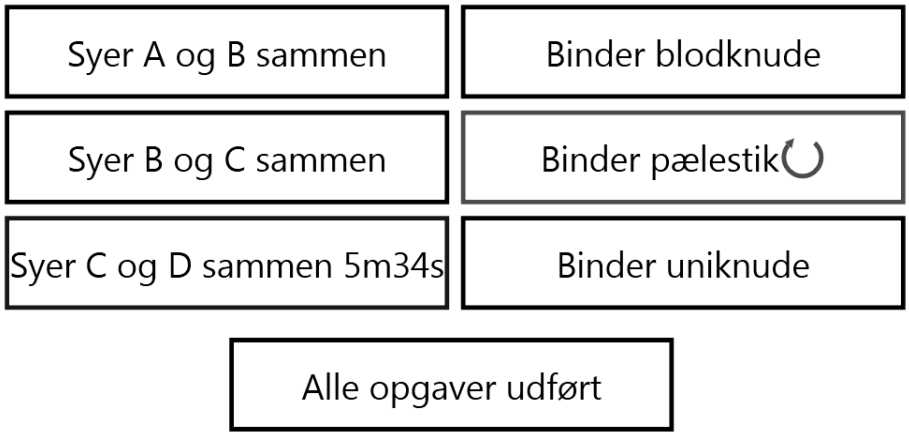


Figure 10.1: Screenshot of the touchscreen prototype. All buttons have a black border except button three ("Syer C og D sammen 5m34s") that is blue and button five ("Binder pælestik") is red.

10.1 Functionality

As with the Pedal Prototype, we used colors with inspiration from e.g. video recorders. However, unlike the Pedal Prototype, there is no need to show which button is "selected" as the user is supposed to press the appropriate button on the monitor. Therefore, the colors for the Touchscreen Prototype are as follows:

- *Black*: Steps that have not been marked up yet
- *Red*: The step that is currently being marked up
- *Blue*: Steps that have been marked up

The colors described above are applied to the border of the button. The user simply presses the button for the step that is to have its start marked up. When a step has ended and the surgeon progresses to the next step, the surgeon assistant presses on the step that is to begin. The prototype then marks the end of the first step and the beginning of the new step at the same time.

While the touchscreen requires focus from the user when a button is to be clicked, the color codes give the user a clear indication of the status of the markup the rest of the time. To improve this effect, the current step being marked up (red) also shows an animation to signal that a markup is currently active. A screenshot of the prototype can be seen on Figure 10.1.

As with the Pedal Prototype, the surgical step descriptions will be present on the buttons, such that the user can easily find out which step is to be started.

10.2 Test

The touchscreen prototype was demonstrated to the Surrogate Challenger to show the features and layout of the prototype. As with the Pedal Prototype, the color codes are too vague to be seen from a distance and the button text is too small for the user to see them from a distance.

Based on the input of our Surrogate Challenger, we chose not to conduct tests of the prototype and instead noted the following issues with this prototype:

- The color codes are not clear enough to be seen using peripheral vision. Consider applying the colors to the background of each button rather than the border.
- The system does not support pauses in the surgery, e.g. for consultation with other surgeons
- If the system is to be used with more than six or eight steps, the touchscreen prototype will not be able to show all of them at the same time. Therefore it is required that there are some indicators to show the user, that there are more step over and/underneath the currently shown steps.

After developing this prototype, we had a meeting with the surgery staff where we discussed the current prototypes. We found that the touchscreen prototype would not work with the surgeon assistant using it because of the combined issues of hygiene, placement and usability. Furthermore, the surgeon assistant often has to prepare for the next step by preparing tools, and while she is able to e.g. use the Pedal Prototype in this state, both her hands are occupied and therefore unable to use the touchscreen prototype efficiently.

10.3 Essence

As with the Pedal Prototype, the surgeon assistant is the only staff in focus. We put up the following scenario:

> **Scenario: Surgeon assistant performing markup during operation**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery.*

To make it possible to markup surgery steps with a touchscreen, we decided to create a button for each step as shown on Figure 10.1. The user then has to press the button that matches the step they are currently starting or ending. This also addresses the next scenario:

> **Scenario: Surgeon assistant performing markup during operation, surgical step order is different from the expected order**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery. The surgeon chooses to do the surgical steps in a different order than usually, and therefore the surgeon assistant has to navigate between the steps to markup the correct steps at the correct time.*

By having a button for each step, the user can press them in any order, however, you cannot mark the start of a new step if there is a step with a marked start and no marked stop. To increase the usefulness of the touchscreen markup tool, we introduced some ideas to test out in this prototype:

Process View: Idea Evaluation

Idea:	**Colour codes to show status for steps**
Pros:	Easily shows the status of each step. Exploitation of peripheral vision.
Cons:	Users have to remember the meaning of each color.
Idea:	**Cancel start/stop**
Pros:	Allows the user to cancel a start or stop by pressing the step button for 2 seconds, giving more precise data if start/stop was pressed by mistake.
Cons:	Time consuming to press the button and hold it down for 2 seconds.
Idea:	**One recording per step per surgery**
Pros:	Expecting a step to be finished after the first recording allows for quicker navigation between the remaining steps that are yet to be recorded.
Cons:	The surgeon will be unable to stop and resume the recording of a step, which may prove problematic for some types of surgery or if the surgeon needs a break during the surgery.

As with the Pedal and Speech to Text prototypes, we did a SWOT analysis of the touchscreen idea as a whole:

Process View: SWOT Analysis

Strengths:	Easy to navigate between steps. Easy to implement through button pressed events.
Weaknesses:	Difficult to place in the operating theatre, must be easy to reach and not in the way. Having to aim for a button might require too much attention from user, removing focus from the patient. Very bad usability if the user is currently using both hands for some o task.
Opportunities:	May be useful for other types of surgery.
Threats:	Might be impossible to place in some operating theatres. Hygiene issues, the surgeon assistant must be sterile.

We have specified the filtering and manifestation dimensions for the touchscreen prototype on the product view:

Product View: Filtering and Manifestation Dimensions

Filtering	**Description**
Appearance	Large buttons.
	Large font.
	Clear colors, red, black, gray and blue.
Interactivity	Input given through touchscreen.
	Output Command - System gives output by refreshing the visual feedback.
Manifestation	**Description**
Material	WPF 4.0 C# application.
Resolution	Performance - Feedback time.
	Interactivity - Usefulness of feedback.
Scope	User interaction.

During the development of this prototype we got some ideas for further use of screens and touchscreens in this project.

Touchscreen might prove useful for other tasks/users, e.g. to markup preparation and finishing steps (preparing the operating theatre for the patient and cleaning up after the surgery). The touchscreens could then be used by the nurses and cleaning personel.

Screens with or without touch may used along with pedals and/or speech to text to give visual feedback and work as a backup solution for the pedal/ speech to text product.

As described earlier in this chapter, the surgery staff found the touchscreen prototype to be too problematic for use with the surgeon assistant. Therefore we chose to stop further development on this prototype.

Project View: Project Planning

Plans on further development of the touchscreen prototyped have been ceased due to the feedback given on the first prototype.

11

Speech to Text Prototype 2

This chapter describes the functionality of the second STT Prototype along with tests and a description the development of this prototype through the Essence Views.

After developing and trying out the first STT Prototype, we found that using long commands increases the risk that the command is misunderstood or not recognized at all. We also found that our test setup had to be changed, as the previous tasks were too easy to solve without focusing fully on the monitor.

11.1 Functionality

As with the first prototype, the main functionality is still:

- Navigation between steps
- Markup the start and stop of each step
- Cancel the latest markup (if a start or stop is marked up by mistake)

However, due to problems when it comes to recognizing long commands (and remembering them as you will need to express them precisely for the system to recognize them), we implemented new commands to navigate between steps. The commands are shown in Table 11.1, and they are still prefixed with *computer*.

With these new commands, steps are chosen by using the *Næste* (next) and *Tilbage* (back) commands to navigate to the step the surgeon is going to do next. As the steps are mostly done sequentially (thus following the order we were given by the surgeons back in previous semester), we chose to implement a shortcut for selecting next step and starting it right away. This new command is the *Bekræft* (confirm) command, and it can only be issued right after using the *Stop* command to mark the stop of the current step. After stopping a step, the audio feedback message will ask you if you want to go straight to the next step (the system will tell the user the number and name of the next step so it is easier to figure if the user wants to confirm it), and then this can be confirmed by using the *Bekræft* (confirm) command.

Functionality	**Command**
Select next step	Næste (Next)
Select previous step	Tilbage (Back)
Start recording of the selected step	Aktiver (Activate)
Stop recording of the selected step	Stop
Cancels the start and/or stop the recording	Annullér (Cancel)
Shortcut for selecting next step and starting it right away	Bekræft (Confirm)
Terminates the markup of the surgery	Afslut (Finish)

Table 11.1: Voice commands without the *computer* prefix

After consultations with the Surrogate Challenger, we chose to implement a state machine in our next prototype. The state machine allows us to use shorter commands and allows the user to navigate between the surgical steps and pick the correct one rather than having to remember the exact name of each step. We implemented the state machine to have two primary states,

- an inactive state, where no step is being marked up and it is possible to navigate between steps and
- an active state, where a step is being marked up and the user can only mark the end of that step.

The state machine is described in the Product View, Section 11.3.1.

11.2 Testing

After meetings with the Surrogate Challenger, we decided to make use of *dot-to-dot drawings* (example shown on Figure 11.1) as tasks for the tests for this prototype. We used the same setup as with the first STT Prototype, which can be seen in Section 8.2. We chose dot-to-dot drawing for the following reasons:

- They require very little if any introduction to the Surrogate Challenger and Surrogate Users.
- Due to the setup with a webcam, it requires full attention to draw the drawing precisely as you cannot look directly your hands nor the paper but have to look at the monitor.
- The tasks take fairly short time to complete (around 1-2 minutes each), which allows us to put the user through several tasks and test out the navigation between steps.

11.2.1 Surrogate Challenger

The first test was done with the Surrogate Challenger as our test user. We gave the following drawings as steps to be marked up using the STT Prototype:

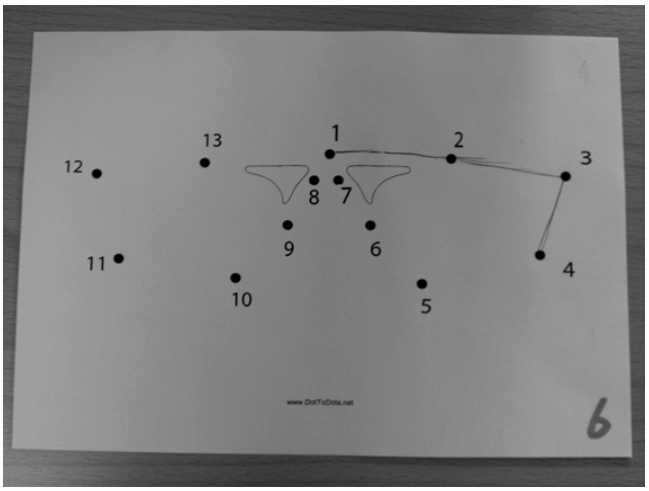


Figure 11.1: Example of a dot-to-dot drawing. This picture is used as step 6 in our test and depicts a pair of glasses

1. Megaphone
2. Heart
3. Rabbit
4. Whale
5. Shark
6. Glasses
7. Cap

On the corner of the dot-to-dot drawing, we drew the step number, so the Surrogate Challenger and Surrogate Users always knew which step number to choose. Although as this is not possible for the surgeons, they have had years of training and experience with the surgery, and thus know which order to do the surgical steps in. Therefore we do not expect this detail to be of importance to the test results.

The Surrogate Challenger was told to start with step 1, Megaphone, and continue sequentially to step 7, Cap. However, to test out the navigation commands, we asked him to stop when he was halfway through step 3 and switch to step 4. After completing step 4 and 5, we asked him to complete step 3 and finally step 6 and 7.

We registered every time a recognition was missed, which happened four times during the test.

