DESIGN SCIENCE BASED COMMERCIAL SOFTWARE DEVELOPMENT

MASTER THESIS SUMMARY

Jesper Lund Andersen, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, origo@cs.aau.dk

Kim Markfoged, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, fogeden@cs.aau.dk

1 INTRODUCTION

In this summary we outline the research contributions made in this master thesis project gathered in three articles; *Design Science Theories Applied as Software Development Frameworks, Adapting a Design Science Theory to a Software Development Process Model*, and *Applying a Design Science Process Model to Commercial Software Development*. We post three research questions and the motivation behind each of them. In order to find the place for our research in the field of DS we describe the major theory contributors, which used extensively, directly or indirectly, in our research.

Following the theory, we explain and justify the various methods used in our articles which we used to reach our results. These methods include: case study, development diaries, discovery experiments and a structured analysis. Next, we include summaries of each of our three articles in order to give a complete overview of the thesis before the concluding section.

In the conclusion we sum up the results of each research question and present observations made during our work that did not fit into any of the articles. Finally, we state exactly what the contribution of our master thesis is to the field of DS.

During our bachelor project we performed a small scale development project using Hevner et al. [2004] almost directly as our development model. Results were good, but due to the lack of structure in our process it was very unclear why they were good. Instead of blindly following our own, somewhat dated, opinion on DS in a software development context, we decided to revisit our own bachelor project as well as two other project which had the same research question but alternative DS theory bases.

Based on the motivation above, our first research question concerns the feasibility of investigating DS based software development methods on a large scale. Before this can be concluded a few sub-questions must be answered. Firstly we must examine whether it is feasible at all to investigate this on a large scale experiment, by processing the results from the bachelor projects.

Given that the response to the first question is positive, we need a much more structured approach to working with DS in a development context. This leads to the second research question of whether a DS research method can be adapted to a software development process model. The DS research used was Hevner et al. [2004] which included both a framework and guidelines, and the task was to figure out how (if possible) to adapt these into a process model which was true to the principles of DS and could easily be applied to software development.

If such a process model can be devised we can determine whether or not DS can be used in a software development project by performing an experiment with real commercial stakeholders and users using the process model.

This experiment must then be processed to ensure that not only was the development process true to the principles of DS and the process model but also, more importantly, if the development results in an artefact which can be used in a commercial context, thus determining if DS can be used for commercial software development.

Finally, the research questions for this master thesis are:

1. Is it feasible to investigate DS based software development methods on a large scale?
2. Can a DSR framework and guidelines be adapted to a software development process model?
3. Can a DS based software development process model be used in a commercial software development project?

2 THEORETICAL BASE

This master thesis has a high emphasis on DSR and especially the work of Hevner et al. [2004]. Therefore it is only appropriate to place Hevner et al. [2004] in context with other DSR contributions, and thereby also locate our place in the field.

One of the first publications to claim that systems development can generate knowledge was Nunamaker [1990], before that Design Science or Science of Design was widely considered disciplines belonging to the fields of engineering. In Nunamaker [1990] they argued that systems development could generate valuable results to the field of Information Systems (IS) research, by viewing it as a research methodology. Later in Walls et al. [1992], experimentation with combining the "science of design" and traditional IS research methods was continued. The article has since become very well cited because of the rigorous method used in the article, where they devise a design theory that was rooted in two well cited theories; Dubin [1978] and Simon [1981]. An example that there has recently been conducted research in the theory building is exemplified in Gregor and Jones [2007], which update the kernel theories developed by Walls et al. [1992] to better fit the theories from Dubin [1978] and Simon [1981].

March and Smith [1995] were the first to really combine the traditional knowledge building IS research (called natural science) with the knowledge using approach from the engineering disciplines (called Design Research). They devised a framework with four activities on one axis and four outputs on the other axis (see figure 1), where half of the activities was taken from natural science theory and the other half was taken from Design Research theory and similar with the outputs. The framework from March and Smith [1995] is rather crude and indefinite despite the high degree of innovation it represented which was symptomatic for a lot of the early DS research articles. It was not until Hevner et al. [2004] that the theory of DS was easily applicable, by supplying an extensive framework and simple (read: easy to understand) guidelines. Hevner et al. [2004] continues the idea from March and Smith [1995] to combine natural science and design research, and adds theory from Silver et al. [1995].

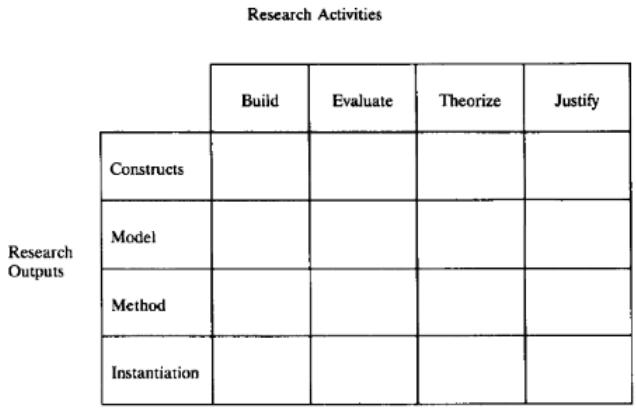

Fig. 1 Framework by March and Smith [1995].

Even though Hevner et al. [2004] is considered one of the most important contributions to the field, there are examples on alternative approaches. While Hevner et al. [2004] use a traditional science philosophy called positivism which, in simple terms, considers an information system a final solution to the phenomenon, there other approaches. An example of theory based on Hevner et al. [2004] is Peffers and Chatterjee [2007] which proposes a new framework which draws from several DS theories, and facilitates multiple entry points in the development process.

For our project Peffers and Chaterjee [2007] may be easier adaptable than Hevner et al. [2004], because of its introduction of multiple entry points. However, Hevner et al. [2004] is considerably

more cited and respected in the field of DS. Based on this, we chose to base our work on Hevner et al. [2004].

Our contribution positioned alongside Peffers and Chatterjee [2007] as theory extending Hevner et al. [2004].

3 METHODOLOGIES

When investigating the three bachelor projects in order to determine whether or not DS based software development methods was worth researching, we conducted a case study of already documented experiments. According to Yin [1994] a case study is "an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" and "relies on multiple sources of evidence". The case study was, as mentioned, based on three bachelor projects and even though they were rather limited in scope, they did employ the important factors in the experiment, i.e. different DS theoretical bases, developing software from a process derived from DS theory, evaluation of the process and theory. Furthermore, since there were several teams, there were multiple sources to support the final statements.

Finding a suitable method for our adaptation of the DSR framework and guidelines from Hevner et al. [2004] to a software development process model was proven difficult, since no generic method found suited the research properly. The method that was used was a step-by-step exposition of the components in framework, converting them into steps and utilities in a process model one by one. The strong accordance with Hevner et al. [2004] is argued rigorously to ensure that the final process model is in fact an adaptation of the framework and not simply a derivation of our own development experiences. A similar approach was applied to the guidelines, which were converted into validation criteria for development processes, intended to evaluate if a development project adheres to the principles of DS.

The experiment method used for the development is the discovery experiment method. Research through discovery experiments is useful when researching hypotheses which have a great degree of uncertainty with regards to the outcome. In a discovery experiment one works freely within the parameters of a predefined scenario while being observed, in order to reveal useful patterns, techniques, or methods which might not have been revealed in an ordinary scientific experiment in which parameters are changed systematically in order to provoke results (Andersen et al. [2007]). His is very similar to traditional natural experiments in which you do not alter variables to see the effect. In fact one of the few things that separate the two approaches is that often in natural experiments you observe situations in which you cannot manipulate variables, e.g. examining the effects of an earthquake on property values (Brunette [1995] and Murdoch et al. [1993]). Even so it is not hard to see the effect in values and defend this as the causal effect of the earthquake Shadish et al. [2001]. Documenting the experiment was done through the use of development diaries. These were maintained during the development experiment and had to follow a predefined template which was composed from the advices and experiences in Jepsen et al. [1997]. According to Jepsen et al. [1997] diaries adhere to the reflection-in-action which "... assumes that the different situations of professional practice are unique, complex, uncertain and even discordant." and "... it is often only possible to see or comprehend small fragments of the situation because situations are typically dynamic, consisting of complex networks of problems and conflicts.". Jepsen et al. [1997] further argue that the diaries contain descriptions, evaluations, and reflections on the actions in the systems development project and that this makes them act like links between the actual project and the reflections thereupon.

4 ARTICLE SUMMARIES

4.1 Design Science Theories Applied as Software Development

The first article concerns the study of three cases of small scale development experiments, which all use DS as a software development method. The introduction describes typical DS in broad terms, and then introduces the three student groups that acted in the development experiments as well as the documentation of these experiments. The section called "Experiment Design" describes the research method used in the experiments. This method is called Discovery Experiments and is a method suited for experiments where very little is known about which variables to monitor before the experiment has begun. This section also describes the process of the experiments and provides a description of the experimenters in terms of experience, skills, etc. Finally, the section describes the goals for each experiment. The theory section of the article briefly explains each piece of theory used in the three cases: Vaishnavi and Kuechler [2008], Walls et al. [1992], Gregor and Jones [2007], and Hevner et al. [2004]. The section "Developing with Design Science" goes through the cases one by one and explains their subject artefacts, theoretical foundation, adaptations and exclusions, and the actual development process of the experiment.

In "Results and Observations" the common tendencies and the results of the experiments are described. Each of the development experiments were successful in the sense that the goals set by the experimenters was reached. Other common tendencies include that experimenters all experienced that adaptations to the DS theory was necessary, that Initial Explorations and Knowledge Bases were beneficial to the projects, that iterative development was in the spirit of DS, and also all of the experimenters chose to implement the evaluation of DS as software tests. Most importantly, the final conclusion emphasizes that all of the experimenters have a positive attitude towards DS after the experiments and it is deemed that enough motivation is present to continue researching the matter on a larger scale.

4.2 Adapting a Design Science Theory to a Software Development Process Model

The second article focuses on the adaptation of the DSR method from Hevner et al. [2004] to a software development process model and begins by introducing the framework and guidelines from Hevner et al. [2004]. In the next section the article the framework is being processed step by step to convert it into parts of a process. Each component of the three columns of the framework has its purpose, tasks and output identified and the choices thereof are justified. Following the framework exposition the sequence of the process steps is outlined in a figure. Next, the guidelines are processed. The original explanation of each guideline from Hevner et al. [2004] is revisited and from this explanation the new interpretation is found.

To test if the process is ready for deployment, an experiment is devised. The scope, time span, and product of the experiment are determined beforehand. For each component of the process model the tasks which are to be included in this particular experiment are listed. Next, the experiment is evaluated against the guidelines to ensure that the experiment in its planned form still adhere to the principles of DS.

Finally, it is concluded that given a process model adapted from a DSR method it is in fact possible to design an experiment that follows the principles of DS. This naturally spawns the question if the experiment can be executed in a real-life scenario, which is underlined in the end of the article.

4.3 Applying a Design Science Process Model to Commercial Software Development

The third article concerns the method and analysis of a three month development experiment. The introduction gives a brief introduction to the experiment, including the documentation method and the process model used therein; the process model developed in the second article "Adapting a Design Science Theory to a Software Development Process Model". It also gives a brief explanation of the

nature of the developed artefact; an MMS based solution to help people with dyslexia. The method section explains the use of diaries as a development documentation method, as well as the analysis method used in the article. The analysis goes through each phase of the process model on a task level to examine the work done in relation to what the process model prescribes with an emphasis on what was gained from either following or deviating from the model. In order to cement the fact that the work done in the experiment is in fact true to the DS principles, seven verifying questions, corresponding to the seven original guidelines from Hevner et al. [2004], are answered.

The conclusion states that since a prototype for use at investor meetings has been successfully developed, the DS based development process can actually be used for commercial development. Moreover, it verifies two of the results from the first article "Design Science Theories Applied as Software Development Frameworks", that the knowledge base and initial research concepts are very useful. Finally, a side note is included stating that there may be benefit to gain from giving developers broader responsibilities. Future work includes looking into the mentioned responsibility question, comparing work done with the DS software development process model used in the experiment to work done with original DS frameworks, and finally experimenting with introducing the knowledge base concept to popular development methods such as SCRUM.

5 CONCLUSION

In the introduction of this master thesis summary, we made known our three research questions:

1. Is it feasible to investigate DS based software development methods on a large scale?
2. Can a DSR framework and guidelines be adapted to a software development process model?
3. Can a DS based software development process model be used in a commercial software development project?

We answered the first research question based on the case study of three bachelor projects containing development experiments using DS-inspired frameworks. The main conclusion of Andersen and Markfoged [2009a] is that "Even though the methods were not followed rigorously they were found useful, maybe because the experimenters were able to help sort out the aforementioned mess that results from mixing IS research with software development. This increased sense of perspective allowed the experimenters to investigate different parts of the project, without losing track of where the project as a whole was going". Furthermore, there was evidence that the knowledge base and initial research concepts stood out as positive components of DS in a development context. Based on all these encouraging observations we conclude that it was feasible to investigate DS based software development methods on a large scale.

The second research question was answered through proof of concept. In order to prove that it is possible to construct a software development process model from DSR framework and guidelines, we did so in converting the framework and guidelines by Hevner et al. [2004] into a process model. Apart from the construction of the process model, we investigated an already planned experiment in order to establish whether the application of the framework and guideline adaptation to this experiment was feasible. Andersen and Markfoged [2009b] present the following conclusion: "we argued that the tasks available in each phase of the framework fit the expected tasks of these in a satisfactory degree. Therefore we now conclude that the process model that resulted from the framework adaptation fits the experiment." and "Since we conclude that the constructed process model is true to the principles of the DS theory, i.e. Hevner et al. (2004) from which it is adapted, we can also conclude by proof of concept, that it is in fact possible to adapt a DSR method to a process model for software development.".

The analysis of our three month development experiment was used to answer the third research question. This answer is the culmination of the entire thesis and what we have been working towards since the first article. In Andersen and Markfoged [2009c] we conclude that "...this experiment has spawned a viable prototype, which is able to receive an image sent via MMS and convert the text contents of image into clear text, which can easily be converted to audio using commercial text-to-speech solutions. Furthermore, investor meetings are planned which supports the notion that DS can, in fact, be used in a commercial scenario". Moreover, the conclusion from the first article that the knowledge base and initial research concepts stood out as positive components of DS in a development context is revisited with the following result: "This has been confirmed during this experiment, where experimenters once again had no need to for theory exploration after the initial research phase. This prevents the flow development from being disrupted, first of all, by taking time from development to explore new theory but more importantly by altering artefact design due to changes in the underlying theory". The components that stood out are interesting because even though we conclude that one can use DS in commercial software development, it does not mean that we believe one should. Instead, we suggest experimenting with the integration of these components in already established development methods such as SCRUM or XP.

