Association for Information Systems
AIS Electronic Library (AISeL)

MCIS 2010 Proceedings

Mediterranean Conference on Information Systems (MCIS)

9-2010

CAN DESIGN SCIENCE BE USED FOR DESIGN?

Jeremy Rose
Aalborg University, Denmark, jeremy@cs.aau.dk

Kim Markfoged
Aalborg University, Denmark, fogeden@cs.aau.dk

Jesper Lund Andersen
Aalborg University, Denmark, origo@cs.aau.dk

CAN DESIGN SCIENCE BE USED FOR DESIGN?

Jeremy Rose, jeremy@cs.aau.dk
Kim Markfoged, fogeden@cs.aau.dk
Jesper Lund Andersen, origo@cs.aau.dk
Dept. of Computer Science, Aalborg University, Denmark

Abstract

Design Science is currently much discussed in Information Systems research. In our analysis we distinguish two distinct threads in this discussion. The first thread is a meta-debate about the nature of IS research, and (in particular) how design work can properly be enacted as research. The second thread is a normative discussion of how design work should be conducted – important in a field where the principle objects of study are designed artefacts called information systems. If these principles for design have any consequence at all, they should be verifiable and they should be evaluated. In this research, conducted with student software designers at Aalborg University, we pose the question 'can design science be used to build software?' Ten experienced students in the Department of Computer Science participated in experiments to use design science theory as the principle inspiration for small software development projects. They studied the literature, chose appropriate starting theories, designed their own development processes, used them to build small mobile applications, documented their experiences and evaluated those experiences by writing research articles. A more experienced researcher helped organise the experiments and supervised the research. Students were surprisingly positive about their experiences with design science and the causes for this enthusiasm, and the underlying contribution of design science to the software developments are discussed.

Keywords: design science, system development, software development

1 INTRODUCTION

Design science is attracting increasing attention in the Information Systems (IS) research community. In many ways it represents a reaction against the Management Information Systems tradition - rigorous study of organisational phenomena and the derivation of explanations of cause and effect. A more inclusive vision of IS research is proposed, which also focuses on the design and building of different kinds of IS artefacts – but principally of computer systems. These concerns have, of course, always been important in the engineering disciplines. An important component of design science is therefore *design as science* - how design can be scientific research, as it is understood in the IS field. At the same time, another strand of the emerging IS design science literature is directly concerned with how to design; a normative literature often expressed as principles or process models. In this literature, researchers concern themselves with the *science of design* – the generalisable set of principles and processes that lie behind design activities. This *science of design*, at first glance, appears weak in two aspects. The first is that it seems unelaborated, or naive in relation to the engineering sciences' prescriptive literatures on how to conduct the design activity. These prescriptive literatures contain extended design methods (such as Rational Unified Process, Microsoft Solutions Framework) design processes (such as Model-Driven Architecture) and detailed frameworks such as information architectures (Zachman 1999). The second is that it is expressed as artefacts – frameworks, process models and sets of principles, which do not live up their own requirements for proper design. In particular, the frequently expressed requirement for proper evaluation is seldom fulfilled. Thus there are many well known expressions of design prescriptions (Hevner, March et al. 2004; Gregor and Jones 2007; Vaishnavi and Kuechler Jr 2008), and many design experiments using design theory as 'a cloak of theoretical legitimacy' (Walls, Widemeyer et al. 2004, p 55). However there are few attempts to understand whether the design principles expressed actually directly work in the development of computer systems.

In this research, an experiment (or more properly three experiments) is conducted in which the second, prescriptive form of design science is held up to inspection and evaluated. The experiments are formulated in response to the question 'can design science be used to design and build software?' The experimenters are students at the Department of Computer Science at Aalborg University who are experienced at conducting software projects, but new to design science. As engineering students they are used to prescriptive accounts of how to develop software, but unused to, and not particularly well-disposed towards abstract theoretical discussions about the nature of research disciplines. Their goal is simply to build useful working software in a way which is productive and stimulating for them. The students conduct three short development projects in which they build small programmes for mobile systems against the backdrop of design science. The development projects are discovery experiments – early experiments which are not very realistic but serve as an entry point to answering the research question. They report on their experiences in a disciplined reflective form: the research article (Andersen and Markfoged 2008; Ejlersen, Knudsen et al. 2008; Haeslegrave, Justesen et al. 2008).