The Surrogate Challenger gave us the following input on the prototype:

- The cancel command is useful and works well
- The commands are easy to learn

- The system should give feedback when the system recognizes "computer" but not the command following it.
- Whenever a command is issued, the system gives you an audio feedback updating the user on the current status of the system. Currently this feedback cannot be interrupted, e.g. by issuing a new command. We should make it possible to interrupt the audio feedback by issuing a new command, so users can do the markup quicker if they know what command to issue next without hearing all of the audio feedback.

During our discussion, we discussed if a shortcut for starting the markup of the next step, after stopping a step should be made, because the surgeons will be following the surgical step order sequentially most if not all of the time.

Of the proposed changes, we managed to implement the shortcut for the next step and that the system will return a feedback to the user if only "computer" is recognized before we tested the system with the Surrogate Users.

Furthermore, he gave us the following input on the test:

- We need to explain precisely what the Surrogate Users must do during the test and in which order, as they have not seen the prototype before nor know about the case we are working on in this project.
- To give the user a better overview of the commands, we should supply them with a paper showing the state diagram and the available commands
- The test user should have some time before the actual test to try out the different commands and feedback from the commands. Furthermore, as the system tends to "learn" the voice of the user, a few minutes head start would making the system more used to the user by the time the actual test takes place
- The tasks are good, as they require focus from the test user without requiring much knowledge on beforehand.

We followed up on all of his suggestions to the test itself and prepared them to make the most of the tests with the Surrogate Users.

11.2.2 Surrogate Users

Almost a week after the test with the Surrogate Challenger, we executed a test with five Surrogate Users. They were given the same steps and order of steps as the Surrogate Challenger. Each test subject was given a few minutes to test out the system before bringing in the tasks and starting the actual tests, we described how they should be using the system:

1. Start the system
2. Task is brought into the cardboard box and becomes visible to the test user.
3. Identify task number and mark the start of the appropriate step
4. Complete the task

5. Mark the end of the step
6. Repeat steps 2-5

As with the Surrogate Challenger, we asked the Surrogate User to stop halfway through step 3. However, we asked them to do step 5, 6, 7 after doing first half of step 3, leaving the order of steps to be 1, 2, first half of 3, 5, 6, 7, last half of 3, 4. This way, the Surrogate Users would have to use both *Next* and *Back* commands to navigate back and forth between the steps to complete the test.

During the tests we noted every time

- a command by the user was not recognized,
- a command by the user was confused with another command,
- when the user issues a wrong command and
- when the user used the status command.

11.2.2.1 Test Results

We gave each command a number, and noted that number every time that command was implicated in one of the error described above. We did not note how many commands each Surrogate User issued. However, we counted how many commands are needed:

- 4x Activate
- 5x Confirm
- 8x Stop
- 2x Next
- 4x Previous
- 1x Finish

To complete the tasks given in this test with as few commands as possible, minimum 24 commands are needed per Surrogate User (120 commands for all five Surrogate Users). Minimum because, if a Surrogate User uses a *Next* command and an *Activate* command instead of a *Confirm* command, the Surrogate User will use more than 24 commands. However, he can still complete the tasks successfully.

The results from the test are shown in Table 11.2.

As specified above, a minimum of 120 commands was needed to complete the tasks for the five Surrogate Users, and during the test we noted 38 missed recognitions out of minimum 158 commands (minimum 120 to complete the tasks, 38 missed recognitions and one command that was out of context).

From these results we found that we have reached some of our goals, especially that no commands are so much alike that they might be misunderstood for each other as e.g. *Start* and *Stop* was in our own tests. In this test we had zero commands that was confused as other commands by the speech recognition. Furthermore, the Surrogate Users knew which step and state they were in

Missed recognition	**Misunderstood recognition**	**Wrong command**	**Status command**
Activate(8) Stop(5) Confirm(4) Next(1) Back(3)			
Activate(1) Confirm(1) Next(2)			
Next(1)		Cancel(1)	
Activate(2) Stop(5) Confirm(1) Back(1)			
Activate(1) Stop(1) Back(1)			

Table 11.2: The test results from the test of the second STT Prototype. Each user is represented by a row. The numbers given in parentheses are the number of times the error occurred, e.g. Stop (3) in the "Missed recognition" column means that the *Stop* command was not recognized three times during the test.

almost all the time, they only issued commands out of context one time in total during the tests.

The biggest issue we found was the recognition, especially for our first test user. However, many of the errors by the first test user were because he got frustrated and therefore did not have the patience to say the commands in a normal manner, and often ended up talking in syllables, which only makes it harder for the system to recognize commands. This was most apparent in the last part of the test where the *Aktiver* (activate) command was missed by the recognition software six times in a row.

The fourth test user also had problems issuing the *Stop* command as the system did not recognize it four times in a row. Furthermore, we found that the headset, while being the best we currently had available, it was not good enough for this system, and a new headset was ordered to be used for the next prototype.

On top of the results, we asked the Surrogate Users a set of questions:

1. What do you think of the precision of the recognition?
2. Did you always know which step and state the tool was in?
3. Did you get the necessary feedback from the system? Did you get too much or too little? Did you miss visual response?
4. Are the commands for navigating between steps logical?
5. Are the commands easy to learn?

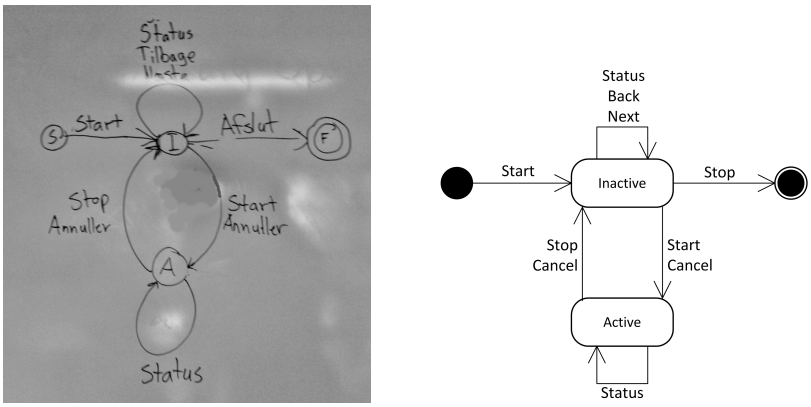
6. Are the commands useful? Should we add or remove any commands?
7. Do the names of each command make sense compared to the functionality of the commands?

As answers to these questions, we were given the following input on the prototype by the Surrogate Users:

- When navigating between more than 1 or 2 steps, the navigation felt very slow because the users had to wait for the audio feedback to finish. A shortcut for choosing a step (e.g. "computer step 5") would be nice for such situations.
- The commands are easy to learn, the names and their functionality makes sense
- The audio feedback was too long, at least for some commands
- The first Surrogate User found the system to be very bad at recognizing commands, the other users found the system to be well beyond their expectations for speech recognition software.
- Some of the users would like to be able to interrupt the audio feedback if they know which command to say next anyway
- The amount of commands is good, makes it easy to remember all of the commands and no commands seem to be missing
- After issuing the first handful commands, the users found that they did not have to focus on speaking legible but were able to speak normally and the system would still recognize the command
- The shortcut for starting the next step worked very well, and if the surgeons follow the surgical steps sequentially, the Surrogate Users believe that shortcut will help a lot
- *Activate* should probably have its name changed, e.g. "Begin" or "Record (Step)".
- The *Cancel* command was a bit confusing at first to some of the users, especially because it does not apply to all commands.
- The appended state diagram was a very useful tool in the beginning to learn the commands and their use, but after a short while most users did not require the diagram to use the system.

11.3 Essence

One of the primary ideas we implemented in this prototype is the use of states and make the availability of commands depend on which state you are in.



(a) First version of the state diagram in Danish

(b) First version of the state diagram in English

Figure 11.2: Overview of the proposed states in the STT Prototype.

Scenario: Surgeon only uses commands when they are in the context
The surgeon only needs certain commands when they are in the context, e.g. when the surgeon has marked up the start of a step, he does not need to mark another start until he has marked the stop of a step.

By introducing states we hope to make the current status more apparent to the user and to make each command make more sense because they can only be used in the correct context.

The state diagram is shown on Figure 11.2. Unlike the first set of prototypes, this setup requires the user to mark both the start and the stop of a step, where the first set of prototypes ended the current step whenever the next step was started. This allows us to make most of the commands situational, when the user is recording a step the only commands available is *Stop* to mark the stop of the current step, and *Status* to get the status of the tool. Furthermore, with this approach the surgeon can take a pause without ruining the markup data, e.g. if the surgeon has to consult another surgeon about a step, as the surgeon can stop the current step and start it again whenever the surgeon is ready to continue the surgery. In the previous set of prototypes the surgeon was expected to complete the surgery without pauses as the time pausing would be added to the current step.

An overview of the commands and their functionality is given in Table 11.3.

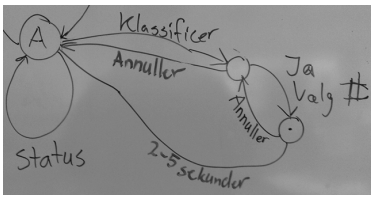
Specifying special criteria was mentioned in the first STT Prototype, however, as the main goal of that prototype was to determine if speech recognition was a viable solution, special criteria was postponed to this prototype.

Examples of such special criteria are:

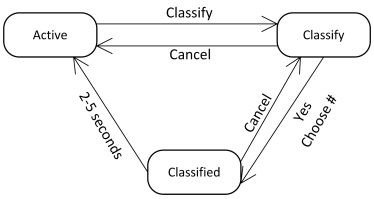
- if the patient has previously had a surgery in the same area as for the current surgery,
- if the patient is severely overweight and

Commands	Functionality
Start	Initial State: Starts the markup system. Inactive state: Marks the beginning/continuation of the currently chosen step.
Stop	Marks the ending of the currently active step.
Status	Gives the user audio feedback on the status of the tool, e.g. the current step and whether the step is active.
Tilbage (Back)	Chooses the previous step. If current step is 2, step 1 will be chosen.
Næste (Next)	Choose the next step. If current step is 1, step 2 will be chosen.
Annuller (Cancel)	Cancels the last Start or Stop command, depending on which one was executed most recently.
Afslut (Finish)	Stops the markup system and saves the markup.

Table 11.3: Voice commands without the *computer* prefix



(a) Classify state diagram in Danish



(b) Classify state diagram in English

Figure 11.3: The classification idea integrated in the STT state diagram.

- whether the surgery goal is to remove only the prostate and save the nerves around the prostate or to remove the nerves as well.

  Scenario: Surgeon wants to mark special criteria on current surgery
  Patients are different from each other, and some patients may have specific properties that change how the surgeon must approach the surgery, changing factors such as time and patient outcome. To compare relevant surgeries to each other, such criteria must be noted. An example of such criteria is that the patient has previously had a surgery in the area where he is being operated.

As this feature will ultimately serve to group surgeries of the same kind for more realistic comparisons, we chose to name the command *Klassificer* (classify). The *Klassificer* (classify) is available when a step is currently being marked up, shown as the Active state on Figure 11.2. When the *Klassificer* (classify) command has been given, the current state will be the Classify state. The system will then give the user a list of classifications one by one, and the user can then respond with a *Bekræft* (confirm) to choose that classification. Alterna-

Commands	**Functionality**
Klassificer (classify)	Goes to the classification state and starts stating the classifications one by one, e.g. overweight patient, previously had a surgery and removal of nerves near the prostate.
Ja (Yes)	Chooses the classification that was just stated by the system and adds it as a classification to the surgery
Annuller (Cancel)	Classify State: Goes back to the Active state, stops stating the classifications. Classified State: Cancels the chosen classification and starts stating the classifications again.
2-5 sekunder (seconds)	A time period where it is possible to cancel the classification, e.g. 2-5 seconds. After this period, the user cannot cancel the choice of classification.

Table 11.4: Voice commands without the *computer* prefix

tively the user can choose the number of that classification to choose it right away. The classifications and their numbers will be on a list near the surgeon module, so the surgeon can obtain the number rather easily. The commands attached to the Classify feature are shown on Table 11.4 and a diagram of the Classify feature is shown on Figure 11.3.