Over the course of our work on this thesis we did have other experiences than just the ones documented in the articles. These were left out either because they were not significant enough to make the cut or because they did not have any direct correlation to the research questions of the articles. One of these experiences was, however, briefly mentioned in the third article; it seems that there may be some benefit to gain from giving back some broad development responsibilities to developers. Back in the early days of the computer, programs were for the most part planned, programmed, tested, published, etc. by one person. However, as the requirements for each piece of software grew, more people had to be included, and this eventually resulted in the development of more rigid development strategies where the responsibilities for planning, programming, testing, publishing, etc. was divided between different roles, while responsibility for the project as a whole was moved upward in the hierarchy e.g. to a project leader, i.e. it professionals and software developers are sometimes reduced to nothing more programming machines, who blindly program features assigned to them from various directions such as the customer, the marketing department, management, other development departments etc. The lack of development resources in the development experiment of our third article forced the experimenters to take up more than one or two roles, but an entire handful of roles. This was thought to be a problem at first, but it turned out that the experimenters had the capabilities to handle the many roles and the fact that e.g. the programmers were the same people as those who interviewed users presented a lot of advantages in internal communication. This sparked the idea that utilising and expanding the skills of the developer may be more efficient, than having more bureaucracy in the form of steps in the organisation. The reason that it may be beneficial to have developers maintain more tasks in the process is that they are the ones who understand the developed product best, as they are the ones who has programmed it, they know the capabilities and limitations if the software. This could be an advantage in the interaction with stakeholders such as customers, who sometimes do not have the sufficient technical insight or problem domain overview to evaluate the software properly. This is of course not an argument for lesser stakeholder interaction, but should be seen more as an encouragement to challenge the stakeholders in their views and demands as only a system developer can.

6 REFERENCES

Jesper Lund Andersen and Kim Markfoged. Design science theories applied as software development frameworks. 2009a.

Jesper Lund Andersen and Kim Markfoged. Adapting a design science theory to software development. 2009b.

S. A. Carlsson. Towards an information systems design research framework: A critical realist perspective. DESRIST, 2006. Robert
Dubin. Theory building. Free Press, 1978.

S. Gregor and D. Jones. The Anatomy of a Design Theory, (TADT). Journal of the Association for Information Systems, 8(19), 2007.

Alan R. Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research, (DSISR). MIS Quarterly, 28(1):75, 2004.

T. Purdin J. Nunamaker, M. Chen. Systems development in information systems research. Journal of Management Information Systems, 1990.

Leif Obel Jepsen, Lars Mathiassen, and Peter Axel Nielsen. Using diaries. Reflective Systems Development, 1, 1997.
M. A. Rothenberger K. Peffers, T. Tuunanen and S. Chatterjee. A design science research methodology for information systems research. Journal of Managament Information Systems, pages 45–77, 2007.

S. T. March and G. F. Smith. Design and natural science research on information technology. Decision Support Systems, 1995.

M. S. Silver, M. L. Markus, and C. M. Beath. The Information Technology Interaction Model: A foundation for the mba core course, (ITIM). MIS Quarterly, 3 (19):361–390, 1995.

H. Simon. The sciences of the artifcial. MIT Press, 1981.
Joseph Walls, George Widmeyer, and Omar El Sawy. Building an information system design theory for vigilant eis. Information Systems Research, 3(1):36–59,

1992.
R.K Yin. Case study research: Design and methods. Sage Publications, Thousand Oaks, 1994.

W. Shadish, T. Cook, D. Campbell. Experimental and Quasi-Experimental Designs for Generalized Causal Inference (2e). Houghton Mifflin Company, 2001.

J. Murdoch, H. Singh, M. Thayer. Systems The Impact of Natural Hazards on Housing Values: The Loma Prieta Earthquake. Journal of the American Urban and Regional Economics Association, 1993.

D. Brunette. Systems Disasters and Commercial Real Estate Returns. Real Estate Finance, 1995.

DESIGN SCIENCE THEORIES APPLIED AS SOFTWARE DEVELOPMENT FRAMEWORKS

Jesper Lund Andersen, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, origo@cs.aau.dk

Kim Markfoged, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, fogeden@cs.aau.dk

Abstract

Design Science (DS) is becoming increasingly popular in Information Systems (IS) research due to its focus on solutions through development of artefacts. The focus of this article is to investigate three student experiments in which DS theories have been used as software development frameworks. We do so to determine whether it would be feasible to further investigate DS based software development methods. The three student experiments were conducted as discovery experiments and the software developed were, in all three cases, relatively small mobile applications. The case study performed in this article argues that it is in fact possible to apply DS theory to small software development projects. Furthermore, all three student groups were enthusiastic about DS as development frameworks, which motivates further investigation. Whether or not DS is suited for larger software development projects remains unknown.

INTRODUCTION

The Information Systems (IS) research method called Design Science (DS) is attracting an increasing amount of attention in the IS research community. In short DS is a research method focusing on the development of an artefact (i.e. software solution, invention, etc.) which solves a given real life problem.

Typical DS research consists of three phases: an initial thorough investigation of the given problem domain, the construction of an artefact to solve the problem, and an evaluation of, to which extent the artefact solves the problem. As opposed to traditional IS research methods, which tend to focus on theory instead of real world application, DS focuses on solving concrete problems, thus dragging researchers further towards industrially suited research results.

One could argue that especially the software industry would often benefit from a slightly more research-like approach to software development in order to better communicate knowledge both internally and externally. Innovative software companies working with new technologies could perhaps in an even greater extent, benefit from a method that emphasizes the development of not just software but also new knowledge as a part of the same process.

This motivates experimenting with applying DS a software development method. Three student groups at Aalborg University (AAU) attempted this for their bachelor projects during the spring of 2008. In order to simulate working as part of an innovative software team, all the groups worked with the development of mobile phone applications, which none of the participants had any significant experience in, prior to the experiment. Since they were all writing their bachelor in Computer Science, they had, prior to the experiment, all attended the same courses in software development methods e.g. agile development, Object Oriented Analysis and Design (OOAD), etc.

These three experiments are documented in the following three articles: Krogh et al. (2008), Sørensen et al. (2008), and Andersen and Markfoged (2008). These three articles are in further sections referenced to as *Case 1*, *Case 2*, and *Case 3*. The reason why the experiments in these articles are interesting is that all participating groups responded positively to using DS as a development method; citing Sørensen et al. (2008): "*This* [the experiment] *showed that it was possible to use DS methods to develop a mobile application. ... The developed application works satisfactory in relation to the requirements stated..."* and Andersen and Markfoged (2008): "*... DS seems to be a good choice for developers who work with new and unfamiliar API's and technologies*".

1 EXPERIMENT DESIGN

1.1 Discovery experiments

The method used in the experiments is called discovery experiment. The reason this particular method was chosen is that it is very easily applied to early experiments, in which parameters, conditions, etc. are not well defined yet. The method is often used when researching agile and/or innovative software development (Andersen et al. [2008]) and is therefore well suited for the type of development experiments carried out here. In a discovery experiment the lack of parameters, conditions, etc. force experimenters to keep an eye on multiple variables all the time, since no single parameter can be identified and altered as one would do in traditional scientific experiments. The lack of constraints is not all bad though, since this is believed to nourish innovative solution thinking. In order to perform a discovery experiment with an acceptable degree of scientific validity, the following should be defined before the experiment is started: the purpose of the experiment, the timeframe of the experiment, the number of people participating in the experiment, and experiment documentation method. The purpose of the experiment is to try to learn more about design science as a software development method; the rest of the experiment design requirements will be elaborated in the following sections.

1.2 The experiment process

The experiment participants started their work with DS through reviewing nine articles, as well as reading the AIS Design Science web page[1]. The nine articles in question were: Arnott [2006], Carroll [1993], Cross [2001], Gregor and Jones [2007], Hevner et al. [2004], March and Smith [1995], Marxt and Hacklin [2005], Purao [2002], and Walls et al. [1992]. Some of the articles were popular theoretical cornerstones within the field, whereas other articles in the selection were less cited but more focused on actually performing DS research. The articles were reviewed and discussed at a seminar attended by everyone involved with the research and focused on ensuring that the participants understood DS well enough to adapt one or more of the methods from the articles to be used as a software development method in their individual development experiments. The only three requirements posted to the three development teams, made up by the participants, were that development had to take three weeks or less, throughout development, written documentation of some sort had to be performed, and that the developed artefact should somehow be usable on or from a mobile phone. The latter of the two requirements was enforced to ensure that development would not end up being 'routine development' for the participants. At this point none of the participants had had any experience in mobile development so everyone was treading new ground. About mobile development in general can be said, that since mobiles are a relatively new development platform, programming technologies for it is still not as well documented as many other and moreover it is a development platform developing rapidly. Making sure that everyone was developing for an 'unknown' platform matched the innovative qualities of discovery experiments very well. Before development started the three participating groups were assisted in creating their own software development focused adaptations of the DS theory reviewed at the mentioned seminar. Midway through the development, a seminar was held at which the groups were given the opportunity to discuss their preliminary findings and opinions. When development ended, focus was turned from the development and experiments to documentation of, and reflection on, the research as the three groups of participants each received the help necessary to document their own findings in a research article.

[1] http://ais.affiniscape.com/displaycommon.cfm?an=1&subarticlenbr=279

1.3 The participants

All participants in the experiment were sixth semester computer science students at Aalborg University. On the sixth semester all courses in programming and software development the University has to offer have been passed. Furthermore, Aalborg University has a heavy focus on project-based education as up to half of the time of every semester is used on one project, carried out in groups of two to eight students. This means that even students without development experience from outside the university are relatively experienced software developers when they reach the sixth semester. The programming and software development courses mentioned earlier include Traditional Software Development (such as Object Oriented Analysis and Design, Mathiassen et al. [2001]), Functional Programming in C, Object oriented Programming in C#, and Agile and Iterative Software Development (such as XP and SCRUM).

1.4 The three development experiments (cases)

The three development experiments performed by the three groups of participants is what we will be using as cases, and will therefore from now on be referred to as cases. The first participant group consisted of four students who had chosen to develop an SMS based calendar service, which is able to send update notifications if any event in a common calendar is changed. The second case concerns the development of a SMS service to aid in sharing contact information, specifically enabling users to request the number of a certain friend from a number of people in his or her phonebook, the trick being, that the selected friends are only prompted to reply with a visit card if they actually have the number requested. This case was also carried out in a group of four students. The last student group consisted of only two students, who developed a piece of mobile software capable of sending images and other files using SMS's to carry the data and reassembling the data on arrival. This enables the user to send relatively large amounts of data over a protocol that, from most Service Providers in Denmark is free, instead of sending MMS messages which are expensive in comparison.

2 DESIGN SCIENCE THEORY

In this section we discuss the DS research theory that was used and adapted by the participants of the experiment to develop mobile software.

Vaishnavi and Kuechler [2008] have proposed the simple process model below, in which iteration is intended between the following individual process steps.

- **Awareness of Problem:** The identification of a problem that needs a (better) solution. The awareness of the problem results in an initial solution proposal.
- **Suggestion:** A suggestion to how the initial solution proposal should be realized. The suggestion results in a tentative design.
- **Development:** The realization of the tentative design, resulting in an artefact.
- **Evaluation:** An evaluation of whether or not the initial problem has been solved by the developed artefact. The evaluation of the developed artefact ends with performance measures such as efficiency, usability, etc.
- **Conclusion:** The conclusion should be based on the performance measures from the evaluation and should state the quality of the solution. The result derived from the conclusion is a measure of how successful the design research process has been.

Walls et al. [1992] originally defined an Information Systems Design Theory (ISDT) as *"A prescriptive theory which integrates normative and descriptive theories into design paths intended to produce more effective information systems"*. Gregor and Jones argue that the ISDT presented by Walls et al. [1992] is insufficient. From several DS related articles Gregor and Jones compile a list of shortcomings in the ISDT presented by Walls et al. [1992]. These shortcomings are presented below.

- It is unclear whether the application of an ISDT should result in a product, a process or both.
- The ISDT by Walls et al. [1992] is based upon Dubin's theory building components; however Walls et al. [1992] have omitted both *unit* and *System states* in their adaptation.
- Whereas the DS literature in general stress the importance of an artefact i.e. design instantiation, Walls et al. [1992] uses artefacts mainly as test of theory.
- A distinction between kernel theories for design processes and design products, respectively, may not be necessary.

Gregor and Jones [2007] propose the following eight components of design theory (2008):

1. **Purpose and Scope:** "*What the system is for?* The set of meta-requirements or goals that specifies the type of artefact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory."
2. **Constructs:** "A definition of the entities of interest in the theory."
3. **Principles of Form and Function:** "The abstract blueprint or architecture that describes an IS artefact, either product or method/intervention."
4. **Artefact Mutability:** "The changes in state of the artefact anticipated in the theory, that is, what degree of artefact change is encompassed by the theory."
5. **Testable Propositions:** "Truth statements about the design theory."

6. **Justificatory Knowledge:** "The underlying knowledge or theory from the natural or social or design sciences that gives a basis for the design (kernel theories)."
7. **Principles of Implementation:** "A description of processes for implementing the theory (either product or method) in specific contexts."
8. **Expository Instantiation:** "A physical implementation of the artefact that can assist in representing the theory both as an expository device and for purposes of testing."

Gregor and Jones [2007] argue that their first five components have a direct correlation to the theory building components presented by Dubin [1978]. Gregor and Jones [2007] then add another three components. These components are intended to function as crucial parts of any good DS Research. Hevner et al. [2004] also provide guidelines, which in general overlap the ones provided by Gregor and Jones [2007].