The following sections of the article contain a description of the experimental design, an analysis of the design science theory used in the experiment and accounts of the three development projects (cases). A concluding discussion leads to a preliminary answer to the question posed in the title.

2 EXPERIMENT DESIGN

2.1 Discovery experiments

The experiment used a method called discovery experiment (Alberts and Hayes 2002; Alberts and Hayes 2005), which has an emerging tradition of use in innovative and agile software development research (Aaen 2008). The method is suitable for early stages of the elaboration of a research problem, where little is known about conditions, constraints or variables, and where it is impossible to

isolate particular variables for manipulation. Like natural and quasi- experiments, they rely on observation of multiple variables. Discovery experiments are designed to encourage creative and innovative problem solving, since there are very few constraints regarding execution. The discovery experiment method requires the formulation of the experiment scenario; the number of participants, the timeframe, the purpose of the experiment, and documentation.

2.2 The experiment process

The students were asked to review nine leading design science articles (Carroll 1993; March and Smith 1995; Cross 2001; Purao 2002; Hevner, March et al. 2004; Walls, Widemeyer et al. 2004; Marxt and Hacklin 2005; Arnott 2006; Gregor and Jones 2007) and the design science pages at the AIS website[i]. The articles were chosen (from the rather limited selection at the Web of Science) as a mixture of well-cited classics, and articles with a focus on system-building topics of interest to the students. The review took the form of a seminar with presentations and discussion, and a blog[ii]. Three development teams were formed and the teams were asked to choose a theoretical framework as their starting point. Simultaneously they were asked to formulate their own development project, where the only constraint was the scope of the development exercise – not more than three weeks of programming. All three groups chose to develop mobile applications (which they work with at this point in their education). Mobile application development has particular characteristics: the development platforms are relatively inconsistent and unstable, and the techniques and methods are emerging, rather than well-known and secure. The groups were given help with developing their chosen framework to a point where they felt comfortable using it, and there followed a concentrated development phase, with a requirement for documenting the experiment through log books, diaries or blogs. There was a mid-way seminar where the students presented their progress with their development work. The design process was iterative and incremental, including progressive evaluation in weekly supervisions. Each group then had help with preparing a research article documenting and reflecting upon their experience.

2.3 The experimenters

The students were in the sixth semester of their education, with courses in programming and development methods behind then. Aalborg University runs project-based education where the students devote half of their time to semester-long projects in groups, so the students have unusually wide development and programming experience. Many of them also have experience outside the university. Projects in the Department of Computing Science focus on technical development skills, with programming as the core discipline. The student culture at the university is independent, self-directed and shows aspects of what is sometimes known as hacker culture. Agile and iterative methods (XP, SCRUM) are well-known and popular; traditional methods (Object Oriented Analysis and Design (Mathiassen, Munk-Madsen et al. 2000)) are taught and used but less popular. C and C# are the preferred languages for programming education.

2.4 The three cases (experiments)

The first of the cases described in this article concern the development of a SMS-based service intended to notify users of changes to a shared calendar – in this case downloading data from the department's automated database and forwarding alerts over changes in the student's study programme. The second case is the development of a SMS service intended to facilitate easier exchange of phone numbers between users. Here the program transfers, upon request and approval, data from one phone's address book to another's, thus relieving a common difficulty of having to paste data from an SMS into an address book on the same phone. These projects had development teams of 4 students. Two students worked with case 3 - a mobile application which is able to send and receive images with SMS as the data carrier. In this programme a relatively large set of data (an image) normally transmitted as an expensive MMS, is broken up into shorter data packages (SMS – inexpensive or free) and re-assembled on arrival at the receiving phone.