Even though the system leaves the 'Active' state, the system will still be active and consider the current step to be active. The idea is that the surgeon chooses classifications while continuing the surgery.

We evaluated these ideas in the Process View:

Process View: Idea Evaluation

Idea: **States**
Pros: Possible to use shorter commands. Follows the surgery state more closely. Commands only available when they actually make sense to be available.
Cons: User needs to remember which state the system is in, or use the *Status* command. User needs to remember when the command is available.

Idea: **Classify surgeries using STT**
Pros: It has to be done somehow, or comparisons of surgeries will be much less useful.
Cons: Increases the amount of commands. The user either has to remember the number of that classification or wait for the system to state the correct classification before confirming it. The alternative to using numbers when classifying would be to have the user speak the name of the classification, but this will most likely give the same issues as we saw in the first STT Prototype.

We revised the SWOT analysis of speech recognition in the Process View with our new knowledge on precision and the prototype:

Filtering	**Description**
Interactivity	Sound Input - User gives spoken commands. Output Command - system outputs a line describing which spoken command it recognized. Audio Feedback - system gives audio feedback to the user to confirm recognition of spoken command.
Manifestation	**Description**
Material	Java console application.
Resolution	Performance - Speech recognition time. Accuracy - Precision of recognition. Interactivity - Usefulness of feedback.
Scope	Speech recognition. User feedback.

Table 11.5: Filtering and Manifestation dimensions for STT Prototype 2

Process View: SWOT Analysis

Strengths:	Can be operated without occupying hands nor feet Does not require visual focus Precision can be improved through use of noise cancelling techniques Use of short commands that sound much different from each other creases the precision a lot
Weaknesses:	Hard to implement Speech recognition is far from fully matured No visual feedback. Navigating between steps is rather slow
Opportunities:	May be useful for other types of surgery. Lots of focus on speech recognition, the technology will surely mat over time
Threats:	Dialects and noisy facilities may be problematic in regards to precisi

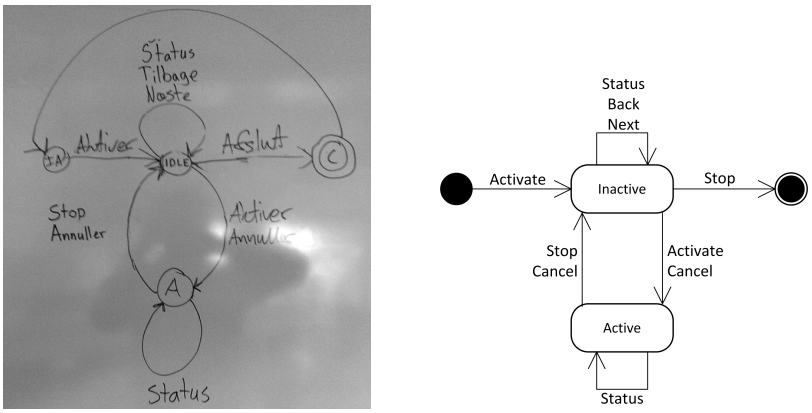
11.3.1 Product View

We decided to continue with the idea of implementing states in the prototype. As described below, *Classify* makes the system a lot more complex, and therefore we chose to leave this part of the prototype out until we have a better solution for classifying special criteria. As this prototype has more goals to satisfy than our first STT Prototype, the filtering and manifestation dimensions have changed somewhat. The revised dimensions are described in Table 11.5

The state diagram was further developed as we began to design and implement the prototype. The final state diagram for this prototype can be seen on Figure 11.4.

During the development of the prototype, we found that the *Start* and *Stop* were too alike, and therefore the system had a tendency to mix up *Start* and *Stop*, which became very problematic. Therefore, we chose to replace *Start* with *Aktiver* (activate). We chose to restrict the *Annuller* (cancel) command such that it is only able to cancel *Start* and *Stop* commands. *Start* and *Stop* are the only commands that have a direct influence on the markup, and therefore the only steps where a mistaken command can mess up the markup.

After developing the prototype and executing tests with Surrogate Chal-



(a) State diagram in Danish

(b) State diagram in English

Figure 11.4: The state diagram of commands in the STT Prototype, revised during development of the prototype.

lenger and Surrogate Users, we found some issues that are to be fixed in a third version of the prototype. Therefore the development and test of a third prototype is planned in the Project View:

Project View: Project Planning

Version 3 of the STT Prototype to be developed for tests from the 30th of March to the 19th of April. Tests on the 19th of April.

12

Pedal Prototype 2

This chapter describes the functionality, test and development of the second Pedal Prototype.

In the first Pedal Prototype we found a number of issues in collaboration with our Surrogate Challenger, e.g. that the visual feedback was not clear enough to actually exploit peripheral vision as it was hard to read the text and see the colors from distance.

For this prototype, the main focus was to improve the visual feedback and implementing the concept of states as was done with the second STT Prototype, which is described in Chapter 11.

12.1 Functionality

As described in the previous Pedal Prototype, increasing the number of pedals will make the system more complex for the user. This is because the user will need to remember what each pedal does and be certain which pedal the user is currently pressing, preferably without looking at the pedals as the surgeon assistant should be looking at the patient and the tools he uses.

In order to allow for more commands while keeping the system simple to use, we implemented the two following concepts in this prototype.

The first concept is states, which is the concept as used in the second STT Prototype. The main idea is to let the pedals get different functionality depending on the current state of the system. An example of this is to use one pedal for marking the start and the stop of a surgery step rather than using one pedal for each. This concept is seen in other electronic products such as music players and some remote controls to make the most of a few buttons. The state diagram for this prototype is shown in Section 12.3.1.

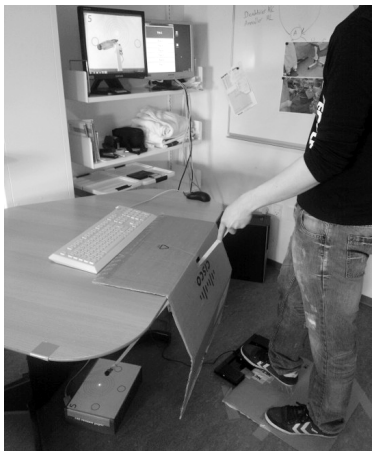
The second concept is overloading pedals to get more commands per pedal. Each pedal can trigger 2 commands, one by pressing the pedal and releasing quickly and one for holding down the pedal. The pedals are shown on Table 12.1.

In order to make the text of each step easily readable, the font size was increased remarkably compared to the first prototype. This however gave another problem as some steps have a long description. In order to comply with this problem, only the currently selected step shows the complete description while

Pedal	Press	Hold
Left	Previous Step	Cycle Previous Steps
Center	Start/Stop Recording	Cancel Start/Stop Command
Right	Next Step	Cycle Next Steps

Table 12.1: Pedals overloaded functions

(a) The device used to represent one of the tools used by the surgeon assistant.



(b) Overview of the setup for testing the second Pedal Prototype.

Figure 12.1: Pictures from the test setup of the Pedal Prototype.

Figure 12.2: Monitor setup for the second prototype.

all other steps only show part of the description. The first part of the description should be enough to distinct it from the other steps and by selecting the step it will be possible to read all of the description if needed.

In order to help the user focus on the surgery rather than the markup monitor, we changed the color coding to affect the background of each button rather than the border. This creates a more visible effect whenever a button changes color and makes it easier to see in the peripheral vision.

Each step can now have several records if needed. This allows the surgeon to stop a step, e.g. to take a break or to move on to another step, and then finish the step later.

The pedals themselves, which was a Logitech G25 in the first prototype, were replaced by a Triple Foot Switch with pedals than are binary rather than analogue. This solves the issue of various springing on the pedals while giving a more clear feeling of whether the pedal is actually pressed or not. The pedals are also further apart which prevents clicking more than one pedal at the same time, which happened quite often with the previous prototype.

12.2 Testing

The main focus for this test was to test whether the use of this prototype would interfere with the other tasks the surgeon assistant has to do. One of the main tools of the surgeon assistant is a tool for sucking up liquids and cleaning organs to give the surgeon a better overview. To represent this tool in a test, we created a device as shown on Figure 12.1a. This device consists of a laser pointer and a felt-tip pen mounted on a stick.

This device is then to be used to hit certain targets just like the surgeon assistant would have to reach certain areas inside the patient to help out the surgeon. For the test we made a series of tasks where the user is to hit several targets. There are two types of targets and four targets in total, three small ones with a red border and one large target with several borders, each border being two centimetres larger in diameter than the previous, like a small shooting target. The targets can be seen on the left monitor on Figure 12.2. Each set of targets has a number written on them and this number represents the step number, e.g. the set of targets with a 5 written on it represents the fifth step in the markup.

The test is then to be carried out by doing as follows. First the test subject steps onto the square shown on Figure 12.3 and grabs the device for marking the targets. A set of targets is placed in the test setup and the test subject is then to mark the centre of the shooting target with the laser pointer. While aiming at the centre, the test subject must mark the beginning of the step using the pedals. When the step has begun, the targets with a red border have to be marked with the felt-tip pen in any order. Afterwards the testsubject is to mark the center of the shooting target with the laser pointer again and then use the pedals to markup that the step has ended. After ending the step, a new set of targets will be placed in the test setup and the process is repeated. The test subject is not allowed to keep his foot placed on the pedals but has to get back to a normal standing position on the square.

This test was conducted to investigate two factors. The first goal of the test was to investigate if moving from a normal standing position to press a pedal

Figure 12.3: View of the pedals and one of the tasks in the test.

may risk that the user loses control of the tool. The second goal of the test was to find out if using the pedals and monitor would be taking a lot of focus away from the task of controlling the device just as the surgeon assistant would have to focus on the tools he is controlling inside the patient.

For the test we had five test subject, three students and two associate professors from our own department. Each test subject was to complete a set of 7 tasks in a specific order: 1, 2, 5, 6, 7, 3, 4, 5. The first time step 5 is started we tell the test subject to cancel the step and thereafter puts on the sixth set of targets in the test setup.

12.2.1 Test Results

We noted every time one of the following issues occurred:

- User selected an incorrect pedal for the situation (e.g. pressing middle pedal to start step 2 when the user was expected to start step 4)
- User selected a pedal but the selection was not registered by the prototype

To complete the test 100% correctly, the Surrogate Users will have issued the following commands:

- 8x Record
- 1x Cancel
- 7x Stop
- 5x Next
- 1x Next (cycle)
- 1x Previous (cycle)

This gives a total of 23 steps in the best case. If the Surrogate User e.g. does not use the overloading on Next and Previous, the Surrogate User will have pressed more than 23 times, however without that being an error/mistake.

The first Surrogate User wanted to stop a recording, pressed middle pedal twice by mistake (thus stopped and started recording again) and instead of pressing Cancel, the Surrogate User stopped the step again. The result of this is that the step in question now has two recordings instead of one and has an additional one or two seconds of extra time added to its duration.

The second Surrogate User stopped step 5 instead of cancelling it when we asked the user to cancel step 5 and go to step 6.

The third Surrogate User got the same error as the first Surrogate User.

The amount of errors therefore totals to five errors, three of them being Stop commands and two of them being Record commands. The minimum amount of steps per Surrogate User is 23 times. With 5 Surrogate Users, this gives 115 steps plus the five errors totalling 120 steps. We consider this amount of errors to be very low, and most of them bound in the quality of the pedals as it is too easy to "double click" with the current pedals. Furthermore, we talked with the Surrogate Users and found that they realized they should have used the Cancel functionality rather than just stopping the step, and the Surrogate Users said

it is most likely a matter of getting used to the pedals and the functions they supply.

We found that the test persons were able to keep the pointing tool stable during the use of the Pedal Prototype and the test subjects themselves said it was easy to focus on the task while using the Pedal Prototype to markup the start and end of each step. None of the Surrogate Users did a large swing with the tool during the test, and we find that managing the tool while using pedals will not be an issue. This was supported by Jane Petersson during the meeting on the 28th of March, where she said managing the tool is easy, and it is possible to use pedals while managing the job of being surgeon assistant. The Surrogate Users found the pedal setup to be intuitive and that it was easy to learn how to use it. Some of the test subject had trouble feeling which pedal they were about to press and therefore had to look down, removing focus from the task.