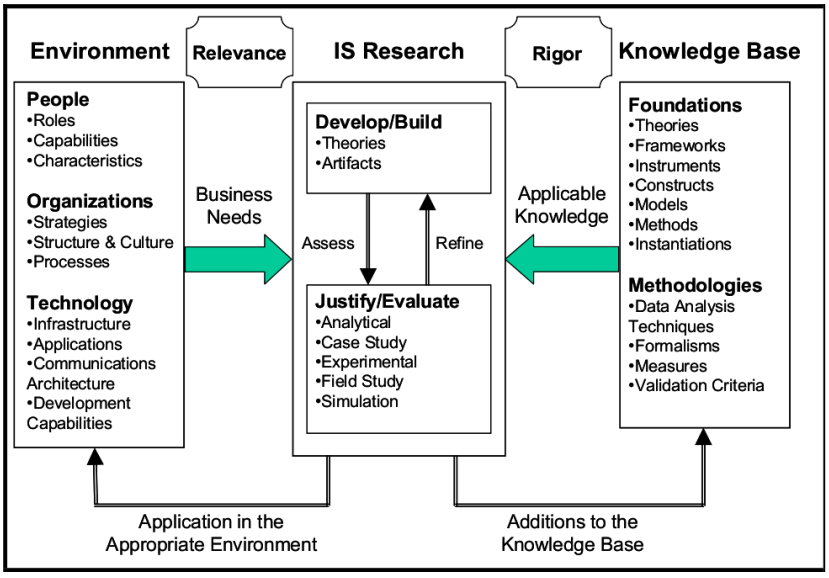

Figure 1. IS research framework, Hevner et al. (2004)

Hevner et al. [2004] argue that the principles of DS can improve behavioural science research and vice versa i.e. instead of the two approaches to research being incompatible the two can complement each other. They define the two approaches as follows: "*Behavioural science addresses research through the development and justification of theories that might explain or predict phenomena related to the identified business need.*" whereas "*Design science addresses research through the building and evaluations of artefacts designed to meet the identified business need.*".

Figure 1 illustrates the IS research framework by Hevner et al. (2004) which is based on both behavioural science and design science.

In Figure 1 the *Environment* column is inherited from traditional IS research where a need provides the motivation for the research, based on people, organizations, and/or technology. The *Knowledge Base* column is a combination of the knowledge base concepts from both behavioural science and DS. The *IS Research* column ties the framework together and Hevner et al. [2004] justify it as follows: "*The contributions of behavioural science and design science in IS research are assessed as they are applied to the business need in an appropriate environment and as they add to the content of the knowledge base for further research and practice.*".

As mentioned, Hevner et al. [2004] provide seven guidelines to DS research in the same manner as the design theory components by Gregor and Jones [2007].

Hevner et al. [2004] propose the following seven guidelines for DS research:

1. **Design as an Artefact:** "Design-science research must provide a viable artefact in the form of a construct, a model, a method, or an instantiation."
2. **Problem Relevance:** "The objective of design-science research is to develop technology-based solutions to important and relevant business problems."
3. **Design Evaluation:** "The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods."
4. **Research Contributions:** "Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies."
5. **Research Rigour:** "Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact."
6. **Research as a Search Process:** "The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment."
7. **Communication of Research:** "Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences."

Notice the similarities between guidelines above and the components by Gregor and Jones, e.g. the almost direct mapping between component 1 (*Purpose and Scope*) and guideline 2 (*Problem Relevance*).

3 DEVELOPING WITH DESIGN SCIENCE

3.1 Case 1 – calendar alert

The first case concerns the development of a service intended to notify users of any changes in a shared calendar. The notification is done via SMS. This development experiment is documented in Krogh et al. [2008].

3.1.1 Theory foundation

The theoretical base for the development by Krogh et al. [2008] is Gregor and Jones [2007]. However, the authors argue that in order to be applied as a development method, the components by Gregor and Jones [2007] need the structure of a framework. Figure 2 illustrates the framework created by Krogh et al. [2008].

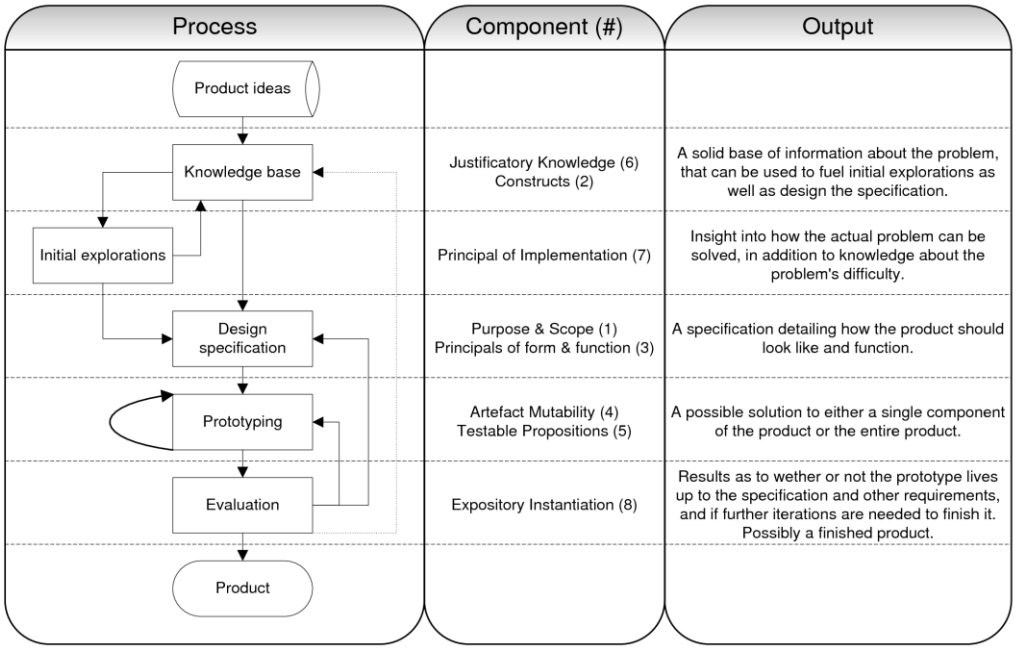

Figure 2 . Case 1 development framework by Krogh et al. [2008]

3.1.2 Development process

In the initial phase of development, the team was split up in two pairs. Each pair developed a module, which was intended to be integrated with the other pair's module later in development. One module was intended to send and receive SMS, while the other was to take care of fetching calendar data and keeping a log of communication etc.

Initial iterations

	SMS Send/Receive	Calendar/Logger
1st	Gather knowledge about how to program a send/receive functionality, and how to read message contents.	Figure out how to write to log and screen, and how to fetch XML data from an online calendar.
2nd	Gather information about deleting received SMS messages using the API.	Program a function to load an application configuration.
3rd	Program an interface for integration with calendar and logger.	Program an interface for integration with SMS send/receive module.

Table 1. Case 1 initial iterations

Following the initial iterations a working prototype was ready, and iterations on the product as a whole could commence.

Later iterations

Whole product
Finish error logging function.
Improve GUI layout.
Program support for plug-in functionality.
Reprogram calendar module to plug-in.

Table 2. Case 2 later iterations.

After having finished the later iterations the quality of the artefact was considered sufficient, thus ending the development experiment.

3.2 Case 2 – Phone number exchange

The second case concerns the development of a SMS based service to facilitate easy exchange of phone numbers between users. This development process was documented in Sørensen et al. [2008].

3.2.1 Theory foundation

The theoretical foundation is Hevner et al. [2004] and Vaishnavi and Kuechler [2007]. The experimenters replaced parts of the framework by Hevner et al. [2004] with the process steps from Vaishnavi and Kuechler [2007], as is illustrated in Figure 3.

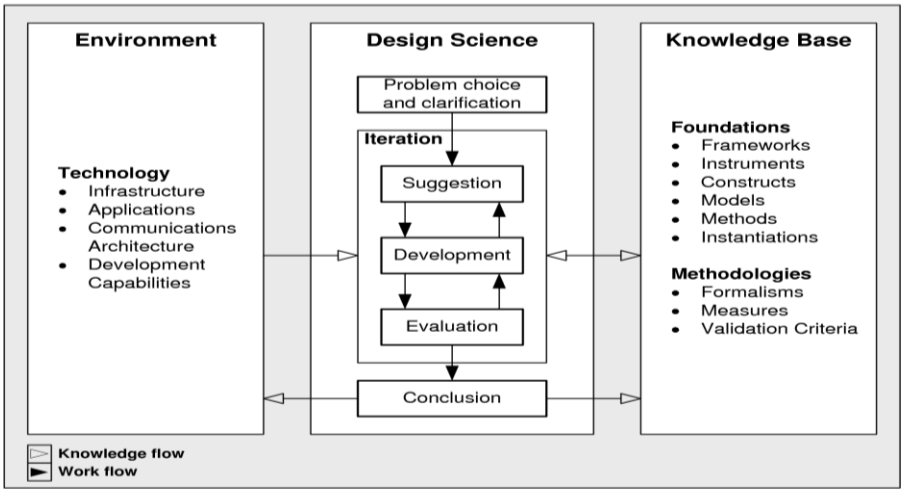

Figure 3: Case 2 development framework (Sørensen et al. [2008])

Moreover, some components were discarded from the framework, as they were deemed irrelevant by the experimenters, since they could not see any way to benefit from them in the very limited context of the development experiment. In addition to the adapted framework, the seven guidelines by Hevner et al. [2008] were also a part of the theoretical foundation.

3.2.2 Development process

The process was initiated with a research phase, where the following observations were made:

- Explore suitable technologies. Java ME fits needs.
- Program code examples for sending/receiving SMS messages, were added to the knowledge bases.
- Comparison of integrated development environment (IDE) was performed. Eclipse was chosen over NetBeans.

	Iterations
1st	Program load phonebook function. Design and program GUI. Program query function.
2nd	Optimize code from 1st iteration. Create a menu and implement send/receive SMS.
3rd	Allow selection between possible phone numbers, and implement request and reply communication, and investigate scrolling problem.
4th	Debug code from 3rd iteration. Add ”Settings”. Create ”Save number” functionality. Corrections of GUI.

Table 3. Case 2 Process iterations

After the fourth iteration the quality of the prototype was deemed sufficient and development was concluded.

3.3 Case 3 – SMS image transmission

The third experiment concerned the development of a mobile application capable of sending images with SMS as a data carrier. The documentation is supplied in Andersen & Markfoged (2008).

3.3.1 Theory foundation

Much like Case 2, the experimenters uses the framework by Hevner et al. (2004) in a customized form. In this case they choose to omit certain elements, but keep the all columns (i.e. Environment, IS Research, and Knowledge Base) of the framework. In addition to the framework, the following five of the seven guidelines were used: Guidelines 1. Design as an Artefact, 3. Design evaluation, 5. Research Rigour, 6. Research as a search process, and 7. Communication of research.

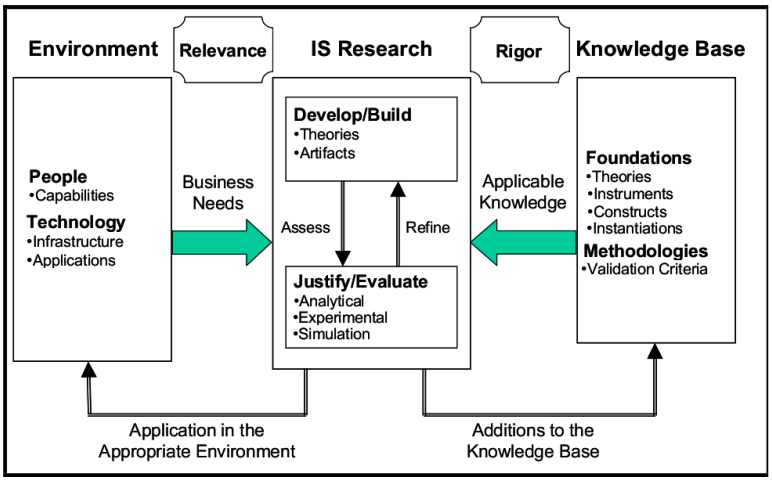

Figure 4. Case 3 framework (Andersen and Markfoged, 2008)

3.3.2 Development process

The first step was a brief analysis of the user group and technologies needed; the Environment column was used for this. The result of this initial analysis was a foundation on which basic features and the technical infrastructure of the application was based. Following this initial analysis the development iterations began. During the first development iteration it became clear that a custom GUI had to be created for the mobile application. During the next iteration knowledge about GUI implementation in Java2 ME was added to the knowledge base. Later in development this knowledge proved useful when developing the actual GUI for the application, as it saved important development time.

During initial development it was believed that a subset of Java2 ME called PIM could be used for file browsing, which was necessary for the application. However, further research showed that this was not the case.

When working with images, two approaches to loading images were explored and added to the knowledge base. A later change in the application design meant switching from one solution to another. This however meant very little extra cost, since information about both approaches was added to the knowledge base.

After having added knowledge about all components of the application to the knowledge base, developers seized contributing to the knowledge base in favour of using it as a specialized API documentation tool.

4 RESULTS AND OBSERVATIONS

4.1 Common tendencies

Hevner et al. [2004] was a popular choice in theory as two of the three groups of experimenters chose to use this as their main theory. This convergence of choice could be based on a number of things; the number of articles at the DS seminar was quite limited, perhaps Hevner et al. [2004] was 'advertised' more at the seminar, or maybe the experimenters just realized that the article is one of the most cited in the field. Because of the very loose guidelines the experimenters felt free to add wanted parts or omit parts of the theory when they did not feel they were appropriate for their experiment. All of the groups chose to do some kind of adaptation to the chosen theory before applying it. Whether this was necessary or not is uncertain. The omissions made were based on the opinion that certain modules could not be put into use in such a short experiment. It could be argued that the experimenters consciously or subconsciously omitted the parts of the theory which they were not entirely comfortable with, such as organization and user interactions.

All of the groups chose to base their research on design principles; some chose to adapt a set of design components into a framework, whereas others chose to inherit guidelines directly from their chosen theory. Another trend in the development was the use of iterative models as even the adaptation of Gregor & Jones [2007] design components resulted in an iterative framework. This may be because iterative development fits the DS theory chosen or it may be that the experimenters subconsciously chose this because it is what they are experienced in and feels comfortable with. The concept of a knowledge base, which is relatively unfamiliar in IS research, was applied to the development, but instead of thinking of the knowledge base as the knowledge of the entire community, the experimenters instead chose to create actual knowledge bases specifically for *their* project. Through the use of these specialized knowledge bases it became natural for the experimenters to use the first part of their time to gather the information needed to do most of the development. This is specified in the framework by Krogh et al. [2008] as *Initial Explorations*.

Finally, a large part of DS is the evaluation of one's own work, but in these cases evaluation was often interpreted as program functionality tests. This is probably the case because of the limited development period, yet it should be noted that it is not entirely in the spirit of DS.