3 DESIGN SCIENCE

This section of article presents the theories in design science that were used in the experiments.

3.1 Design as science

Hevner et al. (2004) the argue that much of IS research can be classified either as behavioural science or as design. *"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."* They develop a new conceptualisation of IS research, where both categories of research contribute to the discipline (Figure 1).

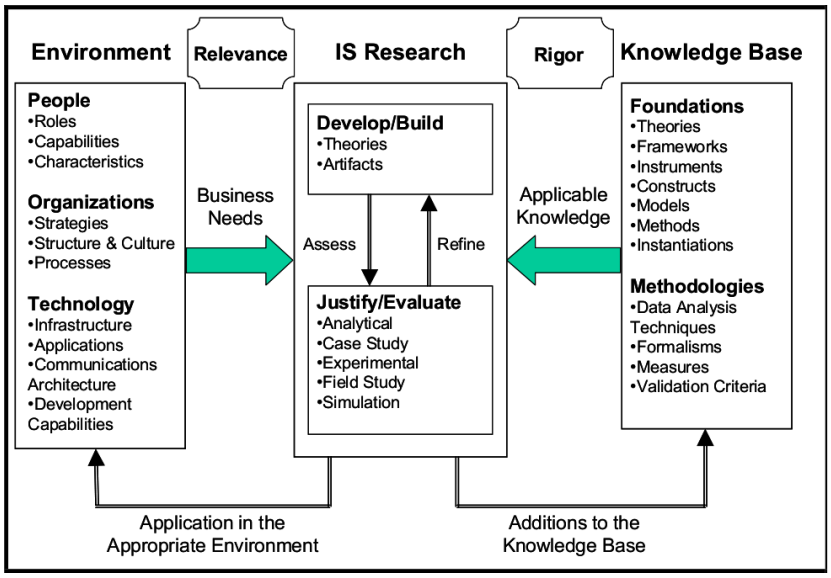

Figure 1. IS research framework, Hevner et al. (2004)

The *Environment* column in Figure 1 is inherited from traditional IS research, where business needs provides the motivation for research based on people, organizations, and technology. The *Knowledge Base* column is a combination of the knowledge base concepts from both behavioural science and design science. The *IS Research* column ties the framework together and Hevner et al. 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.*"

3.2 The Science of Design

A simple model for design research is proposed by Vaishnavi and Kuechler (2008) involving five process steps between which iteration is encouraged.

1. Awareness of Problem - the identification of a problem that needs a (better) solution. The awareness of the problem results in an initial solution proposal.
2. Suggestion - a suggestion to how the initial solution proposal should be realized. The suggestion results in a tentative design.
3. Development - the realization of the tentative design, resulting in an artefact.
4. Evaluation - an evaluation of whether or not the initial problem has been solved by the developed artefact. The evaluation of the artefact results in some sort of performance measures.

5. 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.

Gregor and Jones (2007) do not focus in process, but instead offer eight components of design theory:

1. Purpose and Scope - w*hat the system is for?* the set of meta-requirements or goals that specifies the type of artefact to which the theory applies 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."

The components assume a normative aspect in as much as it is proposed that they form the cornerstones of good design research. Hevner et al. (2004) provide seven overlapping guidelines which are designed to shape good design science research, and, by implication, good design work.

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.

4 DEVELOPING MOBILE APPLICATIONS WITH DESIGN SCIENCE

In this section we describe how the student experimenters adapted the theoretical ideas of these design theories to three live design experiments.

4.1 Case 1 – calendar alert

Here the experimenters develop a SMS based service intended to notify users of any changes to a shared calendar. It is documented in Haeslegrave, Justesen et al. (2008).

4.1.1 Theory foundation

The theoretical base for this development experiment is Gregor and Jones (2007). The experimenters adapt their understandings of Gregor and Jones' eight components, combining them with some of their own process and output thinking, to form the framework which structures their development work.