This problem can become even larger for the surgeon assistant as he has to move a couple of steps during the surgery to pick up tools from a nearby sterile table, so we found that there should be a quick and easy way to find out which pedal the user is pressing without looking down. A solution to this can be a tray to place ones heel in when about to press a pedal so the user can feel which pedal is the middle pedal.

The test subjects also found that using the peripheral vision to detect the feedback from the system was very helpful in order to focus on the patient while using the prototype and that the colors were clear enough to give a indication of what state the system is in. Moving from e.g. step 2 to step 5 required some focus on the other monitor, however, as you will have to read the step description to ensure you are starting the correct step.

Because of the positive results from these tests, we find the prototype to be ready for demonstration to the Department of Urology.

12.3 Essence

The ideas created for the prototypes were noted in the Paradigm view:

The scenarios described in the first Pedal Prototype remains for the second prototype:

> **Scenario: Surgeon assistant performing markup during operation using pedals**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery. To do this, the surgeon assistant has a set of pedals available to navigate between steps and markup the beginning and end of steps.*

> **Scenario: Surgeon assistant performing markup during operation, surgical step order is different from the expected order**
> *The surgeon assistant wants to markup every time the surgeon starts and stops a step in the surgery process while assisting the surgery. The surgeon chooses to do the surgical steps in a different order than usually, and therefore the surgeon assistant has to navigate between the steps to markup the correct steps at the correct time.*

These two scenarios describe the basic functionality implemented in the first prototype, and which is part of the second prototype as well. However, we also added the following scenario:

> **Scenario: Pause during the surgery**
> *The surgeon may need to pause the surgery for some minutes, e.g. to consult other surgeons on the surgery if the surgery has special conditions*

Unlike the first prototype, the second prototype is able to enter a state where no step is currently active.

In the Process View we made a SWOT analysis of the pedal idea as a whole:

Process View: SWOT Analysis

Strengths:	Can be operated without using hands. Easy to implement through key events. Easy to place in the operating theatre.
Weaknesses:	Users may press the wrong pedal by accident. There is a limitation between amount of available commands and pe needed. It may be hard to remember what each pedal does
Opportunities:	Pedals may prove useful for marking up other kinds of surgery Giving each pedal a different physical feel it may be easier to different one pedal from another.
Threats:	As much as pedals may be easy to place in some operating theatres, t may be impossible to place in other operating theatres Stepping on a pedal may give balance/precision issues.

The pedal idea consists of several smaller ideas, and each of them was analysed to determine its advantages or disadvantages, before ultimately deciding whether or not it is actually advantageous to the prototype.

Process View: Idea Evaluation

Idea:	**Color codes**
Pros:	Shows the status of each step clearly. Exploitation of peripheral vision.
Cons:	Memorization of each color.
Idea:	**Cancel start/stop**
Pros:	Cancellation of start/stop errors made by the user, thus saving the data from errors.
Cons:	Requires either an button that must be navigated to, overload of pedals or an extra pedal.
Idea:	**Overload of pedals**
Pros:	Allows each pedal to have more than one command.
Cons:	Makes it harder for the users to remember what each pedal does.
Idea:	**Color codes on the pedals**
Pros:	By looking at the color codes on the pedals (e.g. middle pedal is red and blue), the user can see what each pedal corresponds to on the monitor.

Cons: The user should not be looking at the pedals as it removes focus from the patient.

Idea: **Show only first X characters of each step description**
Pros: Allows us to increase font size and have several steps visible on the monitor.
Cons: You have to select the step to read the full description.

Idea: **Implement states in the prototype**
Pros: By restricting commands to certain states, we can deactivate commands that do not make sense in the current context.
Cons: The user will have to remember when a command is valid, e.g. that he cannot choose next or previous step when a step is currently being recorded.

12.3.1 Product View

As with the STT Prototype 2, this prototype had the concept of states implemented to give the users a clearer overview of the functionality and when it is available. It also allows us to make the most of a few pedals (3 in this prototype) as we can use a pedal for one command in one state and another command in another state. The filtering and manifestation dimensions have therefore changed accordingly as shown in Table 12.3.

We revised the filtering and manifestation dimensions for this prototype. A description of those dimensions is given below:

The state diagram can be seen on Figure 12.4. The diagrams use some abbreviations to show which pedal is to be pressed to execute the given command. The abbreviations are explained in Table 12.4. Furthermore the (cycle) functionality given in the diagram means that it will cycle through the steps as you hold down the pedal as an alternative to press once for every step the user needs to navigate past.

We found that this prototype only needs some finishing touches in regards to font size, especially the size of the step numbers for easy visibility from range. The changes to the font size can be seen on Figure 12.5.

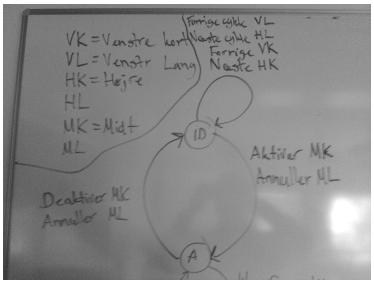
The Pedal Prototype is considered ready to be demonstrated to the surgical staff at the Department of Urology

Filtering	**Description**
Appearance	Large buttons. Large font. Clear colors, red, black, gray and blue
Interactivity	Input - Input given through pedal system and is mapped to keyboard keys. Output Command - System gives output by refreshing the visual feedback.
Manifestation	**Description**
Material	WPF 4.0 C# application.
Resolution	Performance - Feedback time. Interactivity - Usefulness of feedback.
Scope	Navigation using pedals. User feedback.

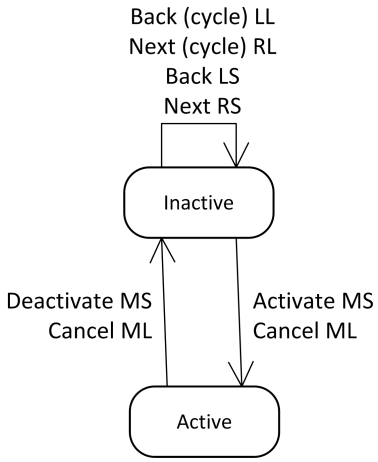
Table 12.3: Filtering and Manifestation dimensions for Pedal Prototype 2

Abbreviation (English)	**Meaning**
VK (LS)	Press left pedal
HK (RS)	Press right pedal
MK (MS)	Press middle pedal
VL (LL)	Press and hold left pedal
HL (RL)	Press and hold right pedal
ML (ML)	Press and hold middle pedal

Table 12.4: Product View: Overview of abbreviations used in the state diagrams. For example LS means "Left Short" where short referes to a short press, and LL means "Left Long", where LL refers to a long press.



(a) Pedal state diagram in Danish



(b) Pedal state diagram in English

Figure 12.4: The state diagram of commands in the Pedal Prototype.

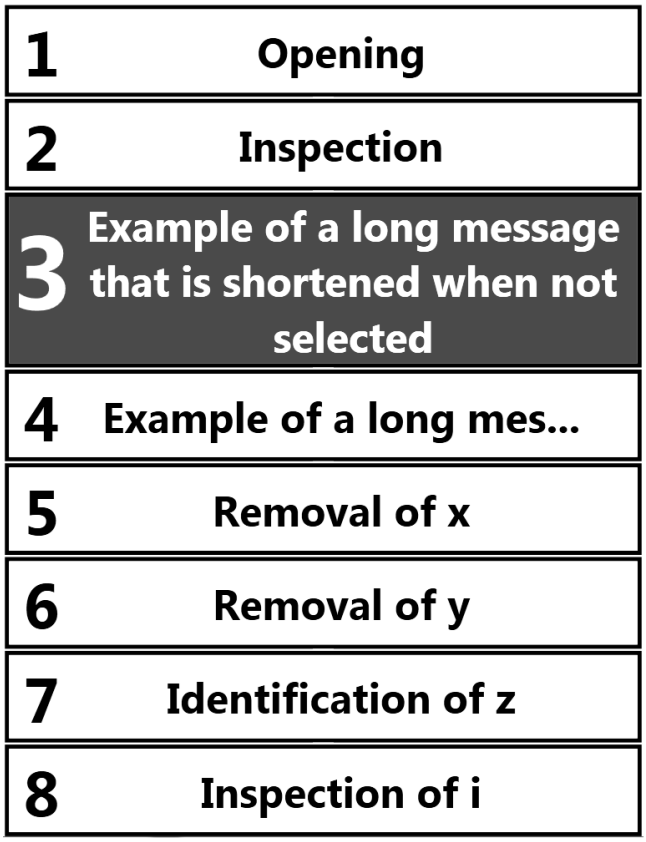


Figure 12.5: Final version of the Pedal Prototype

13

Speech to Text Prototype 3

This chapter describes the third STT Prototype as it went through development and tests along with relevant information from the Essence Views.

We had a meeting with the Department of Urology on the 29th of March, where we discussed the current status of the prototypes and scenarios.

We discussed one of the new ideas of being able to choose a step in the STT Prototype by saying e.g. "computer step one" or "computer step nine" to choose step one or step nine. Johan Poulsen thought that would be a useful way of using the system as remembering the steps as numbers should be fairly easy.

We played some of the audio feedback messages from the commands, and he found them to be too long, at least for his own use. When the system is used with educational purposes, the current amount of feedback might be more fitting.

We decided to keep the *Næste* (next) and *Tilbage* (back) commands as they may be useful for moving one step, e.g. if you are not sure which step number you are currently on but knows you are going to do the next step in the surgical step order.

13.1 Functionality

Based on the feedback from the second STT Prototype, we chose to implement a few features to increase the usability of the prototype and make the state diagram simpler for the users.

First and foremost, the ability to interrupt the audio feedback was implemented. The audio feedback stops if it recognizes a new command, executes the new command and gives audio feedback on the new command. E.g. if a *Next* command is issued and a *Status* command is issued before the audio feedback for *Next* has ended, the system will stop that feedback and give the user feedback of the current status of the system.

The command *Trin x* (step x) was added. This command allows the user to jump to whatever step number they want with one command, as long as it is an available step number. Because of this new feature, the *Tilbage* (back) was renamed to *Forrige* (previous) to depict that it chooses the step before the

current step in the surgical step order. E.g. if the user uses *Trin 5* to get from step 1 to step 5, *Forrige* will navigate you to step 4 and not back to step 1.

The *Activate* command has been renamed to *Record*. *Activate* was the only command where test users felt the name did not depict its functionality sufficiently, and as one of the main goals of this product is to supply video clips for each surgery step, *Record* seems to represent the functionality more precisely.

The Inactive state has been removed from the users point of view, and the system automatically moves to the Idle state when the system starts. Having to turn on the computer and start the software should be enough to begin marking up.

Lastly, we shortened several of the audio feedback messages.

The shortening of most audio feedback messages along with allowing users to interrupt the feedback should make the feedback less annoying when the user does not need the feedback while still providing feedback when needed.

13.2 Testing

The tests for this prototype was done in the same way as with the second STT Prototype, which is described in Section 11.2. The order of step is the same as with the tests in the second STT Prototype: 1, 2, first half of 3, 5, 6, 7, last half of 3, 4.

We did not note how many commands each Surrogate User issued, however, we enumerated the minimum amount of commands needed to complete the tasks in this test:

- 3x Activate
- 5x Confirm
- 8x Stop
- 2x Step x
- 1x Finish

Each Surrogate User will have used at least 19 commands (totalling 95 commands for the five Surrogate Users) to complete the tasks given in this test. If the Surrogate User e.g. uses several instances of *Previous* or *Next* instead of *Step x*, the tasks can still be completed, but it will require more than 19 commands.

The results from the test is shown in Table 13.1.

Compared to the previous test, the amount of missed recognitions has decreased substantially, from 38 to 15 out of minimum 110 commands (minimum 95 commands were recognized plus the 15 that were not), which is most likely because of the new headset used in this test as it a much clearer sound when recording using the microphone compared to the other headsets and microphones we have tested.