4.2 Results

All three development experiments were completed painlessly and without encountering unsolvable problems. The goals set were met within the timeframe. All three applications passed the tests they faced in the evaluation phases and work as intended. All three articles agree that DS contributed with significant process awareness to the development. In fact Case 1 note: *"Improving the framework with the structured model was essential for understanding how to use the framework, as it was difficult to see a flow in the components"*. On the same topic Case 2 state: *"The general experience of using DS ... is that it makes the development process more explicit to the developers, and thereby makes the developers more aware of each aspect of the development."* and Case 3 agree: *"The research-like approach makes the development very controlled and focused in an intuitive way"*. Another common point is that the iterative development that the DS frameworks are said to encourage is a positive trait. For example Case 2 mention: *"The iterations also helped to keep development going..."* and Case 1 continue: *"The iteration process proved invaluable in extending the functionality of the program once the original goals had been met"*. As mentioned even though it was not an explicit part of the frameworks based on Hevner et al. [2004] all development teams utilized some sort of initial exploration process. Case 3 explain: *"The task of formulating the initial theories ... helped identify the challenges of the problem."* and especially Case 1 was impressed with their initial exploration: *"Another important point is the initial exploration, which contributes greatly to the development*

process." and *"The initial explorations from our model helped develop a good knowledge base, which eased the rest of the development."*. This enthusiasm for the knowledge base component was also shared by Case 3: *"...if developers make proper additions to the knowledge base, the risk of doing the same research twice can be minimized. Furthermore, having an explicit agreement to make additions to the knowledge base, developers are forced to share the information gained in the research phase."* and continue *"What seems to make DS shine in this context is the knowledge base..."*. Since mobile development is not as well documented in terms of tutorials and API documentation than older programming forms it is considered less safe than these; Case 3 comment that "*DS seems to be a good choice for developers who work with new and unfamiliar API's and technologies*." Another appreciated part of DS was its generic nature, which was deemed to make DS applicable in all sorts of projects: "*design science .....seems to be an optimization to any kind of development project*." (case 2). Finally, Case 1 comments on the value of DS in development projects: "*Design science, when applied to development projects, is very intuitive, which makes it easier to integrate. There might be some question as to whether design science is necessary, but we find that knowing what design science is and following the principles, helps to understand and ease the development process. This is achieved by supplying explicit ways to understand some steps in the process which could previously be implicit information and therefore harder to express*".

4.3 Limitations

The experiments provided positive results, but the validity hereof should be carefully considered. While experiments with students are common in IS literature, it cannot be guaranteed that real life software developers would respond in the same way as the students did when introduced to DS. Furthermore, the experimenters could be experiencing the Hawthorne effect, where the focus on experimentation affects the result. The documentation of the experiments could also be questioned; though documented, discovery experiments are by nature very loose and unnoticed variables could have had an effect on the results too. Finally, the experiments undertaken were very small, and the results can therefore not be expected to apply to even small commercial development projects.

5 CONCLUSION

The working title for our first article on DS was: “Can Design Science be used for Development?”. In that title laid, admittedly, quite a lot of scepticism, since we actually expected to reply negatively to that question. Initial research on design science gave the impression that it was shallow and somewhat naive. Ordinary IS research is, for the most part, research for the sake of science, whereas DS seemed much too focused on research for the sake of the artefact. Moreover, the guidelines, components, and frameworks provided, seemed to operate on a very abstract level.

The experimenters however, though experiencing a steep learning curve and some degree of culture shock, quickly adapted these abstract tools into less abstract development tools, easily filling the gaps where it was necessary. The mix between research and development matched the projects they had undergone earlier in their education, and some of what had earlier been looked at as a mess now had a structure and formulated terms which made it easier to discuss. Since it took a lot of adaptation and filling of the gaps to make DS theory work as development frameworks, it cannot be verified whether the experimenters fully understood DS before they began adapting it. This means that both intended and unintended adaptations were made, where the latter can be assumed to be caused by the experimenters misunderstanding the thought behind the original theory. The frameworks were not followed as rigorously as it could be expected in a scientific experiment, but note that e.g. agile methods are most often not followed rigorously in real life development either. Even though the methods were not followed rigorously they were found useful, maybe because the experimenters were able to help sort out the aforementioned mess that results from mixing IS research with software development. This increased sense of perspective allowed the experimenters to investigate different parts of the project, without losing track of where the project as a whole was going.

Even though the small experiments treated in this article provide no definitive results for the question of whether or not DS could be used in real life commercial software development projects, they do provide justification and motivation for further research. The fact that DS matched the development projects the experimenters had undergone as part of their educations, motivates to find out whether computer science students in general would benefit from mandatory courses in DS. Also, the positive results from experiments make it very interesting to study whether DS can be used as a structuring factor in more realistic development projects. Therefore, we have organized a much larger experiment with three months of development, actual stakeholders in the shape of a local innovation greenhouse, a group of dyslexic children, third party software vendors, etc. The intention of the project is to develop a prototype of a commercially marketable system.

Acknowledgements

We would like to thank Markus Krogh, Jais Heslegrave, Thomas Justesen, Daniel Korsgård, Morten Bøgh Sørensen, Anders Ejlersen, Michael Stampe Knudsen, and Joachim Løvgaard for letting us use their articles and results for our cases.

Jesper Lund Andersen and Kim Markfoged. *Design science applied as a development framework in unfamiliar programming environment*. 2008.

Morten Andersen, Søren Rode Andreasen, Lasse Bæk, and Philip Bredahl Thomsen. *Facilitating innovations in software development*. Master's thesis, Aalborg University, 2007.

Robert Dubin. *Theory building*. Free Press, 1978.

S. Gregor and D. Jones. *The Anatomy of a Design Theory*, Journal of the Association for Information Systems , 8(19), 2007.

Alan R. Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. *Design Science In Information Systems Research*, MIS Quarterly,28(1):75, 2004.

Markus Krogh, Jais Heslegrave, Thomas Justesen, and Daniel Korsgard. *Design science-appling the anatomy of a design theory to a mobile application*. 2008.

Morten Bøgh Sørensen, Anders Ejlersen, Michael Stampe Knudsen, and Joachim Løvgaard, *Using design science to develop a mobile application*. 2008.

Vijay Vaishnavi and Bill Kuechler. *Design research in information systems*. 2007. http://www.isworld.org/Researchdesign/drisISworld.htm.

Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, and Jan Stage. *Objektorienteret analyse & design (3e)*. Marko, 2001.

Joseph Walls, George Widmeyer, and Omar El Sawy. *Building an information system design theory for vigilant eis*. Information Systems Research, 3(1):36–59, 1992.

ADAPTING A DESIGN SCIENCE THEORY TO A SOFTWARE DEVELOPMENT PROCESS MODEL

Jesper Lund Andersen, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, origo@cs.aau.dk

Kim Markfoged, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, fogeden@cs.aau.dk

Abstract

Recent research has suggested that a Design Science Research (DSR) influence on software development could be beneficial (Andersen and Markfoged [2009a]). The focus of this article is to determine if it is possible to adapt a DSR method, in this case the framework and guidelines from Hevner et al. [2004], to a process model for software development. After a brief introduction to the DSR theory focused on Hevner et al. [2004], we examine each individual component in the framework, as well as the guidelines in order to convert these from a research oriented component to components in a framework for software development, e.g. part of the guideline "Research Rigour" is translated to a phase in development, during which initial research on existing technologies and solutions are studied, for potential reuse or inspiration. Next we describe an already planned software development experiment in which the adapted framework is to be applied. We then discuss the suitability of the framework in the context of that experiment. Finally, we conclude that it is possible to adapt a DSR method to a process model for software development and that the process model is ready for application in future research.

1 INTRODUCTION

In this article we interpret the framework and guidelines by Hevner et al. [2004] and adapt them to a software development context. This is done by identifying tasks and outputs for all the components of the framework, and interpreting the guidelines in order to use them as validation criteria to determine compliance with the principles of DS.

The framework by Hevner et al. [2004] is illustrated in Figure 1.

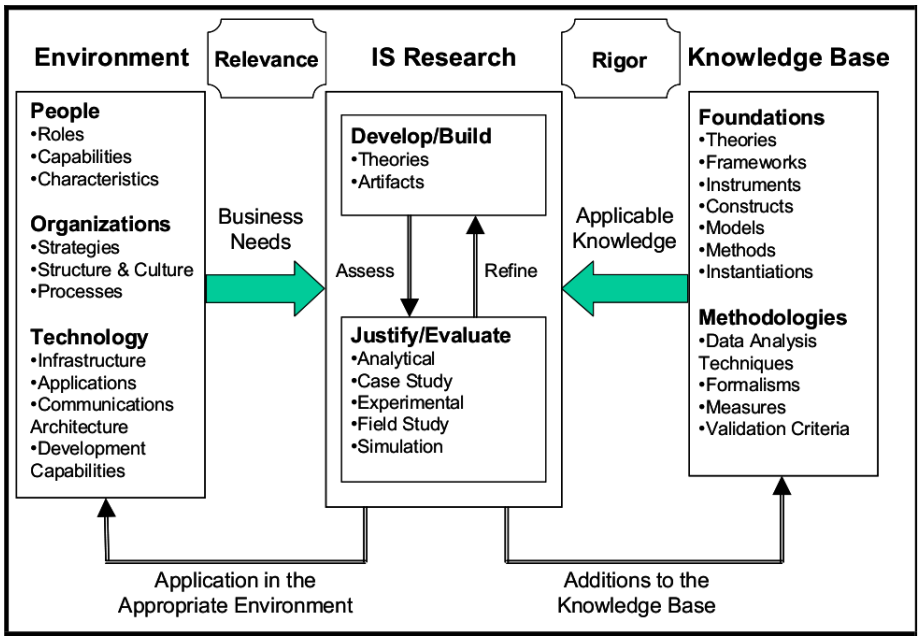

Figure 1: Design Science Research framework by Hevner et al. [2004].

Hevner et al. [2004] argue that that behavioural science and design science are not two separate approaches as in traditional IS research. They describe behavioural science as follows: "Behavioural science addresses research through the development and justification of theories that might explain or predict phenomena related to the identified business need.". They furthermore state that "Design science addresses research through the building and evaluations of artefacts designed to meet the identified business need". As seen in Figure 1, Hevner et al. [2004] combine the two phases into one framework. The environment column is inherited from traditional IS research and gives business needs as input (or motivation) to the middle 'IS research'-column. The 'knowledge base'-column combines the knowledge bases from behavioural science and design science into one. The 'knowledge base'-column gives applicable knowledge as input to the IS research. They justify the 'IS research'-column in the following manner: "The contributions of behavioural science and design science in IS research are assessed as they are applied to the business need in an appropriate environment and as they add to the content of the knowledge base for further research and practice. A justified theory that is not useful for the environment contributes as little to the IS literature as an artefact that solves a nonexistent problem".

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Figure 2: Design Science Research guidelines by Hevner et al. [2004].

As an addition to the framework Hevner et al. [2004] present seven guidelines for conduction good DS research. The original guideline table is illustrated in Figure 2. The guidelines were originally used to evaluate three DS articles, just as we intend to use our adapted guidelines to evaluate DS software development.

Following the adaption of the framework and guidelines we investigate an already planned experiment in order to establish whether the application of the adapted framework and guidelines to this experiment is feasible.

2 FRAMEWORK

In this section we introduce our process model derived from the framework presented by Hevner et al. [2004]. Furthermore, we explain how to understand the components of the original framework as components in a software development process. Later in this article, we argue for the suitability of our process model in the context of the planned experiment.

2.1 Environment

Arguing for the *People* part of the framework, Hevner et al. [2004] write "… perceptions are shaped by the roles, capabilities, and characteristics of people within the organization". In every organization different *People* have different *Roles* which are defined not just by job descriptions, but also by a de facto communicative hierarchy. In software development as well as DS research, both of these takes on *Roles* must be considered and weighed before affecting the design of the solution, no matter if it is structural (behavioural science) or artefact based (classical DS / software development). The *Capabilities* and *Characteristics* of the *People* also affects both research and software development, e.g. when developing a solution, organizational or software based, it is necessary to take in account the skill level of the employees for whom the solution is designed; if the *People* are very skilled and independent, then the solution should not contain too many crutches as these would be unnecessary and at worst a disturbance.

People	In this context people is interpreted as users or the user groups.
Purpose	Gain or refine knowledge about the roles, capabilities, and characteristics of the users, to better understand their needs.
Tasks	Interviews, user group analyses, user experiments, etc.
Output	A formalized understanding of the user roles, capabilities, and characteristics formulated as preliminary requirements.

Arguing for the *Organization* part of the framework, Hevner et al. [2004] write "Business needs are assessed and evaluated within the context of organizational strategies, structure, culture, and existing business processes". The organization mentioned, is the one where the research is being conducted. In DS research the organization benefits from the cooperation either through receiving a report describing how improvements can be made on an organizational level, through receiving an artefact to help improve e.g. production efficiency, or through both. In software development a cooperating *Organization* like this cannot always be found, as in some cases software is developed not for a single customer but for a market. Even in this case though, it is necessary to know something about one's target's *Strategies* (both business and organizational), *Structure*, *Culture*, and work *Processes* as these factors all obviously affect what makes the solution design optimal. Also, it could be noted, that there may be other people standing between you and the end users e.g. investors, non-user decision makers, etc.

Organizations	In this context organizations is interpreted as the stakeholders other than users i.e. investors, non-user decision makers, etc.
Purpose	Gain or refine knowledge about the strategies, structure, culture, and (work) processes of the stakeholders, to better understand their needs and demands as well as how best interact with them.
Tasks	Interviews, organization analyses, stakeholder analyses, etc.
Output	A formalized understanding of the strategies, structure, and (work) processes of the stakeholders formulated as preliminary requirements.

Arguing for the *Technology* part of the framework, Hevner et al. [2004] write that business needs "… are positioned relative to existing technology infrastructure, applications, communication architectures, and development capabilities". This is especially true, when designing an artefact in DS research or through software development. The technological *Infrastructure* and *Communications Architecture* is very important as it, for example, would be useless to develop a mobile solution for a company that does not have the technological infrastructure to support this. Also, no matter the solution type, one should look into which *Applications* the customer or target audience is used to using, as this gives good grounds for improvement and correction of the mistakes of others. Moreover, one should be careful not to over or underestimate one's own *Development Capabilities*.

Technology	In this context technology is interpreted as tools, APIs, competing/alternative solutions, IDEs, source code, development capabilities, etc.
Purpose	Gain or refine knowledge about the technological foundations and limitations of the development project.
Tasks	Comparative analyses, feature tests, trials of APIs, tools, and IDEs, etc.
Output	Preliminary work sheets for or descriptions of the technological aspects which can aid in the design process.