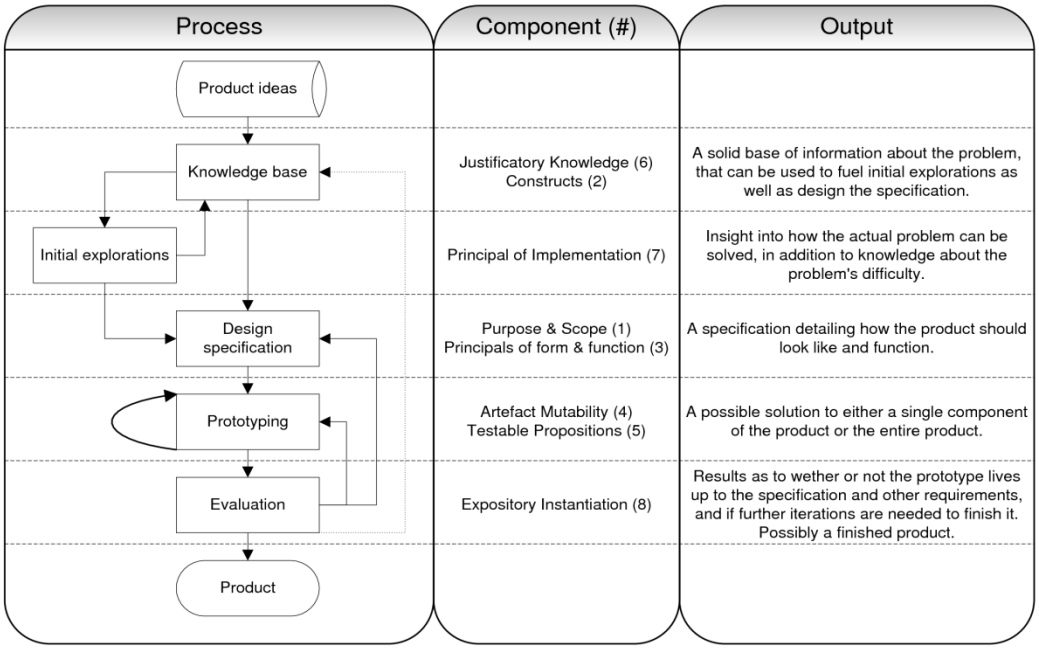

Figure 2. Case 1 development framework (Haeslegrave, Justesen et al. 2008)

4.1.2 Development process

The developers carried out early explorations and programming mock-ups to establish the programming knowledge base they could use to make their product idea work, and constructed a simple design specification, dividing the program into components which could be programmed iteratively and later integrated. They developed tests (testable propositions) to ensure that the software functioned properly, and carried out the actual development work as a series of iterations. Two pairs of programmers developed a module each. Each pair developed separate modules which were later integrated into a working prototype which then went through a refinement phase. They describe their process as follows:

Initial iterations (SMS send/receive).	Initial iterations (calendar and logger)

1. Gather knowledge about how to program a send/receive SMS functionality, and about how to read the content of the messages. 2. Gather information about deleting received SMS messages using the API. 3. Program an interface for integration with calendar and logger.	1. Figure out how to write to log and screen, and how to fetch XML data from an online calendar. 2. Program a function to load an application configuration. 3. Program an interface for integration with SMS send/receive module.
Working prototype	
Later iterations (whole product) 1. Finish error logging function. 2. Improve GUI layout. 3. Program support for plug-in functionality. 4. Reprogram calendar module to plug-in.	

Table 1. development process, experiment 1

Here the development stopped, as the program passed its tests and was considered to be of sufficient quality. However the Gregor and Jones components were used as part of the evaluation of the project effort, the framework used, and the resulting product.

4.2 Case 2 – phone number exchange

The second case is documented in Ejlersen et al. (2008). The experiment concerns the development of a SMS based service intended to facilitate easier exchange of phone numbers between users.

4.2.1 Theory foundation

These experimenters chose to base their development effort on a framework they designed themselves, which combines Hevner's (2004) well known framework with the process steps of Vaishnavi and Kuechler (2008) (Figure 3).