The *Bekræft* (Confirm) command was used in the wrong context a couple of times. The Surrogate Users tried to use *Bekræft* before they had used the *Stop* command.

Missed recognition	**Misunderstood recognition**	**Wrong command**	**Status command**
Stop (1) Step X (1)			
Step X (1) Stop (3)			
Stop (3)		Confirm (3)	
Stop (2) Confirm (1) Record (1)		Record (1)	
Stop (1) Record (1)		Record (1) Confirm (1)	

Table 13.1: Table showing the errors during the test of the third STT Prototype. Each user is represented by a row. The numbers given in parentheses are the number of times the error occurred, e.g. Stop (3) in the "Missed recognition" column means that the *Stop* command was not recognized three times during the test.

13.2.1 Test Feedback

We asked the same question as we asked in the previous test shown in Section 11.2. The Surrogate Users gave us the following feedback:

- It was very easy to talk to the system, no need to talk in syllables or speak slower than usual.
- Feedback was precise and fast. Being able to interrupt the feedback is useful and was used by most Surrogate Users later in the test as they began to remember the commands.
- Being able to use the *Trin x* (step x) command was useful whenever you have to jump more than one step. However, this feature requires the user to remember the number of the step.
- Because the *Trin x* (step x) was so easy to use, some of the Surrogate Users did not use *Næste* nor *Forrige* at all.
- The *Status* command is very useful for those few moments where you may forget which step and state the system is on.
- The audio feedback messages sound very different from each other, which makes it possible to quickly determine which response you are getting, rather than having to hear all of the feedback to realize what the system did.
- Rather than having the *Bekræft* (confirm) command, a substitute could be a *Continue* command that stops the current step, navigates to the next step and starts it right away. This will make it an even faster shortcut than the current *Stop* + *Bekræft* (confirm) shortcut.

13.3 Essence

For this prototype, we started with the following scenario:

> **Scenario: Surgeon jump to a later step in the surgical step order**
> *Sometimes it is necessary to do another step first, e.g. if the patient has had a previous surgery in the area. To mark this up more efficiently, being able to jump to a specific step can save a lot of time.*

In the previous prototype, jumping three steps or more was very slow and gave the user a lot of audio feedback compared to the task. In the first STT Prototype we ruled out the idea of using the names of the steps. However, using the step numbers may be possible, giving a *Trin x* (step x) command where x is the step number the user wants to jump to.

> **Scenario: Surgeon wants to skip the audio feedback**
> *When the surgeon is performing the surgery, he may already know which commands he wants to give the system to get the markup done as quick as possible so he can only 100% on the patient.*

To support this scenario, we made it possible to interrupt the audio feedback by issuing a new command, rather than having to wait for the audio feedback to finish before issuing the next command.

We did an evaluation of these two ideas in the Process View:

Process View: Idea Evaluation

Idea:	**Trin x (Step x) command**
Pros:	Makes navigating over two steps or more a lot faster than previously
Cons:	User needs to remember the number of the step he wants to navigate to
Idea:	**Interruptable audio feedback**
Pros:	Makes it possible to issue a set of commands a lot faster than previously.
Cons:	The user may miss crucial audio feedback, especially if the user issues a wrong command, e.g. saying *Næste* (next) to go to step 4 when he had to go to step 5, interrupting the audio feedback may cause the user to not be told that he is at step 4 and not step 5 before he has mistakenly started the markup of step 4.

We revised the SWOT analysis of speech recognition in the Process View:

Filtering	**Description**
Interactivity	Sound Input - User gives spoken commands. Output Command - system outputs a line describing which spoken command it recognized. Audio Feedback - system gives audio feedback to the user to confirm recognition of spoken command.
Manifestation	**Description**
Material	Java console application.
Resolution	Performance - Speech recognition time. Performance - Step navigation efficiency. Accuracy - Precision of recognition. Interactivity - Usefulness of feedback.
Scope	Speech recognition. User feedback.

Table 13.2: Filtering and Manifestation dimensions for STT Prototype 3

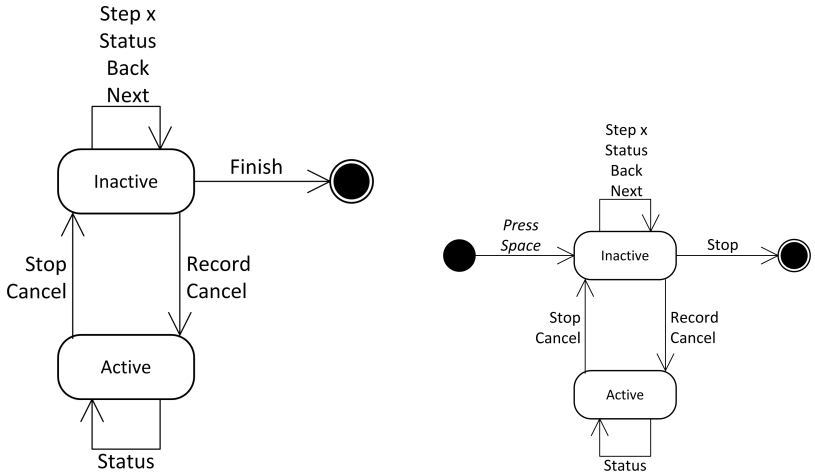
Process View: SWOT Analysis

Strengths:	Can be operated without occupying hands nor feet Does not require visual focus Precision can be improved through use of noise canceling techniques Use of short commands that sound much different from each other creases the precision a lot
Weaknesses:	Hard to implement Speech recognition is far from fully matured No visual feedback.
Opportunities:	May be useful for other types of surgery. Lots of focus on speech recognition, the technology will surely mat over time
Threats:	Dialects and noisy facilities may be problematic in regards to precisi

13.3.1 Product View

We mostly performed some changes to the existing features rather than adding new functionality. The filtering and manifestation dimensions for this prototype are described in Table 13.2.

One of the changes was shortening the audio feedback messages as shown in Table 13.3.1 to make the system feel quicker.



(a) States shown on paper to the test users

(b) States in the third STT Prototype

Figure 13.1: State diagrams from the third STT Prototype

Product View: Audio Feedback Messages

Original Message	**New Message**	**English Translation**
Computeren er aktiveret. Klar til at påbegynde opmærkningen. Nuværende trin er: *[step-details]*	System aktivt. Nuværende trin er *[step-details]*.	System activated. Current step is *[step-details]*.
Optagelse af trin annulleret. Det nuværende trin er: *[step-details]*.	Optag annulleret. Nuværende trin er: *[step-details]*.	Recording cancelled. Current step is: *[step-details]*.
Stop af trin annulleret. Fortsætter optagelse af trin: *[step-details]*.	Stop annulleret. Optager trin *[step-details]*.	Stop cancelled. Recording step *[step-details]*.
Gemmer opmærkningen og stopper computeren.	Opmærkning / Optagelse gemt.	Recording saved.
Optagelse stoppet. Du er i trin *[step-details]*, bekræft hvis du vil optage trin: *[step-details]*.	Optagelse stoppet. Du er i trin *[step-details]*, bekræft for at optage trin: *[step-details]*	Recording stopped. Current step is *[step-details]*, confirm to start recording step: *[step-details]*.

The one feature we added in this prototype is the *Trin x* (Step x) command, which allows the user to jump to a specific step is the user knows the step number. Because of this new feature, we changed *Tilbage* (Back) to *Forrige* (Previous) to better illustrate the functionality of the command as described earlier in this chapter.

To make the state diagram easier to overview as a user, we chose to eliminate the initial state as it does not add any critical information to the user. The new state diagram is shown on Figure 13.1b and the state diagram display to the user is shown on Figure 13.1a.

After the test of this third STT Prototype, we considered it ready to be tested at Aalborg Hospital with the surgeons as the test users, and therefore confirmed our planned test date, the 14th of May:

Project View: Project Planning

Final test at Aalborg Hospital the 14th of May confirmed.

14

Demonstration and Final Tests of the Prototypes

This chapter describes how we demonstrated and tested the prototypes with the surgical staff at Aalborg Hospital and the results and feedback we got from the tests.

On the 14th of May, we met at Aalborg Hospital at 7:30 in the morning to join some of the surgical staff of the Department of Urology, more specifically Johan Poulsen, Jane Petersson and Grazvydas Tuckus. We joined them at the animal facilities of Aalborg Hospital, where they educated other surgeons on the use of the *da Vinci*® Surgical System and set up our prototypes for use in the operating theatre. The operating theatre at the animal facilities can be seen on Figure 14.1.

14.1 STT Prototype

Before setting up the prototypes in the operating theatre near the animal facilities, we demonstrated the STT Prototype to Johan Poulsen. He tested it in a nearby room rather than the operating theatre as it was currently occupied and we could easier explain how to use it because we had the room to ourselves.

After explaining the functionality of the prototype, we set up a process template that described the steps in the surgical procedure to be marked up

Figure 14.1: The operating theatre at the animal facilities

while performing the procedure on the pig. The template described the removal of lymph nodes on a pig:

- Open back layer
- Remove lymph nodes
- Identify nerves
- Inspection of surgery

We did not have access to our text to speech module (see Chapter 11) because it requires an internet connection. The module was necessary to create voice feedback based for the process template. Instead the feedback only specified the step number while e.g. selecting a step, instead of giving a short description of the step as it did in previous prototypes.

14.1.1 Test with the Chief Surgeon

We had Johan Poulsen use the tool to markup the "surgery". He completed the markup without any errors and was surprised speech recognition worked so well.

As Johan Poulsen was teaching new surgeons how to perform RAS, he did not have time for a thorough interview. Instead, we asked him a short set of questions to get his thoughts on the prototype and the general idea of using speech recognition to markup surgeries:

- *What do you think of the precision of the recognition?*
  The precision is good, the system recognized all I said.
- *Did you always know which step and state the tool was in?*
  Yes, and the system is easy to use and logical.
- *Did you get the necessary feedback from the system? Did you get too much or too little? Did you miss visual response?*
  The feedback was good, and it is good if it also gives a short description of the step as feedback.
- *Are the commands for navigating between steps logical?*
  Yes.
- *Are the commands easy to learn?*
  Yes, but I might have to use the state diagram once in a while to remember all of the commands.
- *Are the commands useful? Should we add or remove any commands?*
  Next and Previous might not be needed, saying Step x is probably enough.
- *Do the names of each command make sense compared to the functionality of the commands?*
  Yes.

Figure 14.2: Chief surgeon Johan Poulsen tests the STT Prototype during a surgery.

After testing in the nearby room, we took the STT and Pedal Prototypes to the operating theatre to the tested while conducting test surgeries on a pig. To test out the system, we reused the template described above, and Johan Poulsen then used the system while removing the lymph nodes of a pig. Where the test in the nearby room went smoothly and without any missed recognitions, the ambient noise in the operating theatre (which was a lot louder than the real operating theatre) caused the system to miss most of the commands given unless Johan Poulsen yelled the command. A picture of Johan Poulsen testing the STT Prototype is shown on Figure 14.2. The same was the case afterwards when Grazvydas Tuckus tested the system during a test surgery on the pig.

While this is problematic, it is worth noting this is most likely a hardware problem which we address in Chapter 17 and not a problem with the speech recognition system itself. Johan Poulsen had however seen the system working fine in the nearby room, and he understood the problem and believed that it was indeed solvable.

14.1.2 Test with the Surgical Assistant

We took the prototype back to the nearby room in order not to be disturbed by the ambient noise in the operating theatre. Here we tested it on surgical assistant Jane Petersson, and like Johan Poulsen, she completed the markup without any errors nor missed recognitions. Because that Jane did not have time for a thorough interview, we asked her the same questions as we asked Johan Poulsen:

- *What do you think of the precision of the recognition?*
  It works well.
- *Did you always know which step and state the tool was in?*
  Yes, and the status command is very useful for ensuring you know the state of the tool.
- *Did you get the necessary feedback from the system? Did you get too much*

or too little? Did you miss visual response?
The feedback is good.

