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.

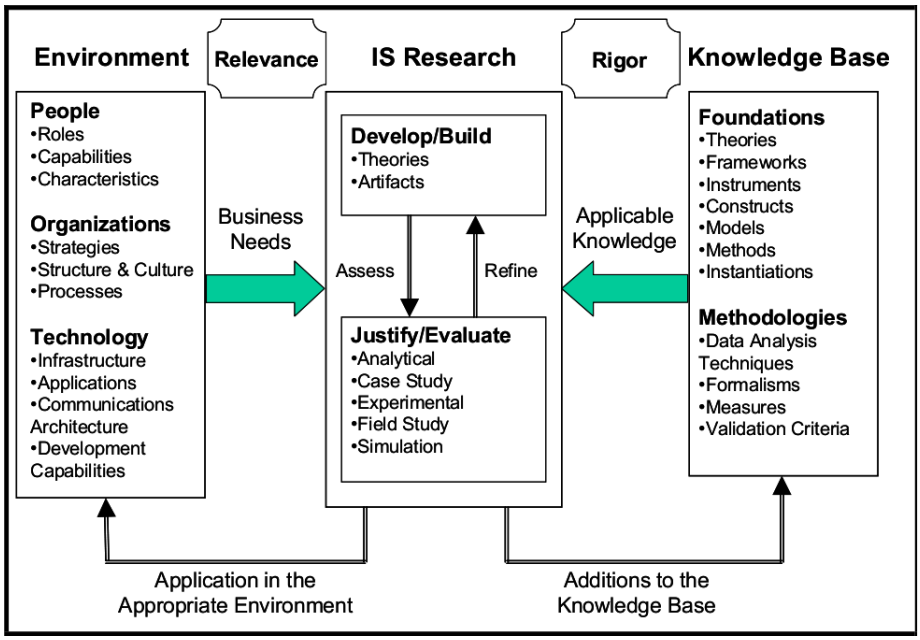

Figure 1: Design Science Research framework by Hevner et al. [2004].

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.

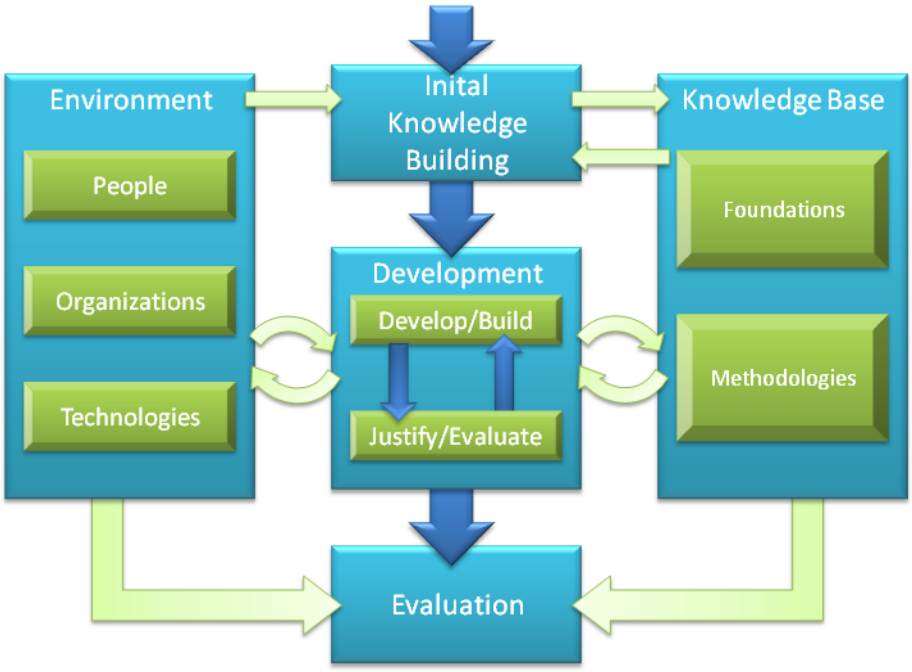

Figure 3: Design Science Software Development Process derived from the framework by Hevner et al. [2004].

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.