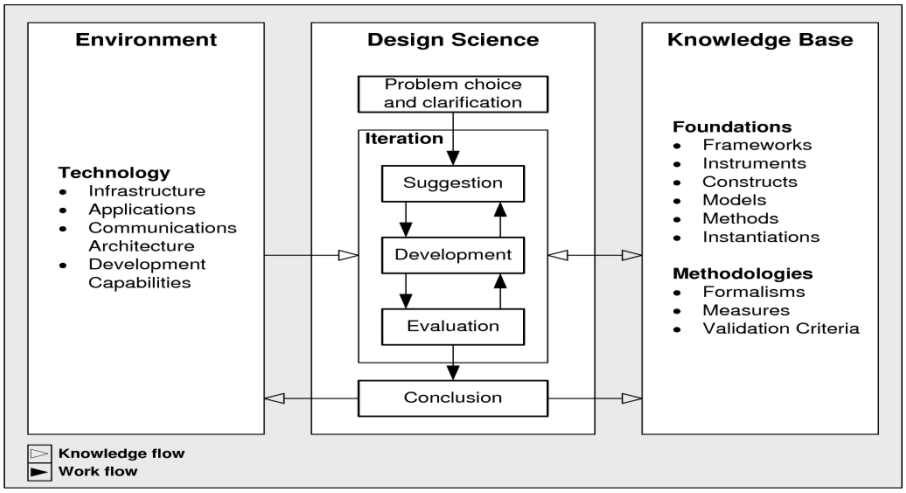

Figure 3: Case 2 development framework (Ejlersen, Knudsen et al. 2008)

Parts of the Hevner framework are omitted; including the whole of the central IS column which is replaced by the process model. Those that remain are those that the experimenters deemed to be relevant to their project. Hevner's seven guidelines also figure in the theoretical foundation for the experiment.

4.2.2 Development process

Initial Research (problem choice and clarification, and investigation of the technology environment and underlying programming knowledge base) consisted of determining the appropriate programming technologies and building a small knowledge base of code examples dealing with sending SMS's and managing contact information. Finally the developers experimented with two integrated development environments (Eclipse and Netbeans) and chose Eclipse.

The development itself was carried out as four iterations, each of which explicitly follows the suggestion/development/evaluation model:

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

Here the programme was working well enough to pass its tests and the experiments stopped. The conclusion phase and resulting knowledge flow was understood as the article writing. A second aspect of the experiment was explicit analysis of the seven design principles in relation to the project as a means of understanding where the principles fit in (are instantiated) in the design effort.

4.3 Case 3 – SMS image transmission

Andersen and Markfoged (2008) describe the development of a mobile application which is able to send and receive images with SMS as the data carrier.

4.3.1 Theory foundation

The theoretical base for the this experiment is again Hevner et al. (2004). The developers omit some parts of the framework to better fit the experiment, but unlike the previous experiments, nothing is added. This trimmed framework is illustrated in Figure 4.

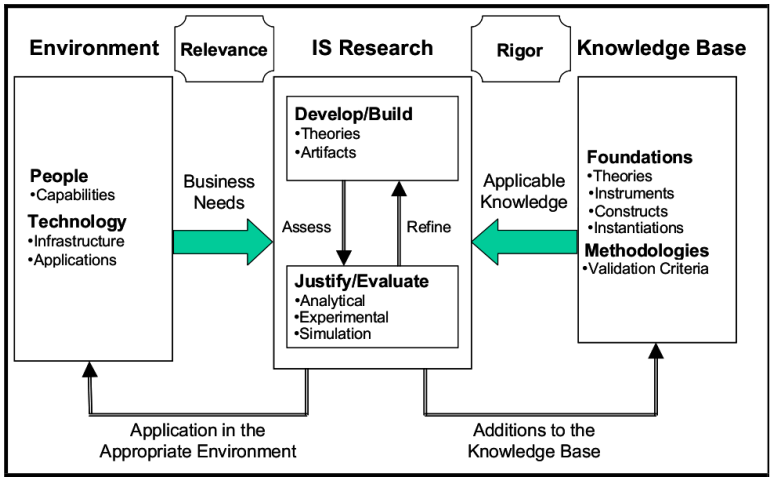

Figure 4. Case 3 framework (Andersen and Markfoged 2008)

A selection of the guidelines (1. Design as an Artefact, 3. Design Evaluation, 5. Research Rigour, 6. Research as a Search Process, and 7. Communication of research) also contribute to the experiment.