- *Are the commands for navigating between steps logical?*
  Yes.
- *Are the commands easy to learn?*
  Yes, but I might have to use the state diagram once in a while to remember all of the commands.
- *Are the commands useful? Should we add or remove any commands?*
  Yes. No.
- *Do the names of each command make sense compared to the functionality of the commands?*
  Yes.

Figure 14.3: The pedals near the surgeon assistant Jane Petersson

Figure 14.4: The monitor giving visual feedback near the monitor with the camera feed

14.2 Pedal Prototype

When Johan Poulsen tested the STT Prototype while removing the lymph nodes, Jane Petersson used the Pedal Prototype to markup the surgery at the same time. The test went very well, and she found it very easy to use the system even when assisting the surgery at the same time. The Pedal Prototype setup in the operating theatre is shown on Figure 14.3 and Figure 14.4.

In the real operating theatre, the distance between the surgeon assistant and the monitor is between one meter and one and a half meter. In the operating theatre in the animal facilities the distance is three meters. To offset this issue we increased the font size and button size such that the visual feedback was visible from that distance.

Johan Poulsen tried the Pedal Prototype after wards and was impressed by how simple and well-functioning the prototype is, and the visual feedback seemed to work fine, giving Jane Petersson a good indicator of the state of the prototype.

15

Discussion

This chapter discusses our work described in this master thesis. First we look at the results of the collaboration with the Department of Urology in terms of prototypes and the functionality that they provide to the department. We then discuss our method, the results of using Essence, Prototyping, Surrogate Challenger and Users, and problem of mapping problems from one domain to another.

15.1 Collaboration with the Department of Urology

During our previous semester we came to the conclusion, in collaboration with the Department of Urology, that a system to collect statistics from surgeries could help them improve their overall efficiency and quality. During this master thesis, we have further developed that idea and suggested a way of marking up surgical processes using speech recognition and pedals.

The project course is depicted on Figure 7.3 (page 54), and shows that we only had two meetings with the Department of Urology. In the previous semester we had three short meetings and a day where we visited the operating theatre and observed two surgeries. A total of six meetings were held where we shared knowledge, discussed ideas, working environment and complications etc. Many ideas were created during these meetings and we further developed them in the Software Innovation Research Laboratorium on Aalborg University.

As mentioned, our work resulted in two prototypes that were used as proof of concept and demonstrated to the Department of Urology on the 14th of May 2012. Both prototypes is able to markup the beginning and end of each step in a surgical procedure. These data can be used to split the video feed of the endoscopic camera, and the clips can be stored together with the statistics for future analysis.

Based on these prototypes, the Department of Urology should be able to decide whether or not they are interested in continuing working with the ideas and further develop the prototypes into a real system that is usable during real surgeries. Or we can decide whether or not there is a foundation for establishing

a company, either in collaboration with the Department of Urology, or alone.

15.2 Method

As described throughout this master thesis we have investigated our research questions described in Section 1.1. Our method has been inspired by Essence with focus on finding the best ideas and proving these to the Department of Urology.

Given the nature of this project, where the focus is on being creative and innovative, we have not had a *strict* method that dictated when to do what and what artefacts (e.g. documents, requirement specifications, diagrams) to produce during the process. This was a deliberate decision, because putting creativity into boxes and under strict supervision *will* impede the process. In order to be creative, there must be room for improvisation, also in the process. Essence definitely supports this view as Ivan Aaen designed Essence to be light on procedures and methods and heavy on structures. Essence does not dictate *how* to do it and *what* artefacts to produce, but it provides means of organizing the project, information and knowledge, and suggests artefacts that one could choose to produce during the process. It is then up to the team to decide what artefacts to produce in what contexts.

We chose to exploit the power of prototypes as the primary artefacts in our project, described in Chapter 7. But also our tests act as artefacts (experiments) because their focus is to evaluate the ideas; trying to find the best possible ones before testing it with the customer. During these tests we observed the Surrogate Users, collected errors and discussed the prototype with the Surrogate Users in order to evaluate the ideas manifested in the prototype.

The prototypes also helped us communicate with the Department of Urology during meetings, where they acted as *boundary objects* thus ensuring a common understanding of the present circumstances and the problems that the prototype were to solve. For example, during some of the first meetings with the surgical staff during the previous semester, the discussions often drifted away into other subjects, less important to the idea. Having a prototype ensured that focus was kept on the problems that it was supposed to solve.

15.3 Views and Roles

Using Essence while exploring the problem domain and generating ideas helped us organize and share discoveries and knowledge about the problem domain, and improve the overall creativity.

Through the whole process we made use of the different Views described in Essence, although Paradigm and Process View were the ones used most. This helped us organize our discoveries and knowledge within the problem domain, and because we were working in SIRL, we had the Views visible to us all the time. This enhanced communication and made it very focused because the discussions mostly originated from ideas and scenarios described on the Paradigm View. For example, while discussing scenarios, either with or without our Surrogate Challenger, new ideas came up that were instantly noted on the Paradigm View. While discussing and evaluating ideas, the results were noted on the Process

View in the form of advantages and weaknesses (PMI) for each idea. While PMI was used primarily for the "smaller ideas", SWOT were used more thoroughly on the "bigger ideas", such as the idea of performing the markup using pedals (while "smaller ideas" are about how to solve *that* problem). Using these tools forced us to think about advantages and weaknesses for each idea, and as a result of that, we continued working only with the ideas that was evaluated to be best suited.

The Product View was used to note down technical solutions and discussions such as state diagrams and class diagrams, and evaluations of the researched speech recognition frameworks. We were however not as focused on using this View as we were with the Paradigm and Process Views. This was most likely because most of this project has been concerning idea generation and evaluation, and as the project progresses and prototypes are getting more features, the focus will change from Paradigm and Process View to the Product View. Furthermore, architectural problems and design considerations were often decided upon between the two developers, right before implementing them, and therefore we did not use Product View as much as we would do with more developers in the team.

We do however definitely acknowledge the use of the Product View, but believe that it is more crucial when developing more complex solutions and not only prototypes which are simple and limited in nature, and when there are more team members to discuss technical considerations and implementation details. The same applies to the Project View, which we used to plan project deadlines, technologies to research, and meetings with the Department of Urology and our Surrogate Challenger. Initially, it was also used for giving an overview of the idea, its challenges and how to solve these.

During the beginning of the project course, while studying and starting to work with Essence, we found it difficult to decide which tools, mentioned in Essence [Aaen, 2012] (e.g. SCAMPER and Six Serving Men), we should use during the project. Because of this, we ended up using only a few simple tools such as PMI, SWOT and the Toulmin Structure. These tools were of great use, but we are still unsure whether or not other tools would have helped us even further. Furthermore, it was difficult in the beginning to remember where all the information belonged on each View. We solved this by having a description of each View next to the actual View in the SIRL lab. This made it easier, and eventually it became easier to remember, and then we really saw the power of having these Views. Thus it was difficult and to some extend annoying in the beginning, because it is different from how we normally work, but after learning how to use the Views and tools it became very supportive, and it is something that we would definitely recommend.

Looking at the Roles in Essence, they are described as roles that team members can "take on". We did not think of Roles this way because we did not find it necessary to do it so explicitly. Roles were shifting freely from team member to team member at all times depending on the context. For example, while exploring ideas on the Paradigm View, the Responder shifted in and out of the Child role while coming up with ideas and at the same time evaluating it with a technical perspective (Responder). The Responder also took on the Challenger role (not to compare with the Surrogate Challenger role), evaluating whether or not the idea was realistic based on our knowledge of the problem domain.

Obviously, this is primarily due to the fact that the team only consists of

two persons. This makes it impossible to fill out all roles independently, and especially the Anchor role which was left out—its responsibilities were delegated to the Responders. The biggest problem is that the Challenger role is partly filled out by the Responders. While we gained knowledge about the problem domain through meetings and by visiting the operating theater, we will never have the same knowledge as a real Challenger (for example the chief surgeon). Thus we might approve or discard ideas that the Challenger would have decided differently upon. We tried to minimize this problem by introducing the Surrogate Challenger, discussed in Section 15.5.

15.4 Prototyping

During this project we incorporated Prototyping into our development process together with Essence. We have used prototypes as a tool for enhancing creativity and the overall innovativeness of the team.

This is in contrast to how prototypes are traditionally used in software engineering, where it is a process to support requirements engineering, and evaluation of implementations. Instead we look at prototypes from a more architectural perspective, and inspired by how Schön and Lim et al. talks about prototypes as a technique for traversing a design space, enhancing communication and knowledge sharing. All these elements improve idea generation thus enhancing the chances of generating more ideas that might result in a more innovative product.

Through prototypes we manifested ideas as physical objects. By looking at these objects and interacting with them, you see the problem domain in the light of this prototype, making it easier to reflect upon the situation—what the prototype solves, what it does not solve and what it can solve better. What happens in our mind when seeing such a prototype in relation to the problem domain, is that the person (designer, developer, Responder, Challenger, etc.) sees what is "there", makes his personal relations to it, and comes up with new ideas, be it changes, improvements or inconveniences. This is what Schön describes as a *seeing-moving-seeing* pattern [Schön, 1992] and it is experienced differently by each person working with the prototype. This is what makes the process so powerful; it generated ideas and views within our minds which were further discussed and new prototypes were built (or existing prototypes were modified). The process continued throughout the project course until we had prototypes that we thought the Department of Urology would be pleased with.

In order to get the most out of a prototype, it is important to consider how the idea should be manifested and what the goal of the prototype is. To support this, we have used Lim et al. description of *filtering* and *manifestation dimensions*. We took these dimensions into consideration when we developed our prototypes. For example, in the Pedal Prototype the *Resolution* manifestation dimension of the prototype was to test the feedback when a pedal has been pressed, and if this feedback was visible to the user without looking directly at the monitor by exploiting peripheral vision (the *Appearance* filtering manifestation). The Resolution manifestation dimension concerns the level of detail of the prototype, and for the sake of this example, if that dimension was not considered, the idea could have been manifested with a simple set of pedals and a paper prototype that was changed manually when the user pressed a pedal.

Because the accuracy of the markup and feedback is very important for the finished product, we decided to include feedback as an important detail in the prototype.

Considering the different filtering and manifestation dimensions helped us think about what the focus of the prototype is. This is very useful because it "guides" the users of the prototype, making them focus on what is important in the designed prototype. Returning to the previous example, the choice of focusing on the Resolution and Appearance dimensions made our minds more focused around those dimensions and different ideas came up on how to improve the feedback.

The same advantages can be exploited when using the prototypes to communicate with the customer, users or testers. Using the dimensions, the prototype can be designed to focus on *what* the developer wants to show or discuss with the customer or user, or *what* the testers should be concerned about while performing the tests. Thus we can say that prototypes can act as *boundary objects* [Star and Griesemer, 1989, Star, 2010], that also helps ensure that the team, customer and users understand each other, which is difficult to achieve with more formal presentations such as documents and diagrams.

15.5 Surrogate Challenger

Having an on-site customer available for the team is very useful because he is able to supply a lot of knowledge about the domain. that can be of great use when generating ideas and looking at possible solutions. However, like Inayat et al. documents in [Inayat et al., 2012], it is very common that the customer is not able to supply a full-time on-site customer, or Challenger in Essence.

Trying to cope with a missing Challenger, we introduced the concept of a *Surrogate Challenger* (SC). The SC is not intended to replace the customer, but as a supplement who is able to engage in frequent meetings where the team is located. It is important that the customer is still available because they have the best knowledge about the problem domain and in the end it is ultimately their decision of what is to be developed.

We chose our supervisor, Ivan Aaen to be our SC because he has some medico-technical background combined with knowledge on relevant technologies, process improvement, and system development.

The SCs responsibility is to challenge the Responders and their views on the problem domain. When working on a problem as a Responder, one might become very focused on a set of ideas thus having a hard time abstracting away from those, and seeing the problem from another, and possibly better, view. In this case, the SC can challenge the Responders, forcing them to think differently or about an entirely different scenario. This might be even more important when working with prototypes because the developers can be very focused on implementing details and optimizing performance that might not be necessary for the purpose of the prototype.