2.2 Knowledge Base

Hevner et al. [2004] describes theories, frameworks, instruments, etc. as tools from referential disciplines that provide the '*Foundations*' on which the *Build/Evaluate* process works. This is analogous to the *Knowledge Base* in our Design Science Software Development Process (DSSDP) where it acts as a support tool for the *Initial Knowledge Building*-phase and the *Develop*-phase. The *Knowledge Base* has slightly different roles in each of the two phases. In the *Initial Knowledge Building*-phase the information drawn from the knowledge base is primarily specific knowledge about the needed technology and specifications from the *Environment*-column. The additions in the *Initial Knowledge Building*-phase consist of the gathered knowledge about the relevant technologies and alternatives such as small prototypes, performance tests, and feature sheets. These things should be investigated in a manner, that they provide enough information to make properly informed design decisions in the *Develop*-phase.

Foundations	In this context 'foundations' is interpreted as the theoretical, non-technical, foundation for the development project, e.g. linear algebra for a ray-tracing project.
Purpose	Gain or refine knowledge about the theories, instruments, constructs, models, (theory) instantiations, and methods that are relevant to the project.
Tasks	Explore theory within the domain of the development project, i.e. reading articles, books, etc.
Output	Preliminary work sheets for or descriptions of the theoretical aspects which can aid in the design process.

In the context of the *Development*-phase the *Knowledge Base* functions as the source of the *Methodologies* used to test and evaluate the development process. The knowledge gathered in the *Initial Knowledge Building*-phase is extracted and utilized from the "local" knowledge base, and very few contributions are made to it from this point. Again this is in accordance with Hevner et al. [2004], since the *Methodologies* are intended to provide guidelines for the *Develop/Build*-component in their framework. Rigor then is achieved by using well established methodologies.

Methodologies	In this context 'methodologies' is interpreted as the tools used to test and evaluate the development process.
Purpose	Define, refine, or apply the method(s) with which the development project should be evaluated, in close correspondence with the output from Environment tasks, to ensure the quality of the final product.
Tasks	Identify, formalize or apply the appropriate data analysis techniques, formalisms, measures, or validation criteria for evaluating relevant parts of the solution.
Output	A formalized method of how to evaluate the product in both the justify/evaluate part of the Development phase and the Final Evaluation phase.

2.3 IS Research

To justify the *Develop/Build* part of the framework Hevner et al. [2004] write "Behavioural science addresses research through the development and justification of theories that explain or predict phenomena related to the identified business need. Design science addresses research through the building and evaluation of artefacts designed to meet the identified business need". Not only does this defend the structure of the "IS Research"-column, but it also sums up the point of the two parts therein. The *Develop/Build* phase can alter between being construction of *Theory* or of *Artefact*. In ordinary software development, one would focus entirely on the artefact, but since we try to incorporate DS in software development, the theory building can have a purpose.

Develop/Build	In this context develop/build is interpreted as the individually for theories and artefacts. Theories are interpreted as knowledge; knowledge building implies performing tasks from the Environment column or from the Foundation part of the Knowledge Base column. Artefact development implies developing the software solution based on the knowledge gained through these tasks.
Purpose	Advance in knowledge iterations or development iterations.

Tasks	Develop software or perform tasks from the Environment column or from the Foundation part of the Knowledge Base column.
Output	Entries in the internal knowledge base or a new instantiation of the developed software product.

Regarding the *Justify/Evaluate* part of the framework Hevner et al. [2004] argues that "... research assessment via the *Justify/Evaluate* activities can result in the identification of weaknesses in the theory or artefact and the need to refine and reassess". Then how does one assess the research or, in a software development context, the developed software solution? Hevner et al. [2004] suggest several approaches in their framework. These approaches are *Analytical*, *Case study*, *Experimental*, *Field Study*, and *Simulation*, and while these are mostly used in research, they could all be applied in a pure software development context as well. HCI analysis is one example of how software developers can be *Analytical*, and in a knowledge based software development company, there is plenty of knowledge to be gained from performing e.g. *Field Studies*.

Justify/Evaluate	In this context justify/evaluate is interpreted as the tools used to test and evaluate the development process.
Purpose	To either reach an understanding of what the next develop/build iteration should focus on, or to conclude that the solution is ready implementation in the user domain.
Tasks	Perform tasks from the Methodologies part of the Knowledge Base column with an emphasis on applying already existing data analysis techniques, formalisms, measures, and validation criteria.
Output	The removal or addition of entries to a backlog or a final written assessment stating that the state product meets the validation criteria.

2.4 Framework Chronology

In order to utilize the framework from Hevner et al. [2004] as a software development framework, it should be expressed as a process that is fairly easy to follow and understand for the developers, but without changing the utility of the components. The process synthesized from the framework by Hevner et al. [2004] is called Design Science Software Development Process (DSSDP) and is illustrated in figure 3.

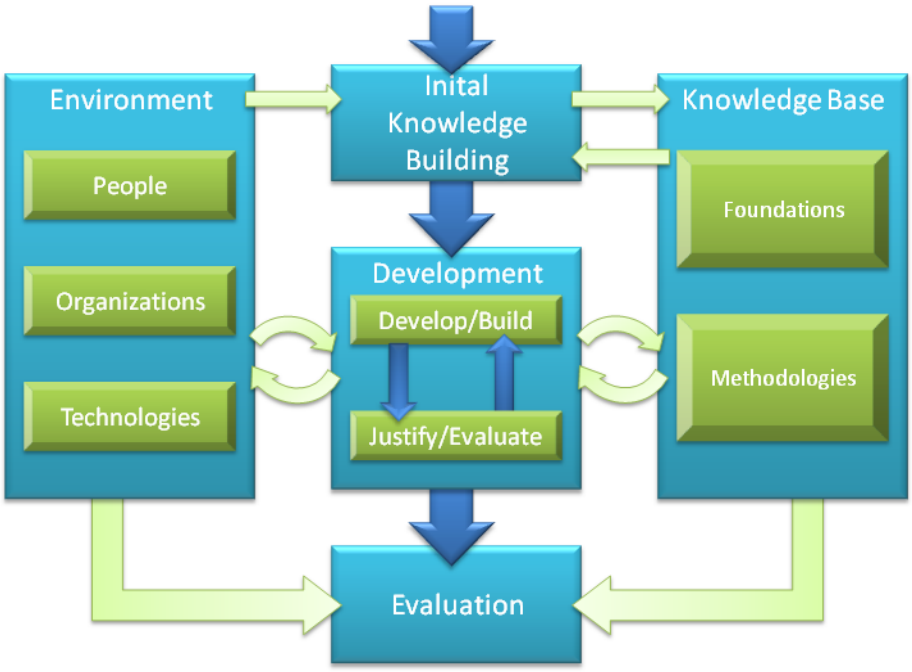

Figure 3: Design Science Software Development Process derived from the framework by Hevner et al. [2004].

The process consists of three phases: *Initial Knowledge Building*, *Development*, and *Evaluation*, where the *Initial Knowledge Building* is the first phase. The next phase is the Development phase which is derived from the "IS Research" component in the original framework, and as its role model it is iterative. The process concludes with the *Evaluation*-phase. In the *Initial Knowledge Building*-phase tasks from the *Environment* and *Knowledge Base* columns is performed, with exception that "apply-tasks" from *Environment* and *Methodologies* are not performed. In the Development phase tasks from *Develop/Build* and *Justify/Evaluate* is performed. Note however, that some of these tasks refer to other tasks in other columns. The final *Evaluation*-phase primarily consists of "apply-tasks" from *Environment* and *Methodologies*.

3 GUIDELINES

Hevner et al. [2004] write in their introduction: "The primary goal of this paper is to inform the community of IS researchers and practitioners of how to conduct, evaluate, and present design-science research. We do so by describing the boundaries of design science within the IS discipline via a conceptual framework ... and by developing a set of guidelines for conducting and evaluating good design-science research". Just as the last section in this article focused on adapting the framework by Hevner et al. [2004] to our context, we now adapt their DS guidelines in order to provide a tool with which to evaluate the process performed using the adapted framework as a whole (as opposed to evaluating tasks within the process). The adaptations made are done in accordance with what we believe is the *principles* of the original guidelines, and are only to be used in the experiment planned. The suitability of this adaptation is discussed in the Experiment section of this article.

Describing their first DS guideline Hevner et al. [2004] write: "The result of design-science research in IS is, by definition, a purposeful IT artefact created to address an important organizational problem". This is of course seen in contrast to ordinary IS research in which the effort is often focused on aspects of IS other than artefacts. One could assume that the adaptation of this guideline to a software development context is purely trivial, but some specifications regarding the state of the artefact instantiation is needed.

Guideline 1	**Design as an Artefact**
Original Description	Design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation.
Interpretation	In our experiment development must produce an artefact in the form of a complete software solution or a software prototype.

The explanation of the second guideline reads as follows: "The objective of research in information systems is to acquire knowledge and understanding that enable the development and implementation of technology-based solutions to heretofore unsolved and important business problems". The focus on *unsolved* and *important* problems stem from scientific research tradition; in order to get your work published, you are required to provide new knowledge thus not replicating other people's work. This guideline can be almost directly mapped to software development, as uniqueness is an important trait of a healthy software company.

Guideline 2	**Problem Relevance**
Original Description	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Interpretation	In order to perform economically viable software development, the developed software artefact must solve important and relevant business problems.

"The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods. Evaluation is a crucial component of the research process." Hevner et al. [2004] write to introduce Guideline 3: Design Evaluation. Evaluating one's artefact is important in research as the publication of erroneous results can lead to many unwanted implications. Similarly, an erroneous software solution can be fatal to the reputation of the software company publishing it.

Guideline 3	**Design Evaluation**
Original Description	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.
Interpretation	The utility, quality, and efficacy of the developed software artefact must be rigorously demonstrated via well-executed evaluation methods.

To introduce their fourth guideline Hevner et al. [2004] write: "The Effective design-science research must provide clear contributions in the areas of the design artefact, design construction knowledge (i.e., foundations), and/or design evaluation knowledge (i.e., methodologies)". However, in the context of software development the edge your company has over its competitors can be the difference between getting a contract and not getting a contract. Therefore, in this context the knowledge gained during a project should stay within the company.

Guideline 4	**Research Contributions**
Original Description	Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies.
Interpretation	The knowledge gained over the course of the development process must be documented in a form that allows the knowledge to be utilized in later development projects.

"Rigor addresses the way in which research is conducted", Hevner et al. [2004] begins their description of the fifth guideline. They continue: "Design-science research requires the application of rigorous methods in both the construction and evaluation of the designed artefact". Rigor in a research context can both be through rigorous data collection or through mathematical proofs; the transition to a software development context is minute.

Guideline 5	**Research Rigor**
Original Description	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
Interpretation	In the experiment software development must rely upon the application of rigorous methods in both the construction and evaluation of the software artefact.

To describe the meaning of their sixth guideline Hevner et al. [2004] write: "Design science is inherently iterative. ... Heuristic search strategies produce feasible, good designs that can be implemented in the business environment". This is in accordance with the trend in software development as the linear *waterfall* type of development is being abandoned for the less rigid iterative development models.

Guideline 6	**Design as a Search Process**
Original Description	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

Interpretation	In the experiment software development must emphasize iterative development in close correspondence between the current state of the artefact and the current requirements from the environment.

”Design-science research must be presented both to technology-oriented as well as management-oriented audiences”, Hevner et al. [2004] write to introduce their seventh and final guideline. As DS research is intended for both audiences, the results should be communicated efficiently to both. In software development, there are multiple stakeholders who all perceive the developed software solution differently.

Guideline 7	**Communication of Research**
Original Description	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.
Interpretation	The software artefact must be presented effectively to all stakeholders, i.e. investors, users, external experts, etc.

4 SOFTWARE DEVELOPMENT EXPERIMENT

Recent research has indicated that a DS influence on software development may be beneficial. This cannot be proven or disproven without a thorough experiment. The following section describes the experiment we intend to carry out, discusses how well our adaptation of the framework by Hevner et al. [2004], which resulted in a process model fits the experiment, and finally how the adapted guidelines can be utilized to evaluate whether or not the work carried out in the experiment is in tune with the principles of DS.

4.1 The Planned Experiment

The experiment is planned in the sense that timeframe, resources, and assignment is predefined, but no decisions regarding the development process, technical solutions, etc. have been made.

The software development experiment concerns the development of a mobile solution for dyslexics. It involves several stakeholders including a local innovation greenhouse as well as the grade school teacher who originally pitched the idea that mobile phones could be used to aid dyslexics to Aalborg University. The experiment is carried out over three months of development, and is intended to produce a software artefact in the shape of a *proof of concept* prototype to be used in negotiations with possible investors. The experiment is carried out by two master students at the University at Aalborg, both of which are experienced programmers and familiar with the concepts of DS. The experimenters are required to keep track of their work in an electronic diary and the knowledge gathered is to be added to an online representation of their knowledge base in the form a wiki; this is done in order to allow researchers to analyze not just the work done, but also the quality of the knowledge gathered.

Since none of the experimenters have any specific knowledge about dyslexics or dyslexia as a condition, knowledge will need to be gathered before the actual programming can begin. Apart from requirements regarding context (mobile phones and dyslexia), it is entirely up to the experimenters how they choose to approach the problem they are presented with in terms of programming technologies etc.

After the experiment is completed, the quality and credibility of its results will be evaluated by the degree of faithfulness with which the experimenters have followed the process model.

4.2 Suitability of our Process Model

Following this, our process model will be discussed. The model utility is analyzed with respects to the phases in figure 3, starting with the *Initial Knowledge Building* and ending with the final *Evaluation* phase.

4.2.1 Initial Knowledge Building

The *Initial Knowledge Building* phase is about exploring the domain before beginning the actual software design, instantiation, and implementation. Recall that "In the Initial Knowledge Building phase tasks from the Environment and Knowledge Base columns is performed, with exception that 'apply-tasks' from Environment and Methodologies are not performed". This means that the tasks at hand are:

- Interviews (people), user group analyses, user experiments, etc.
- Interviews (organization), organization analyses, stakeholder analyses, etc.
- Comparative analyses, feature tests, trials of APIs, tools, and IDEs, etc.
- Explore theory within the domain of the development project, i.e. reading articles, books, etc.
- Identify or formalize the appropriate data analysis techniques, formalisms, measures, or validation criteria for evaluating relevant parts of the solution.

The suitability of this phase depends on whether these tasks sufficient for the *Initial Knowledge Building*-phase of the experiment. Note how the tasks include interviews and theory exploration, which could be one way to gain knowledge about the user group. Moreover, the tasks above include a technological aspect, allowing researchers to get an idea of what kinds of technologies should be used in the project. Finally, the tasks include formalizing validation criteria, in order to identify what requirements the software artefact should satisfy before development can be considered a success.

4.2.2 Development

The development phase is designed to be the core artefact development phase. This is where all design and programming work is carried out. Recall that "In the Development phase tasks from Develop/Build and Justify/Evaluate is performed. Note however, that some of these tasks refer to tasks in the other columns". When we explore the tasks in this phase in depth we end up with the following list of tasks:

- Develop software or perform tasks from the Environment column or from the Foundation part of the Knowledge Base column.
- Perform tasks from the Methodologies part of the Knowledge Base column with an emphasis on applying already existing data analysis techniques, formalisms, measures, and validation criteria.
- Interviews (people), user group analyses, user experiments, etc.
- Interviews (organization), organization analyses, stakeholder analyses, etc.
- Comparative analyses, feature tests, trials of APIs, tools, and IDEs, etc.
- Explore theory within the domain of the development project, i.e. reading articles, books, etc.
- Identify, formalize or apply the appropriate data analysis techniques, formalisms, measures, or validation criteria for evaluating relevant parts of the solution.

Note how almost all of the model components and their tasks are in play during this phase, allowing the experimenters to develop both software and knowledge needed to make the software better. Moreover, the tasks gained through the Justify/Evaluate activity allows experimenters to evaluate the existing software instantiation as well as the knowledge in the knowledge base to better determine where the focus of the next develop/build activity should be.

We argue that the set of tasks in this phase is contains all of the tasks of an ordinary development phase in a software project.

4.2.3 Evaluation

The final evaluation phase was designed to meet the strict requirements regarding rigor in DS. Recall that "The final Evaluation phase primarily consists of "apply-tasks" from Environment and Methodologies".

- Interviews (people), user group analyses, user experiments, etc.
- Interviews (organization), organization analyses, stakeholder analyses, etc.
- Comparative analyses, feature tests, trials of APIs, tools, and IDEs, etc.
- Apply the appropriate data analysis techniques, formalisms, measures, or validation criteria for evaluating relevant parts of the solution.

With this set of tasks, the experimenters will be able to do traditional evaluation of software. User experiments, feature tests, etc. are common in software development and the broad array of possible tasks ensure that no matter how the experimenters chose to evaluate their artefact, their method will, in one way or the other, be included in the list above.

4.3 Evaluation Based on the Guideline Adaptation

When the experiment is done, we will need to evaluate whether or not the work done in the experiment can be considered to be in accordance with the *principles* of DS. Just as Hevner et al. [2004] used their original guidelines to analyze existing work claiming to be DS, we will use our adapted guidelines (described in the Adapted Guidelines section of this article) to establish how well the process model keeps developers on the DS path. When the experiment is done, the experimenters devotion to each guideline will be examined as the following seven questions, originating from the seven adapted guidelines, will be answered.'

1. Did development produce an artefact in the form of a complete software solution or software prototype?
2. Does the developed software artefact solve important and relevant business problems?
3. Was the utility, quality, and efficacy of a software artefact rigorously demonstrated via well-executed evaluation methods?
4. Was the knowledge gained over the course of the development process documented in a form that allows the knowledge to be utilized in later development projects?
5. Was rigorous methods in both the construction and evaluation of the software artefact applied?
6. Did the development emphasize iterative development in close correspondence between the state of the artefact and the requirements from the environment?
7. Was the software artefact must be presented effectively to all stakeholders?

Obviously, these questions cannot be answered with a mere ”yes” or ”no”, but will have to be discussed in depth in order to establish whether they are fulfilled to an acceptable degree. If this is the case for all or most of the questions, the development can be considered to be in accordance with the *principles* of DS.

5 CONCLUSION

In this article we interpreted the framework and guidelines by Hevner et al. [2004] and adapted them to a software development context. Next we investigated an already planned experiment in order to establish if the application of the framework and guideline adaptation to this experiment was feasible.

In the *Software Development Experiment* section we argued that the tasks available in each phase of the framework fit the expected tasks of these in a satisfactory degree. Therefore we now conclude that the process model that resulted from the framework adaptation fits the experiment. Furthermore, we also converted the adapted guidelines into a list of questions to be evaluated. This was done in order to provide a tool for determining whether the software development project in the experiment, is in tune with the DS principles. Since we argue that our adapted guidelines are true to the original guidelines by Hevner et al. [2004], and our list of questions is derived directly from the adapted guidelines, we conclude that if the DS software development project in our experiment is true to the DS *principles* it will be able satisfy all or most of the questions on the list.

Since we conclude that the constructed process model is true to the principles of the DS theory, i.e. Hevner et al. [2004] from which it is adapted, we can also conclude by proof of concept, that it is in fact possible to adapt a DSR method to a process model for software development.

This naturally motivates the actual execution of the experiment in order to determine if the process in this paper works in practice and if DS has any beneficial effects on commercial software development.

Such an experiment will be carried out during the first quarter of 2009 at Aalborg University.

Alan R. Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. *Design Science In Information Systems Research*, MIS Quarterly,28(1):75, 2004.

Jesper Lund Andersen and Kim Markfoged. *Design Science Theories Applied as Software Development Frameworks*. 2009.

Jesper Lund Andersen and Kim Markfoged. *Design science applied as a development framework in unfamiliar programming environment.* 2008.

Morten Andersen, Søren Rode Andreasen, Lasse Bæk, and Philip Bredahl Thomsen. *Facilitating innovations in software development*. Master's thesis, Aalborg University, 2007.

Robert Dubin. *Theory building*. Free Press, 1978.

S. Gregor and D. Jones. *The Anatomy of a Design Theory*, Journal of the Association for Information Systems , 8(19), 2007.

Markus Krogh, Jais Heslegrave, Thomas Justesen, and Daniel Korsgard. *Design science-appling the anatomy of a design theory to a mobile application*. 2008.

Morten Bøgh Sørensen, Anders Ejlersen, Michael Stampe Knudsen, and Joachim Løvgaard, *Using design science to develop a mobile application*. 2008.

Vijay Vaishnavi and Bill Kuechler. *Design research in information systems*. 2007. http://www.isworld.org/Researchdesign/drisISworld.htm.

Joseph Walls, George Widmeyer, and Omar El Sawy. *Building an information system design theory for vigilant eis*. Information Systems Research, 3(1):36–59, 1992.

APPLYING A DESIGN SCIENCE PROCESS MODEL TO COMMERCIAL SOFTWARE DEVELOPMENT

Jesper Lund Andersen, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, origo@cs.aau.dk

Kim Markfoged, Department of Computer Science, Aalborg University, Selma Lagerløfs Vej 300, 9220 Aalborg, fogeden@cs.aau.dk

Abstract

In this article we analyze a development experiment carried out using a software development process model that is an adaptation of the Design Science Research (DSR) framework and guidelines by Hevner et al. [2004]. The main focus is to determine whether or not a DS based development process can be used in a real-life, commercial software development project. The software process model used in this experiment was developed in Andersen and Markfoged [2009b] and the experiment concerns the development of a software prototype intended to aid dyslexic people in everyday reading tasks. Based on the state of the prototype at the end of the experiment, we conclude that the process model from Andersen and Markfoged [2009b] can in fact be used in a commercial software development project. Finally, based on observations made during the experiment we conclude that many software development projects could benefit from adopting the rigour of the knowledge building and documentation aspects of DS.

1 INTRODUCTION

In this article we analyze a software development experiment conducted with an adaptation of the work of Hevner et al. [2004] in order to determine if Design Science Research has contributions to software development. In order to analyse the experiment we investigate a set of diaries, which has been kept during the development of the artefact, and from these diaries we evaluate firstly, if the experiment can be considered as following the DS principles, and secondly if the diaries show any indications of the development process benefiting by following the DS principles. These diaries contain information about which tasks were being processed, problems and solutions, and experiences with technologies. This should give a clear impression of the development on a day-by-day basis. The artefact, or product, of this software development experiment is a prototype MMS content reply service for people suffering from dyslexia. It was developed during a three month period from December 2008 to February 2009.

The software development experiment is performed using a process model (See Figure 1) which is an adaptation of the DS framework from Hevner et al. [2004] and is described in detail in Andersen and Markfoged [2009b].

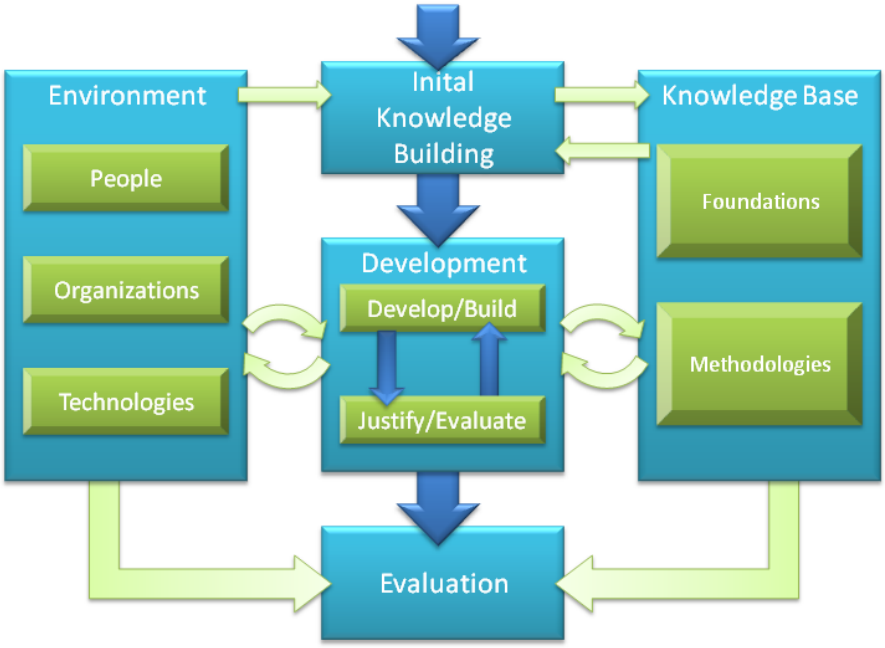

Fig. 1 Design Science Software Development Process from Andersen and Markfoged [2009b].

The process model consists of three phases; Initial Knowledge Building, Development, and Evaluation. These three phases constitute the union of what was once the IS Research column in the original framework by Hevner et al. [2004] and the concepts of initial research and final evaluation known from typical DSR. The Environment and Knowledge Base-columns have been adapted to the software development context¸ but otherwise remain intact. Each of the phases have a set of tasks; the task list for Initial Knowledge Building contains tasks such as Interviews (people), Interviews (organizations), Comparative analysis & Explore theory, and Identify or formalize evaluation criteria. Development tasks include Interviews (people), Interviews (organizations), Develop software, Evaluate software, Comparative analysis, Explore theory, and Formalize evaluation criteria, while Evaluation tasks among others consist of Interviews (people), Evaluate software, Interviews (organizations), and Evaluate software. For a more detailed description of tasks consult Andersen and Markfoged [2009b].

In addition to the adaptation of the framework from Hevner et al. [2004], the guidelines from the same article have been interpreted and have resulted in seven questions intended to evaluate whether the development can be considered true to the DS principles. Note that this evaluation in no way the principle focus of this article; that we ensure that development is true to the DS principles is merely a precondition that needs to be met for our conclusion about DS in a software development context to be possible.

2 METHODOLOGY

In order to collect and analyze the data from the software development experiment it is important to have a standardized format. This is emphasized by the fact that the researchers and practitioners in this experiment are the same people, and thus are forced to be self-analytical. The format we have chosen in to document the development in this experiment is called diaries.

In Jepsen et al. [1997] it is proposed that the purpose of diaries is to have a better correspondence between the planning phase and the evaluation phase, which is paramount for this project. They also point out some of the advantages of utilizing diaries as being improved documentation of the development process and identification of ineffective/bad working habits. This makes diaries well suited for this experiment.

Jepsen et al. [1997] propose a series of guidelines or advices for implementing diaries, based on their experiences. These guidelines are as follows:

1. Make the intention clear
2. Be disciplined and careful
3. Make a checklist of issues
4. Be selective and thorough
5. Decide on when
6. Reflect on how
7. Consider other related techniques

Note that they emphasize that these are only meant as advices, and should be adapted to the concrete project. Our implementation of these advices manifested in diary directives, which acts as a template for diary entries, is presented below.

- *Today's work* - A short informal description of today's tasks.
- *Problems processed* - Which problems and tasks has been the focus of today's work.
- *Hevner et al. DSR Basis* - Is there anything in today's work that directly relates to the theory of Hevner et al. [2004].
- *Problems solved (and how)* - Which tasks has been solved and what/who helped solve them?
- *Problems occurred* - Which new problems have occurred in today's work?
- *Knowledge Base addition* - What has been added to the Knowledge Base today?
- *Backlog snapshot* - A snapshot of how the backlog looked today, such that problematic tasks can be identified.

This template follows the guidelines in Jepsen et al. [1997]. The first guidelines are fulfilled since we have a clear intention of what the diaries should document (development experiences, difficulties, strengths, weaknesses, etc.). The second guideline is not easy to prepare, but with the introduction of directives we hope to ensure the rigorous execution of each diary entry. The third guideline is manifested in the backlog, as it acts as an advanced checklist of problems. This is also the case for the fourth guideline, where the backlog helps prioritize the tasks at hand. The directives do not dictate the time of day an entry should be written. However, it has been decided that an entry is added after each development day. The sixth guideline is covered by using more time on diary writing the first couple of days, to completely agree on the writing style and abstraction of the entries. For the seventh guideline we have supplemented the diary writing with a wiki, which contains the (internal) Knowledge Base.

The diaries for this experiment can be found in the appendix.

2.1 Analysis method

In order to ensure quality in our analysis, the following process is used during the analysis.

The experimenters' development process, documented in the diaries, is analysed step-by-step in relation to the process model explained in the introduction. Each step is analysed in the following manner:

1. How did the experimenters follow the process model, and what was the consequence?
2. How did the experimenters deviate from the process model, and what was the consequence?
3. What do these questions tell us about the utility of the process model during this step?

Finally, after having answered these questions for all three steps (Initial Research, Development, Evaluation) we collect the observations to evaluate the overall experience of using a DS based software development process. This is done in order to determine whether the overall result is "greater than the sum of its components".

3 DEVELOPMENT PROCESS ANALYSIS

In this analysis we go through the different phases of the process model which the experimenters used, in order to determine where they stuck to the model and gained from that, but also where they did not follow the model and what was gained from that. Recall from the introduction the concept of tasks meant to be performed during each phase, as the analysis is based on the performed tasks during each phase.

3.1 Initial knowledge Building

Interviews (people)

To make sure that the experimenters had a good understanding of the user group and identify where their difficulties lay, the experimenters interviewed Bjarne Jensen, who is a reading/dyslexiaa consultant. This decision was made because, while members of the user group could give the experimenters feedback about user interface etc., Bjarne Jensen supplied the experimenters with some relevant theory about dyslexia and had a very good overview of alternative solutions as noted in the diary: "The meeting was very constructive, as we intended to get an expert opinion on our solution. The knowledge from the meeting is added to the knowledge base as an addition relevant to the environment".