The primary goal of the SC is to provide a different view on the situation and thus engage discussions with this view. Depending on the SCs knowledge within the problem domain, he might also be able to decide whether or not some ideas should be discarded or continued working with. However, it is not only knowledge within the problem domain that makes the SC useful. Knowledge

within e.g. software and usability engineering is also very useful, especially when there is increased focus on developing prototypes. In this case the SC can contribute with technical suggestions and ideas, or usability improvements.

However, when the SC has increased knowledge within the same field as the Responders, the discussions might become very technical focused thus neglecting what is really important, the focus on generating ideas, trying to find the best possible solution. Actually this problem is not only tied to the SC, because when we remove the Customer from the daily discussions in the team, there will be more focus on the technical solutions. This problem is hard to solve completely, but it could be reduced by choosing a SC with less "technical" knowledge, and knowledge closer to the customer's domain.

Depending on the customer, it may be an advantage that the customer is not present during the whole idea generation phase of the project. The customer might have some prejudice against e.g. technical solutions and thus he might shoot down ideas that later could have turned out to be very good. He could also be very focused on a single technology and negative against others, thus forcing discussions to concentrate around his preferred technology. However, we must assume that this is not the case given the nature of these kinds of projects, where the goal is to be innovative, thus an open mind is critical in order to success.

15.6 Surrogate Users

It is crucial to test the developed prototypes in order to receive feedback on the ideas they present and technical aspects as well if needed, from the actual users. To receive the best possible feedback, the testing environment needs to resemble the real environment. This is especially important in our case, where the system is supposed to blend into their environment, not disturbing the daily work routines, and not removing focus from the surgery and the patient.

Ideally, we would have a operating theatre available that resembles the real operating theatre, but because this is obviously very resource demanding, Aalborg Hospital was not able to provide it. The same applies to the test subjects who, in our case, needs to be the surgical staff that performs surgeries and who are supposed to use the system on a daily basis. They are not able to engage in frequent tests because their capabilities are needed on the hospital to perform daily surgeries.

We believe that customers not being able to supply testing environments nor/or test subjects are far from rare and thus we have tried to perform tests, and received feedback through the use of Surrogate Users. It is not possible to receive as high a quality of feedback like if you were testing in the actual environment and with real test subjects. But we do believe that one could approach this quality if the tests are designed correctly and by choosing suitable aspects of the system to test. At some point it will be necessary to perform tests in the real environment, but in this way we can minimize the demands placed on the environment and test subjects.

But what is a Surrogate User (SU)? We describe a SU as a test subject who is able to engage in frequent tests of the system, or prototypes hereof. The SU does not necessarily need to have any deeper knowledge to the problem domain, but if he does, it might give better feedback because he will be able to relate it

to the domain.

It depends on how the testing environment is designed. It should be designed in such a way that the SUs perform tasks *similar* to that of the real environment. Because the real users are competent at performing their daily tasks (because of their experience and professionalism within their field), the tasks that the SUs should perform, should be tasks that the SUs are competent at. While having a SU, if he is not a surgeon, conduct a surgery is problematic for several reasons, it will also be a task that the SU is probably unable to complete at all.

We call this *mapping* of the problems, because the realm problem solvable by the real users is mapped to a "problem" solvable by the SUs.

For example, we see this mapping of problems when training astronauts where the problem of training in weightless space is mapped to training in water tanks on earth, because it is practically impossible to set up a training programme for astronauts in space.

During our project, we used associate professors, Ph.d. students, and software engineering students as Surrogate Users because they were able to engage in frequent tests. Thus our SUs did not have any experience within the medical field and so we designed a process of tasks that was solvable by our SUs. In case of the Speech-to-Text (STT) Prototype, the SUs were supposed to markup this process using voice commands while they solved the tasks. Each task consisted of a dot-to-dot drawing that was relatively hard to solve with precision because of the skewed camera angle and slower feedback compared to looking directly at the paper. Thus the SU was forced to keep focus on solving the task, while performing the markup.

Here we map the problem of a surgeon performing the markup while keeping focus on the patient, and we were able to test if the

- SU was able to keep focus on the tasks while using the prototype;
- voice commands made sense in the context of marking up a non-linear process;
- the audio feedback was sufficient; and if the
- accuracy of the speech recognition software was sufficient.

While the problem of drawing a dot-to-dot drawing is not comparable to that of a surgery, it still helps us in deciding whether or not a solution using speech recognition is realistic. The idea will of course have to be tested during a surgery before it can be decided whether or not it is usable of if it removes too much focus from the surgery.

What it gave us, was that we were able to find and test ideas that we could present to the Department of Urology with a greater chance of them being favorable towards the potential solutions, because we have thought about and tested scenarios important to them and the environment. This was confirmed when we tested our solutions, a STT and Pedal Prototype, with the Department of Urology. Despite the noise problems in the their testing environment, which is indeed solvable, they were very impressed with the ideas and prototypes. The chief surgeon Johan Poulsen was very pleased with both systems. He found the STT Prototype to be very intuitive; he appreciated the few and short commands. A minor downside was that he thought the feedback could be shorter and more

precise. He was also very pleased with the Pedal Prototype and the fact that it was even simpler than the speech prototype.

The chief surgeon expressed that he definitely could see this system being incorporated into the operating theatre and used daily for marking up surgeries.

16

Conclusion

In this master thesis we have studied how *Prototyping* can support a innovative software process based on Aaen's Essence framework. The goal of this process was to develop ideas, and prove these through prototypes, of a system capable of capturing statistics during robotically-assisted surgeries, in collaboration with the Department of Urology at Aalborg Hospital.

Our case has several issues we needed to handle to develop useful prototypes. The first issue is the lack of an on-site customer. Developers must realise that the agile concept of having an on-site customer is very resource demanding for the customer (especially the person who is the on-site customer), and if not taken seriously it constitutes great risk to projects depending on one. To cope with the fact that our customer, the Department of Urology, did not have the resources to supply a on-site customer and only had little time available for meetings, we introduced the concept of a *Surrogate Challenger*, who is able to take on parts of this role, but with increased availability to the project.

Furthermore, we did not have access to staff from the Department of Urology for testing early versions of the prototypes. To offset this issue we used *Surrogate Users*, users who do not have the domain knowledge nor training that the real users (the surgeons) have. Since the Surrogate Users do not have this knowledge, we chose to primarily test the technologies used in the prototypes and the intuitivity of the user interfaces.

Robotically assisted surgeries are very seldom in Denmark and only exists in a few operating theatres throughout the country. Therefore we were not able to use an actual operating theatre for testing, which required us to build simpler testing environments for testing the prototypes. While this limits the aspects we can test, we used *mapping* to minimize the impact of having no real test environment available.

We mapped the problem of the surgeon who needs to keep focus on the patient while using our system, to a simpler task of completing dot-to-dot-drawings. We created an environment with the requirements having *a*) full focus on the task; *b*) finger proficiency; and *c*) lack of tactile feedback as the surgeons experience. However, to a much lesser extent to make the tests possible to complete for our Surrogate Users.

The ideas we came up with while exploring the problem domain were evaluated and matured, and eventually manifested in a prototype. To further evaluate

the idea or technical aspects of the prototype, we conducted simple tests of the prototype with Surrogate Users. We tested the final prototypes together with the Department of Urology in an operating theatre used for educating surgeons, and during the process few meetings were held with their chief surgeon and a surgeon nurse.

The first research question concerns how prototyping can support the Essence innovation process. We studied how prototypes are traditionally used during software projects as a tool for identifying and evaluating requirements. Prototypes are indeed useful in this context, but as we have shown during this project, their real strength lies within their capabilities of enhancing the way in which we, as designers, organically and evolutionarily learn, discover, generate and refine designs. Looking at a problem through a prototype, each individual makes his personal relations to it, sees the problem space in a new light, and possibly comes up with new ideas, be it changes, improvements or inconveniences to what the prototype tries to solve or promote.

A prototype acts as a point of origin for the discussion around it, and if designed correctly the designer can guide this discussion around parts of the idea for which the designer wants feedback on. The prototype acts as a *boundary object*, ensuring that the team, customers and users understand each other. Something that is difficult with more formal presentations such as documents and diagrams. To design a prototype that reflects the intentions of the designer, we suggest using Lim et al. filtering and manifestation dimensions. Looking at prototypes from this view makes them very suitable for Essence as they promote different views of the problem space. By viewing the problem space from several views, idea generation is increased and eventually enhances the chance of developing a more innovative solution. They enhance discussions and understandings on the team as boundary objects and by providing different views of the problem space.

An example of this is the Speech to Text Prototype, where one view of the problem space is the exactitude of the audio feedback and another view is the implementation of states and how this makes it easier for the user to follow the surgical process with the tool.

The second research question concerns whether or not it is possible to replace the on-site customer with a Surrogate Challenger in cases where the customer is not able to supply one. Through our experiment, we were forced to partly replace the on-site customer with a Surrogate Challenger, which we did with great success. It is however important to consider *what* the Surrogate Challenger should be involved in, as he does not necessarily have the knowledge and experience as the customer, and ultimately it is the customer who makes the final decisions.

Thus the Surrogate Challenger should not *replace* the customer, but we have shown that he is a very valuable addition to the team when it comes to technical discussions, but also discussions around the problem domain where he is able to supply a different view on the situation and thus *challenge* the developers. This challenge is what makes the Surrogate Challenger very valuable. Because he comes from outside the team, he does not have his mind focused on a particular solution or technology that might impede the diversity of the generated ideas. Coming to the team, he is introduced to what they are working on, for example with a prototype, and then he shares his ideas that might come up.

The second research question also concerns whether or not it is possible to

use Surrogate Users instead of real users. Using Surrogate Users allowed us to perform frequent tests of the prototypes; something that we could not have done with real users in our experiment. Being able to perform frequent but simple tests of the prototypes allows for valuable feedback from the Surrogate Users, thus we were able to narrow down some of the problems of the manifested idea.

As with the Surrogate Challenger, it needs to be carefully considered *what* the Surrogate User is supposed to test. Because the tests do not take place in the systems real usage context and without real users, only certain aspects of the idea can be tested. We suggest testing technical aspects of the prototypes that are not strictly context dependent. In order to test the more context dependent parts of the prototype, we looked into mapping problems from the problem domain to a domain familiar to the Surrogate Users—the third research question.

In order to create this mapping, one must carefully design the *test-context* in which the prototype has it usages. The test-context is what maps the problems from the customers domain to a domain that fits the Surrogate Users. Both the test-context and the prototype needs to be designed in such a way that the Surrogate Users focuses on the problems intended by the designer. More specifically, we designed our test-context so that the Surrogate Users was forced to focus on the test-context while using the prototype (the surgeons needs to focus on the patient while using our system).

Although our mapping worked—confirmed by our final test with the Department of Urology in a close to real context—it needs to be investigated further in order to conclude something general. We have shown that it is indeed possible, but that it depends a lot on the context and the system to be developed. We do not believe that Surrogate Users and test-contexts can replace the real users and context entirely, but it is a less expensive way of testing that can greatly enhance the entire process with valuable feedback, if approached correctly by the designers.

The same goes for the usages of a Surrogate Challenger and Surrogate Users. While we have used both with great success in this project, it is necessary to study their limitations more thoroughly before they can be recommended for specific types of projects or scenarios. We believe the main reason that our process worked so well, is because our solution is an "add on" to their daily work that is not deeply integrated with the rest of the operating theatre. On the other hand, one of our goals is to collect data with the least possible impact on the way the surgical staff works. The use of surrogates might be more complicated if the problem space is closer integrated to their work, e.g. designing improvements of surgery tools, the *da Vinci*® Surgical System itself, but it is something that is still to be determined. A possible solution to this problem , could be to train the Surrogate Challenger within domain knowledge, e.g. by being an apprentice to the actual Challenger. The gap between the Challenger and Surrogate Challenger should be reduced through this "internship", making the Surrogate Challenger more useful for projects that integrate deeper with the domain.

Even if it turns out to be remarkably more complicated, we still believe that the use of a Surrogate Challenger is very valuable because sharing his view on the problem space is something that is always useful, and at least it inspires the team and the mind of each individual.

17

Future Work

Having answered our research questions and developed two prototypes that the Department of Urology saw a lot of potential in, there is still work to be done both with the research and development of a more complete product.

This chapter describes a few ideas that can be researched and developed further to enhance our research and supply the Department of Urology with a product able to actually perform the markup during surgeries and later analyse the statistics that the data constitutes.

17.1 Product

As this project ended with two proof of concept prototypes, one using pedals and one using speech recognition, much has to be done before the data collection and analysing can take place at Department of Urology.

First of all, the department has to decide whether they want to put more resources into the project, and hereafter the focus will most likely be to further develop the proof of concept prototypes to a product capable of being used in the operating theatre. With the markup product completed, the surgical staff can begin collecting data for use when the analysing tool is completed, and thereby have a foundation of data both for testing the analysing tool prototypes and for use when the analysing product has been developed.

The prototypes developed during this project have to be integrated into one system, such that the prototypes can be used at the same time. Marking the start or stop of a step with e.g. pedals will then give feedback both on the monitor and in the headset of the STT product and vice versa.

17.1.1 Pedal Prototype

The Pedal Prototype needs further development as it currently does not record the duration of each step. Furthermore, these durations should be used to mark points on the video records from the surgeries, effectively giving us a video clip for each step.

The monitor for displaying visual feedback has to be changed from the current LCD monitor as it is very large and difficult to place in the operating

theatre. It may be advantageous to use a tablet or monitor of same size, as it can be attached to the monitor that displays the feed from the endoscopic camera and thereby allow for greater exploitation of peripheral vision.

17.1.2 STT Prototype

The STT does record the duration of each step, but it still needs functionality to be able to apply these durations to split up the endoscopic camera recordings to build a video clip for each step.

As we discovered during the test in the animal facilities, the accuracy needs to filter out background noise. This can be fixed by e.g. getting a better microphone and/or using two microphones and noise filtering software-techniques.

The STT Prototype could also be tested for use by the surgeon assistant as it may prove to be a better solution than pedals.

17.1.3 Use of Touchscreen

Even though we abandoned the touchscreen prototype fairly quickly, we kept the touchscreen in mind for other uses in this project. One of the main ideas we worked on in the end of the semester is the use of a large touchscreen in the operating theatre to supply several features:

- General overview of the surgery, e.g. show the current state of the markup
- Markup of pre- and post-surgery steps, e.g. preparation of patient and cleaning of the equipment
- Markup of special criteria (described as classification in the prototype chapters), e.g. if the patient has previously had a surgery in the same area as for this surgery
- Entering of information that is currently written on paper and afterwards added to patients electronic health record ("Elektronisk Patient Journal" in Denmark)
- Access the analysis functionality, e.g. watch statistics from other surgeries and watch videos from similar surgeries considering classifications (e.g. find similar surgeries where the patient have had previous surgery in same area).

This touchscreen could also be used for markup. This allows the nurses to markup when the surgeon and surgeon assistant is busy conducting the surgery. Whenever the surgeon marks up a start or stop of a step, the visual feedback on the screen of the surgeon assistant will update to show the added markup.

17.2 Research

The following describes ideas that could enhance our research.

17.2.1 Prototyping in Essence

We see a huge potential in the use of prototypes in Essence as prototyping allows for idea generation, maturation and knowledge sharing, all of which are important properties in Essence.

If Essence is to integrate the use of prototyping, it is necessary to give a clear overview for potential Essence users, so they know when to use certain types of prototypes depending on the challenge they are working on and which dimensions they are focusing on.

17.2.2 Surrogate Challenger and Users

It is evident to investigate if Surrogate Users can be used in other areas than testing which we have experimented with during this project. While we were interested in Surrogate Users and Challengers because they were more available than our users, it would be interesting to determine which other qualities makes the Surrogate Users useful in a project.

It may be useful to investigate theories such as the seven intelligences described by Howard Gardner [Howard, 1983] and investigate if ensuring Surrogate Challenger and Users have comparable intelligences in the same areas as the actual challengers and users. Finding a way to exploit this might make the surrogates more useful. For this project, the users probably have high body-kinetic and spatial intelligence as the surgeons have to be good at using their hands while navigating inside the patient. Seeking Surrogate Challengers/Users with same level of intelligence in those areas may be helpful when testing prototypes and especially when mapping the problem domain to a domain the Surrogate Challenger/Users can easily adapt to.

17.2.3 Mapping of Problems

When designing tasks for the Surrogate Users, precise mapping is needed to ensure the tasks are solvable for the Surrogate Users while still representing the tasks of the real users. To enhance this mapping, a set of properties could be made to categorize tasks and thereby find other tasks sharing the same kind of properties.

For this project, the main properties are the constant focus on the patient, the high level of finger proficiency and spatial abilities, and therefore we mapped this task of conducting a surgery to a task that requires the same properties, however in a much simpler fashion to offset the training and high skill of a surgeon.

As with the Surrogate Challenger/Users, researching into intelligences and using them for mapping problems from one domain to another may be useful for effective and more precise mapping.

Bibliography

[Aaen, 2008] Aaen, I. (2008). *Essence*: Facilitating software innovation. *EJIS*, 17(5).

[Aaen, 2012] Aaen, I. (2012). *Essence*: Team-based software innovation. [In-progress, unpublished].

[Alexander, 1964] Alexander, C. W. (1964). *Notes on the Synthesis of Form*. Harvard University Press.

[Beynon et al., 2001] Beynon, M., Nehaniv, C., and Dautenhahn, K. (2001). *Cognitive Technology: Instruments of Mind : 4th International Conference, Ct 2001, Coventry, Uk, August 6-9, 2001 : Proceedings*. Lecture Notes in Computer Science. Springer.

[Boar, 1984] Boar, B. H. (1984). *Application prototyping: a requirements definition strategy for the 80s*. John Wiley & Sons, Inc., New York, NY, USA.

[Brooks, 1987] Brooks, Jr., F. P. (1987). No silver bullet essence and accidents of software engineering. *Computer*, 20(4):10–19.

[Buxton, 2007] Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Connell and Shafer, 1989] Connell, J. L. and Shafer, L. (1989). *Structured rapid prototyping: an evolutionary approach to software development*. Yourdon Press, Upper Saddle River, NJ, USA.

[Dybå and Dingsøyr, 2008] Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859.

[Dyba and Dingsoyr, 2009] Dyba, T. and Dingsoyr, T. (2009). What do we know about agile software development? *Software, IEEE*, 26(5):6–9.

[Floyd, 1984] Floyd, C. (1984). A systematic look at prototyping. *Approaches to Prototyping*, pages 1–18.

[Howard, 1983] Howard, G. (1983). *Frames of Mind: The Theory of Multiple Intelligences.* Basic Books.

[Inayat et al., 2012] Inayat, I., Noor, M. A., and Inayat, Z. (2012). Successful product-based agile software development without onsite customer: An industrial case study. *International Journal of Software Engineering and Its Applications*, 6(2).

[Inc.,] Inc., I. S. The da vinci surgical system. `http://www.davincisurgery.com/davinci-surgery/davinci-surgical-system/`. [Online, Danish; accessed 20-November-2011].

[Jakobsen and Follin, 2011] Jakobsen, T. and Follin, M. (2011). *Software Innovation: Using Prototyping to Improve Idea Generation with Lead Users.*

[Johnson, 2010] Johnson, J. (2010). *Designing With the Mind in Mind: Simple Guide to Understanding User Interface Design Rules.* Morgan Kaufmann. Elsevier Science.

[Larman, 2004] Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition).* Prentice Hall PTR.

[Lettl, 2007] Lettl, C. (2007). User involvement competence for radical innovation. *Journal of Engineering and Technology Management*, 24(1-2):53 – 75.

[Lettl et al., 2006] Lettl, C., Herstatt, C., and Gemuenden, H. G. (2006). Users' contributions to radical innovation: evidence from four cases in the field of medical equipment technology. *R&D Management*, 36(3):251–272.

[Lettl and Hienerth, 2008] Lettl, C. and Hienerth, C.and Gemuenden, H. G. (2008). Exploring how lead users develop radical innovation: Opportunity recognition and exploitation in the field of medical equipment technology. *IEEE Transactions on Engineering Management*, 55(2):219–233.

[Lim et al., 2008] Lim, Y., Stolterman, E., and Tenenberg, J. (2008). The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15.

[Martin et al., 2004] Martin, A., Biddle, R., and Noble, J. (2004). The xp customer role in practice: Three studies. In *Proceedings of the Agile Development Conference*, ADC '04, pages 42–54. IEEE Computer Society.

[Medicin,] Medicin, D. Landets bedste til mandesygdomme. `http://www.dagensmedicin.dk/nyheder/aalborg-udfordrer-arhus-og-riget/`. [Online, Danish; accessed 20-November-2011].

[Naumann and Jenkins, 1982] Naumann, J. D. and Jenkins, A. M. (1982). Prototyping: The new paradigm for systems development. *MIS Quarterly*, 6(3):29.

[Nokes, 2003] Nokes, S. (2003). *The Definitive Guide to Project Management: The Fast Track to Getting the Job Done on Time and on Budget*. Financial Times Prentice Hall.

[Nonaka and Konno, 1998] Nonaka, I. and Konno, N. (1998). The concept of ba: Building a foundation for knowledge creation. *California Management Review*, 40(3):40–54.

[Parnas and Clements, 1985] Parnas, D. and Clements, P. (1985). A rational design process: How and why to fake it. In *Formal Methods and Software Development*, volume 186 of *Lecture Notes in Computer Science*, pages 80–100. Springer Berlin / Heidelberg.

[Rasmussen and Svendsen, 2005] Rasmussen, H. M. and Svendsen, T. M. (2005). *Large Vocabulary Continuous Speech Recognizer for Danish and Language Model Adaption*. Aalborg University, Department of Communication Technology.

[Rudd et al., 1996] Rudd, J., Stern, K., and Isensee, S. (1996). Low vs. high-fidelity prototyping debate. *interactions*, 3(1):76–85.

[Schön, 1982] Schön, D. A. (1982). *The Reflective Practitioner: How Professionals Think in Action*. Harper Collins.

[Schön, 1992] Schön, D. A. (1992). Designing as reflective conversation with the materials of a design situation. *Research in Engineering Design*, 3(3):131–147.

[Star, 2010] Star, S. L. (2010). This is not a boundary object: Reflections on the origin of a concept. *Science, Technology & Human Values*, 35(5):601–617.

[Star and Griesemer, 1989] Star, S. L. and Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science*, 19(3):387–420.

[Sygehus, a] Sygehus, A. Et år med robotkirurgi. `http://www.aalborgsygehus.rn.dk/Fakta+og+tal/Nyheder/EtAarMedRobotkirurgi.htm`. [Online, Danish; accessed 21-November-2011].

[Sygehus, b] Sygehus, A. Flere patienter tilbydes robotoperation. `http://www.aalborgsygehus.rn.dk/Fakta+og+tal/Nyheder/FlerePatienterTilbydesRobotoperation.htm`. [Online, Danish; accessed 21-November-2011].

[Sygehus, c] Sygehus, A. Landets bedste til mandesygdomme. `http://www.aalborgsygehus.rn.dk/Fakta+og+tal/Nyheder/Nyhed_10.12.10.htm`. [Online, Danish; accessed 20-November-2011].

[Sygehus, d] Sygehus, A. Nøgletal. `http://www.aalborgsygehus.rn.dk/Afdelinger/BoerneOgKirurgiCenter/Fakta+og+tal/Noegletal/`. [Online, Danish; accessed 21-November-2011].

[Sygehus, e] Sygehus, A. Om aalborg sygehus. `http://www.aalborgsygehus.rn.dk/Fakta+og+tal/Om+Sygehuset/`. [Online, Danish; accessed 20-November-2011].

[von Hippel, 1986] von Hippel, E. (1986). Lead users: a source of novel product concepts. *Manage. Sci.*, 32(7):791–805.