Interviews (organizations)

The experimenters had two organizational contacts: Anette Damgaard who is a teacher and supplied the initial idea for the artefact, was the contact to the Danish school system, and could facilitate relevant contacts and Steen Palle who is Head of Investments in the Innovation department of Novi which is a publicly funded company supportig the establishment of innovative projects and companies. Steen Palle analysed the business potential and commercial possibilities of the product idea, and deemed it worthy of future investments if a functional prototype could be provided as seen in the following quote from November 6th: "Steen also thought that there was business potential in the idea, and we agreed that there should be developed a functional prototype".

Comparative analysis & Explore theory

The experimenters did some initial comparative analyses of several technologies, which could be suitable for the artefact. In this phase the research was primarily focused on, which features the alternatives supplied and how easily they could be implemented into the design. In the case of the MMS gateway the experimenters had three alternatives: Code their own MMS Gateway, a fully featured Open Source MMS Gateway, and a fully featured proprietary MMS Gateway. After comparing their resources and the features of the alternatives, the experimenters chose the proprietary solution: "We looked into an open source MMS Gateway for Linux called Mbuni as well as a proprietary solution for MS Windows called NowSMS. First impressions are that NowSMS seems to be the easier and more reliable solution." (November 10th). Another example of comparative analyses is the optical character recognition (OCR) module; in both cases the experimenters stuck with their choice, which makes it seem like the comparative analyses was a precise and useful tool during this phase. The experimenters also studied some of the theory supplied by Bjarne Jensen.

Identify or formalize

The experimenters did not determine any formalisms or techniques for testing and evaluation during this phase. This made it possible for them, to perform the testing and evaluation they saw fit during development instead of having to follow a predetermined method.

3.2 Development

Interviews (people)

In order to verify that the user group could adopt the user interface, and to identify any weaknesses early on, the experimenters conducted an interview with two members of the user group, as seen here: "They both confirmed our belief, that it would be best to use the original MMS interface for the service - This means that, for now, we can stop worrying about GUI design. Furthermore, the children went into details about their capabilities in everyday life, and told us that if a service like the one we are attempting to create existed, they would definitely use it". The confidence gained from the user interview was a definite confidence boost for the experimenters, as well as a foundation for designing the rest of the artefact.

Interviews (organizations)

The experimenters had a discussion with Anette Damgaard, who supplied the initial artefact idea, on patents and intellectual property rights, which is apparent from the entry on December 13th: "Today we had a meeting with one of our key stakeholders. Anette, who had the initial idea for the product, was worried about protecting her idea when speaking with possible investors. After a long discussion we had managed to calm her down, and had promised that we would look into seeking patents and other ways of protecting the idea. We later contacted a Department innovation initiative called Greenhouse, and they told us that software patents were extremely expensive and hard to create and maintain. This confirmed our initial assumptions about patents". One might speculate, that had the

process model not encouraged stakeholder meetings as it does, the experimenters might have ignored Anette in this case, which might have caused strife between the experimenters and their stakeholders.

Develop software

There are several examples of the software being developed, e.g. "Today we also focused purely on development" (November 24th) and "Furthermore, the linear solution to mono-chroming images was implemented and seems to be working." (November 20th).

Evaluate software

The experimenters did performance tests of some of the modules, where it was measured which modules were bottlenecks, as mentioned on November 20th: "Optimization of this solution [linear mono-chroming] is needed though, and currently it seems like multi threading is the best solution to this challenge". As later optimization and evaluation showed that the particular bottleneck mentioned here had been removed, the running evaluation during development must be said to have helped development, as a later discovery of this bottleneck might have ruined development flow.

Comparative analysis

The experimenters did more detailed testing of the OCR modules: "Today we worked with OCR, mainly the Tesseract software. We found that it is quite accurate, but the success depends of the quality of the image pre-processing. We experimented with lighting and how to enhance the image contrast. We found a page comparing several free OCR software solutions, which concludes that Tesseact and Ocropus are the most accurate out there with 99% accuracy".

Explore theory

The experimenters did not research any new theory at this point in development. Since no problems arose from this fact, it seems that no theory was explored, because it was not needed.

Formalize evaluation criteria

There were no further suggestions on how to test the technical aspects of the artefact.

3.3 Evaluation

Except for the obvious fact, that evaluation gives experimenters a better understanding of the artefact developed, not much can be said to have come from the evaluation phase. This is especially the case, since the documentation of the experiment ends with the evaluation. However, the experimenters look past the experiment, to a development period after the prototype development experiment as the following shows.

Interviews (people)

Though the experimenters did not do any user tests during their evaluation phase limited by the experiment, they did plan to test the solution when the prototype would be ready. On January 12th they write in their diary: "Today we talked about how to carry out user tests. After discussing using the HCI lab to conduct a usability test using 4-6 dyslexic people of different ages, we agreed that it would be a better test to find two users, whom we believe is typical for our user group and let them use the solution for a week or two". More than this, they also discussed with the teachers of the children whom they interviewed the possibilities of letting some of the children use the solution during school hours etc. These actions might be a sign of good intentions, but fails to hide the fact, that no users were included in the evaluation phase of the development experiment.

Evaluate software

Since the DS process model and guidelines require rigorous verification of the quality of the software artefact the experimenters performed unit tests at several stages of development as well as during the evaluation phase. Specific for the evaluation phase is an entry from January 25th:"In order to verify that our implementation is up to par, we did a test of each module today. Every module we have made works as intended, so now we just need to do a test of the performance and precision of the entire system". This unit test ensured the experimenters that there were no errors in the modules of the software, but had errors been found there would have still been time to fix them before the test of the entire system.

Interviews (organizations)

One thing that was not entirely accomplished was stakeholder input on the prototype. This was due to time restrictions, but as the following quote shows, the experimenters did plan to have meetings with stakeholders in the form of investors, and during this planning process they involved a partner outside the development team, namely Anette, who had provided contacts in the education system as well as the original, unprocessed idea for the artefact. January 26th: "We spoke with Anette today and she proposed that we meet with possible investors in the near future. We discussed it, and agreed that it would be better to use the knowledge gathered in the evaluation phase to do additional improvements to the prototype (after the experiment has ended). In order to impress investors we need to make the application more 'sexy' and make sure that virtually no technical problem can ruin the presentation. It is a shame that we are not ready to present the prototype yet, since it would be nice to have some input from people outside the development team for our evaluation phase".

Evaluate software

Last but not least, evaluation in the shape of a system wide black box test using several different input images was performed. The experimenters only performed this task once, as the first test was a success. As the quote from the diary entry of January 28th shows, the experimenters performed a quite thorough test of the software: "Today we performed a black box test with 5 different input images, each cut down in size in order to fit the following image sizes: 1280*960, 1024*768, 800*600, and 640*480. We measured both time and precision for each image and for all sizes. The averages reveal that smaller images with less white space around the text can improve our performance from an average of 2.49 words per second to an average of 3.77 words per second. We should look into how we can get input with as little wasted space in the pictures as possible. Precision averages look good (from 94.06 to 94.49), and the mistakes that are made are deemed non-crucial, as the listener is likely to understand the sentence even if 'thc' is read aloud instead of 'the' etc.". Especially the rigour put into this test is very much in line with the guidelines the experimenters were to follow.

3.4 Guidelines

The purpose of the guideline adaptations in this context according to Andersen and Markfoged [2009b] is "...we will use our adapted guidelines to establish how well the process model keeps developers on the DS path". The guideline adaptations are only included to provide formal verification of the following of the DS principles during the experiment, as it can be expected that the experimenters, who are familiar with DS research, has developed with the principles of DS research as parts of their mindset.

1. *Did development produce an artefact in the form of a complete software solution or software prototype?* An early functional prototype, with most of the functionality of the product, was developed.

2. *Does the developed software artefact solve important and relevant business problems?* Both, Steen Palle (Head of Investments at Novi) and Bjarne Jensen (Dyslexia Consultant) thought that there was plenty of business potential in the artefact. Furthermore, a recent EU law concerning accessibility opens the possibility of public funding from the EU or the Danish government.
3. *Were the utility, quality, and efficacy of a software artefact rigorously demonstrated via well-executed evaluation methods?* There was no real user test, but there were a series of unit tests and performance tests mostly tested with black box testing.
4. *Was the knowledge gained over the course of the development process documented in a form that allows the knowledge to be utilized in later development projects?* The experimenters used a Wiki to document knowledge of new technologies, comparative analyses, etc.
5. *Were rigorous methods in both the construction and evaluation of the software artefact applied?* Not to a satisfactory extend. Among other things, this is based on the lack of user input on the final prototype.
6. *Did the development emphasize iterative development in close correspondence between the state of the artefact and the requirements from the environment?* Yes. In fact there was a clearly iterative process, where modules was constructed, evaluated, and improved again.
7. *Was the software artefact presented effectively to all stakeholders?* The artefact has not been presented at the time of writing, but a presentation has been planned.

Apart from the third and fifth, all the adapted guidelines had a positive response, which indicates that the process model does keep developers in accordance with the DS principles.

4 CONCLUSION

The main focus of this article is to determine whether DS can be applied in a commercial software development project. We conclude that this is possible based on the following facts:

- The experiment discussed in this article has spawned a viable prototype which is able to convert an image sent via MMS and convert the text contents of the image into clear text, which can easily be converted to audio using commercial text-to-speech solutions.
- Investor meetings are planned which supports the notion that DS can, in fact, be used in a commercial scenario.

Note that this argument holds true, even if the investors rejects the prototype or the finished product becomes a commercial failure, since the experiment only concerns the development of an artefact and not post-development oriented tasks e.g. marketing, economic feasibility, etc.

During our work with Andersen and Markfoged [2009a] it was noted that the Knowledge Base, and Initial Research phase, seemed to be one of the advantages of DS as development method. This has been confirmed during this experiment, where experimenters once again had no need to for theory exploration after the initial research phase. This prevents the flow of development from being disrupted, first of all, by taking time from development to explore new theory but also more critically by altering artefact design due to changes in the underlying theory.

On a side note, it was observed that since this experiment had limited resources and thus did not allow specific responsibilities, such as programmer, user interface designer, interviewer, system architect, etc. to be allocated to individuals. This could have become a problem since assigning the responsibility for a task to everyone often equals assigning it to no one. The structure of the DS process model used in the experiment as well as the adapted guidelines ensured however, that even

though the experimenters had a programming-heavy background they could not ignore e.g. stakeholder and user group related tasks. Every experimenter performed tasks related to modules for which he was responsible; he programmed the software based on interviews he had participated in himself and thus, could design the module to fit the input directly from the interviews. This saves internal communication time and reduces the risk of developers receiving contaminated user input. This indicates that there may be advantages to be found in assigning more responsibility to developers.

Future work could include an experiment in an actual software development company in which the individual developer is given even more responsibility for tasks from all columns of the process model (See Figure 1), to see which advantages and disadvantages occurs when a project is managed in a less top-down manner.

Another possible future experiment could involve comparing the development in this DS experiment with its development emphasis to that done in an ordinary DS research project, to determine the differences between two processes that stem from the same principles.

Finally, it would very interesting to try to introduce the concept of a knowledge base, similar to that of DS, in other established iterative development methods such as SCRUM. This could help determine whether the knowledge base is only useful in the context of DS or whether it is a tool that can be benefited from widely.

5 REFERENCES

Alan R. Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. *Design Science In Information Systems Research*, MIS Quarterly,28(1):75, 2004.

Jesper Lund Andersen and Kim Markfoged. *Design science applied as a development framework in unfamiliar programming environment*. 2008.

Jesper Lund Andersen and Kim Markfoged. Design science theories applied as software development frameworks. 2009a.

Jesper Lund Andersen and Kim Markfoged. Adapting a design science theory to software development. 2009b.

Morten Andersen, Søren Rode Andreasen, Lasse Bæk, and Philip Bredahl Thomsen. *Facilitating innovations in software development*. Master's thesis, Aalborg University, 2007.

Robert Dubin. *Theory building*. Free Press, 1978.

S. Gregor and D. Jones. *The Anatomy of a Design Theory*, Journal of the Association for Information Systems , 8(19), 2007.

Markus Krogh, Jais Heslegrave, Thomas Justesen, and Daniel Korsgard. *Design science- applying the anatomy of a design theory to a mobile application*. 2008.

Morten Bøgh Sørensen, Anders Ejlersen, Michael Stampe Knudsen, and Joachim Løvgaard, *Using design science to develop a mobile application*. 2008.

Vijay Vaishnavi and Bill Kuechler. *Design research in information systems*. 2007. http://www.isworld.org/Researchdesign/drisISworld.htm.

Joseph Walls, George Widmeyer, and Omar El Sawy. *Building an information system design theory for vigilant eis*. Information Systems Research, 3(1):36–59, 1992.

Leif Obel Jepsen, Lars Mathiassen, and Peter Axel Nielsen. Using diaries. Reflective Systems Development, 1, 1997.

Appendix

Diaries

November 6th

Today we had a meeting with Steen Palle and Anette Damgaard. The project is initially Anette's idea, and she will contribute with the various contacts to the target user group. Steen is the facilitator between Anette and ourselves. Steen also thought that there was business potential in the idea, and we agreed that there should be developed a functional prototype. Before going to the meeting we had thought about a solution where the user only interacts with the system by MMS thus simplifying the interface as much as possible, both Steen and Anette agreed with this approach.

Problems processed

Issues regarding licenses for external software were discussed with Steen, he proposed that we look into a service called adgangforalle.dk, which has a good speech synthesis. Meetings with members of the target group were discussed with Anette. We presented our solution to Anette and Steen, and discussed alternatives in depth. We shortly discussed issues regarding who has the rights to a possible commercial solution.

Problems solved (and how)

Anette and Steen agreed that our solution, involving MMS, was the way to go.

Problems occured

No new problems occured.

Knowledge Base additions

No new additions.

Backlog snapshot

No backlog exists.

Edit

November 10th

Problems processed

Today we worked on solving the problem of setting up an MMS Gateway. We looked into an open source MMS Gateway for Linux called Mbuni as well as a proprietary solution for MS Windows

called NowSMS. First impressions are that NowSMS seems to be the easier and more reliable solution. Furthermore, we looked into MMS Gateway rental prices, which quickly turned out to be too expensive for our project in its current state. Moreover, we contacted 'IT- og Telestyrelsen', who are behind Adgangforalle.dk, via e-mail regarding using their danish speech synthesis in our solution. Other than Adgangforalle.dk we have looked into a company called Acapella Group, that specialises in speech synthesis covering many languages, including Danish.

Problems solved (and how)

No problems were solved today though, based on our first impressions of NowSMS, we have become less concerned about encountering unsolvable problems with the MMS Gateway.

Problems occured

It has become clear to us, that we do not have the resources to develop our own MMS Gateway. Futhermore, though it is not a problem that worries us too much, rental prices for MMS Gateways are alarmingly high. In the future it will probably be a good idea to keep this in mind, when considering the economics of the project.

Knowledge Base additions

MMS Gateway: We added some information about GSM Modems, which are a requirement if we set up our own MMS Gateway. Furthermore, we added links to information pages on the two MMS Gateway solutions NowSMS (proprietary) and Mbuni (open source).

Speech Synthesis: We added links to the information pages of the mentioned Adgangforalle.dk and Acapella Group.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Arrange meetings	6.11.08		Arrange meetings with a dislexic consultant and dislexic children. Waiting for Anette to supply contact information.
Speech synth.	10.11.08		Explore more danish speech synthesis solutions. 'IT- og Telestyrelsen' who are behind Adgangforalle.dk have been contacted.
MMS Gateway	10.11.08		Explore if it is possible to write our own MMS gateway, or we should investigate other alternatives.
OCR	x.11.08		Explore various Optical Character Recognition solutions. Possibly open source.
Preprocessing Module	x.11.08		Start development of preprocessing module
MP3 Converter			Investigate MP3 Conversion solution.

Edit

November 13th

Problems processed

Today we worked with OCR, mainly the tesseract software. We found that it is quite accurate, but the succes depends of the quality of the image preprocessing. We experimented with lighting and how to enhance the image contrast. We found a page comparing several free OCR software solutions, which concludes that Tesseact and Ocropus are the most accurate out there with a 99% accuracy. Furthermore, we explored the alternatives for image manipulation libraries for C++. CImg seems to meet our needs in the context of image manipulation. It contains, among numerous other things, lots of functions to rotate images (and parts thereof), draw everything from lines to fractals, and manipulate single pixels.

Problems solved (and how)

No problems were solved.

Problems occured

We realized that we need to able to firmly control lighting balance in input images for the Tesseact OCR.

Knowledge Base additions

Tesseract: We are now able to use the Tesseact software and have the source for possible implementation in the future.

CImg: We have looked into the CImg C++ image manipulation library, and it seems to fit our needs. A link to CImg's online documentation has been added to the knowledge base.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested.
OCR	13.11.08		Tesseract OCR is investigated. Accurate when supplied with proper images. Further investigation in lightning balancing.
Speech synthesis	10.11.08	13.11.08	Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dislexic children. Waiting for Anette to supply contact information.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dislexic consultan, on 14.11.08.

MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.

Edit

November 14th

Today we worked with a piece of OCR software called OCRopus. Testing of CImg continued in the form of application to our monochromification task.

Problems processed

We tried to install OCRopus on Linux (Ubuntu 8.10). The linear approach to making an image monochrome without ruining the text within was attempted with CImg.

Problems solved (and how)

During installation it turned out that OCRopus used Tesseract for character recognition, thus further testing would be pointless.

Problems occured

We ran into problems with reading RGB values of pixels in images represented as CImg images. This problem was worked on for several hours without luck. Unless this is solved soon, we might need to look into other solutions to image manipulation.

Knowledge Base additions

Knowledge about OCRopus was added to the knowledge base.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested.
OCR	13.11.08		Tesseract OCR is investigated. Accurate when supplied with proper images. Further investigation in lightning balancing.
Speech synthesis	10.11.08	13.11.08	Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Waiting for Anette to supply contact information.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexia consultant, on 14.11.08.

MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.

Edit

November 17th

Today we continued work on CImg to better the quality of our input images in the preprocessing module. We experienced progress as we are now able to read an manipulate pixels in an image.

We also looked into Spell checkers, as we had the idea that a spell checker might correct some of the errors the OCR module made. We looked at Aspell, but we don't have a working prototype yet.

Problems processed

- Trying to read and manipulate pixel values.
- Working with the Aspell library.

Problems solved (and how)

We are now able to read manipulate pixel values, which is an important part of the preprocessing module.

Problems occured

It currently takes about 30 seconds to generate 12 alternative 2 megapixel monochrome images. We need to optimize this, possibly through writing the code that is executed in seperate threads.

Knowledge Base additions

Knowledge about pixel manipulation, in the form of our working test code, was added to the knowledge base here.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as Aspell. In order to improve running time, we should look into threading for the module.
OCR	13.11.08		Tesseract OCR is investigated. Accurate when supplied with proper images. Further investigation in lightning balancing. Fatter is better.
Speech synthesis	10.11.08	13.11.08	Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech

			synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.

Edit

November 20th

Today we continued work on the ASpell spell checker, it's API is, however, rather poorly documented, and that has caused us to explore other alternatives. We looked at a .Net Spell Checker called NetSpell, and it seems quite easy to implement. So we will not continue working with Aspell. Furthermore, the linear solution to monochroming images was implemented and seems to be working. Optimization of this solution is needed though, and currently it seems like multi threading is the best solution to this challenge.

Problems processed

- Implemented spell checkers.
- Implemented the linear monochroming solution.

Problems solved (and how)

Tesseract is choosen as the OCR engine for the solution. By converting images to grayscale before proccessing them, we were able to increase the monochroming modules accuracy.

Problems occured

Monochroming is too slow - optimization is needed.

Knowledge Base additions

Knowledge about Aspell and NetSpell was added to the knowledge base.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.

Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. Aspell is being tested.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

November 21th

Today we had out meeting with dyslexia consultant Bjarne Jensen. The meeting was very constructive, as we intended to get an expert opinion on our solution. The knowledge from the meeting is added to the knowledge base as an addition relevant to the ***environment***. The additions has up until now primarily been ***technology*** based. We are now determined that our solution should not be an alternative to the current solution, but more as a supplement. His initial worries was the final cost of the solution, and the response time on the system. In this context time is now a factor we take serious.

Problems processed

Gathering of expert knowledge about dyslexia.

Problems solved (and how)

We are now quite convinced what our users can handle interface-wise, and that our solution only should deal with reading text aloud.

Problems occured

Time is now an important factor. It is neccesary to have a response time under 30 seconds.

Knowledge Base additions

Knowledge about dyslexia is added to the knowledge base.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. Netspell is being tested.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

November 24th

Today we focused purely on development. We continued development of the spell checking module, which is now nearly complete. It only needs to be tested and adapted to the interface of the rest of the solution. We are now able to spell check a text and correct potential spelling mistakes, with the help of a dictionary, which is in a commonly used format, i.e. we support a whole array of languages.

We also continued work on the image pre-processing module, where we exprimented with threads in order du make the runtime faster, since it has become a consern of ours. There has been some concurrency issues, but we expect to be able to solve these.

Problems processed

Spell checking and threading in image preprocessing.

Problems solved (and how)

Problems occured

Concurrency issue regarding writing to multiple image files at the same time.

Knowledge Base additions

Links regarding Win32 threads has been added to the knowledge base.

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

November 27th

Today we also focused purely on development. The spell checking module is now being very close to finished. We also continued work on the image pre-processing module, where we exprimented with threads in order du make the runtime faster, since it has become a consern of ours. There has been some concurrency issues, but we expect to be able to solve these.

Problems processed

Spell checking and threading in image preprocessing.

Problems solved (and how)

Problems occured

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

December 13th

Today we had a meeting with one of our key stakeholders. Anette, who had the initial idea for the product, was worried about protecting her idea when speaking with possible investors. After a long discussion we had managed to calm her down, and had promised that we would look into seeking patents and other ways of protecting the idea. We later contacted a Department innovation initiative called Greenhouse, and they told us that software patents were extremely expensive and hard to create and maintain. This confirmed our initial assumptions about patents.

Problems processed

Stakeholder relationships were nourished.

Problems solved (and how)

We managed to ensure Anette that she need not worry, which gives us time to focus on finishing our program prototype.

Problems occured

Through the proccess of calming Anette down, we had to promise that we would look into patents etc.

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
Arrange meetings	6.11.08		Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Patents			Look into patent possibilities.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

January 12th

Today we talked about how to carry out user tests. After discussing using the HCI lab to conduct a usability test using 4-6 dyslexic people of different ages, we agreed that it would be a better test to find two users, whom we believe is typical for our user group and let them use the solution for a week or two.

Problems processed

Initial considerations for user tests was performed. Further planning needed.

Problems solved (and how)

We do not need to concentrate on HCI Lab experiments.

Problems occured

We need a proper testing plan and representative members of the user group

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Patent			Look into patent possibilities.
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

January 16th

After a lot of delay due to christmas and exams, today we finally had a meeting with two dyslexic children. The meeting was held at a school in Suldrup, where a lot is done for children with dyslexia. The purpose of the meeting was to establish how good the children were with mobile phones, MMS, and built-in cameras. Furthermore we hoped to get more insight into the lives of dyslexics and into the problems they face every day.

Problems processed

User group knowledge gathering (mobile phones, MMS, and built-in cameras). User group knowledge gathering (daily problems, coping, and wishes).

Problems solved (and how)

Through the interview it quickly became clear, that both children (attending the 6th and 8th grade, respectively) were very profficient with mobile phones and their technologies. They both confirmed

our beleif, that it would be best to use the original MMS interface for the service - This means that, for now, we can stop worrying about GUI design. Furthermore, the children went into details about their *capabilities* in every day life, and told us that if a service like the one we are attempting to create existed, they would definately use it. This decreases our fear, that a mobile supplement for the current solutions would not be able to get a grip on the market.

Problems occured

None.

Knowledge Base additions

Summary of interview with dyslexic childen

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Explore danish speech synthesis solutions. Adgangforalle.dk is proposed. UPDATE: Adgangforalle.dk will not let us use their speech synthesis. New solution must be explored.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Patent			Look into patent possibilities.
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

January 21st

Today we worked on the MMS gateway and the pre-proccessing module as well as contact Acapela Group about their danish speech synthesis.

Problems processed

- Action taken to get danish speech synthesis in place.
- Pre-proccessing module was modified.
- MMS Gateway was set up. So far SMS's can be received.

Problems solved (and how)

Problems occured

None.

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Patent			Look into patent possibilities.
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

January 22nd

Today we designed a new algorithm for the spell checker. What the algorithm does is optimize the output gotten from the OCR run on the different monochrome attempts.

Problems processed

- Optimization gain from having more than one monochrome attempt.

Problems solved (and how)

- The algorithm is ready to be implemented. The Algorithm was constructed with pen on paper within the development team.

Problems occured

None.

Knowledge Base additions

Pseudo code for spell checking optimization

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented. Optimization Algorithm.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
MP3 Converter			Investigate MP3 Conversion solution.
Patent			Look into patent possibilities.
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

Edit

January 23nd

Today we began implementing the algorithm from yesterday. We expect to be done with the implementation in the near future. Furthermore, we looked into WAV to MP3 conversion and after some consideration, we decided to use the open source converter LAME.

Problems processed

- Implementation af yesterdays algorithm.
- WAV to MP3 conversion.

Problems solved (and how)

- MP3 conversion is now taken care of.

Problems occured

None.

Knowledge Base additions

LAME

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented. Optimization Algorithm.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
Patent			Look into patent possibilities.
MP3~~ Converter~~	23.01.09	23.01.09	Use LAME
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.

January 25th

In order to verify that our implementation is up to par, we did a test of each module today. Every module we have made works as intended, so now we just need to do a test of the performance and precision of the entire system

Problems processed

Module black box testing has been performed.

Problems solved (and how)

None.

Problems occured

None.

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented. Optimization Algorithm.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
Patent			Look into patent possibilities.
MP3	23.01.09	23.01.09	Use LAME
Arrange meetings	6.11.08	16.01.09	Arrange meetings with dyslexic children. Wrote an email to the principal on the school Anette works. Waiting for a reply.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.
Testing	25.01.09		Black box done. Performance needed.

January 26th

We spoke with Anette today and she proposed that we meet with possible investors in the near future. We discussed it, and agreed that it would be better to use the knowledge gathered in the evaluation phase to do additional improvements to the prototype (after the experiment has ended). In order to impress investors we need to make the application more ’sexy’ and make sure that virtually no technical problem can ruin the

presentation. It is a shame that we are not ready to present the prototype yet, since it would be nice to have some input from people outside the development team for our evaluation phase.

Problems processed

Started the investor discussion.

Problems solved (and how)

Agreed with Anette to do further improvements to the prototype before investor meetings.

Problems occured

None.

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented. Optimization Algorithm.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
Patent			Look into patent possibilities.
MP3	23.01.09	23.01.09	Use LAME
Arrange meetings	26.01.09		Arrange meetings with investors.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.
Testing	25.01.09		Black box done. Performance needed.

January 28th

Today we performed a black box test with 5 different input images, each cut down in size in order to fit the following image sizes: 1280*960, 1024*768, 800*600, and 640*480. We measured both time and precision

for each image and for all sizes. The averages reveal that smaller images with less white space around the text can improve our performance from an average of 2.49 words per second to an average of 3.77 words per second. We should look into how we can get input with as little wasted space in the pictures as possible. Precision averages look good (from 94.06 to 94.49), and the mistakes that are made are deemed non-crucial, as the listener is likely to understand the sentence even if 'thc' is read aloud instead of 'the' etc.

Problems processed

Performance tests performed.

Problems solved (and how)

None.

Problems occured

None.

Knowledge Base additions

Backlog snapshot

Topic	Begin Date	End Date	Comments
Preprocessing Module	13.11.08		Start development of preprocessing module. CImg c++ librabry is being tested as well as NetSpell. In order to improve running time, we should look into threading for the module.
Spell Checking	17.11.08		Spell checking to correct misstakes made by the OCR module. NetSpell is being implemented. Optimization Algorithm.
Speech synthesis	10.11.08		Acapela group was contacted about their danish speech synthesis solution. We now await their answer to our request.
MMS Gateway	10.11.08		We are unable to write our own MMS Gateway, a properitary solution seems like the only way.
Patent			Look into patent possibilities.
MP3	23.01.09	23.01.09	Use LAME
Arrange meetings	26.01.09		Arrange meetings with investors.
Arrange meetings	6.11.08	13.11.08	Arranged meeting with Bjarne Jensen - dyslexic consultan, on 21.11.08.
OCR	13.11.08	20.11.08	Tesseract OCR is choosen as our OCR engine.
~~**Testing**~~	~~25.01.09~~	~~28.01.09~~	~~Black box done. Performance needed~~.