4.3.2 Development process

In this project the experimenters used the Environment part of the framework to build a brief analysis of their product idea – developing an understanding of who would use the product, what the available

technical infrastructure enabled or constrained and what the basic features of the application would be. A feature of this project is again the use of a project knowledge base, this time more extensive and built up iteratively through the life of the project. The developers again used the database primarily to build up knowledge of coding techniques relevant to their project. The project followed a cycle of iterations according to the Hevner model; however these were conceived both as building the program (the artefact) and as building the necessary knowledge to progress further in the programming (theory). Evaluation and Justification consisted of both assessing the current state of the program in relation to the project objectives, and determining which programming knowledge should be added to the knowledge base. Hevner's guidelines were used not to determine what should be done, but to characterize how it should be done. Some examples of the iterative building of code and programming knowledge follow.

The first development iteration revealed the need for a custom GUI and knowledge about GUI implementation in J2ME was added to the knowledge base. The knowledge was used for implementation of the GUI, in alter iteration. During initial development it was believed that a subset of J2ME call PIM could facilitate file browsing. Shallow knowledge about PIM was therefore added to the knowledge base. Further investigation showed however that PIM was not applicable for the problem at hand, and was therefore not used in further development. When working with images, two different approaches to loading images were explored and added to the knowledge base. A later change in application design meant a change from one solution to the other, but since both solutions were already in the knowledge base, the decision had little extra cost. After having added knowledge about all components of the application to the knowledge base, developers ceased contributing to the knowledge base in favour of using it as a specialized API documentation tool.

5 RESULTS AND OBSERVATIONS

5.1 Experiment trends

The experiments produced a lot of 'framework thinking' – use and adaptation of a diagrammatic type of framework as the principle motivating force for the development. This is not necessarily natural for computer science students, and probably reflects the rather prominent position of framework thinking in the articles they read. Hevner et al's ideas occurred repeatedly, but it cannot be established whether they are particularly suitable for this type of design work, or whether the students picked up a focus on this seminal article in the articles they studied. The frameworks were always adapted and personalized, though the students received no particular instructions to do this. The rationales behind the adaptations are several. The students can be seen to focus on what they consider important - building their technical understandings and programming skills, and to subtract what they are less comfortable with from the frameworks they use. Mainly this is the organisational, the behavioural, the methodological and explicit theory building. All forms of interactions with users or organizational settings are removed usually with the (only partially justified) reasons that the time spans are too short, and that the applications are generic (and therefore have no specific setting).

Sets of design principles and process models are commonly adopted as a suitable basis for the development projects. The students are rather familiar with development process models from the education and earlier products. Sets of design principles are understood as normative prescriptions for design work, and therefore suitable foundations for a project. Students commonly adopted an iterative design process model (although more linear models are available in the literature) and deployed it as a mixture of iterative and incremental development. This represents a natural way of thinking for them as they have experience of, and respect for agile methods. The idea of a knowledge base, which is a relatively unfamiliar idea in systems development work, was adapted in a particular way to represent that body of knowledge which the students found necessary to acquire in order to complete their projects. Evaluation, which is a rather extensive concept in design science, was interpreted as program test. The students felt free to integrate other techniques (for example pair programming) which are not part of their reading about design science. Typical for the projects was some kind of experimentation and knowledge gathering phase (called *Initial Exploration* in case 1), where the necessary programming techniques were assembled and mastered before the development

work proper started. This is not explicit in design science, but probably stems from a concentration on iteration and knowledge. Design science concepts which were more appropriate to research than software development (theory, research rigour) were not ignored, but simply twisted so that they had a direct bearing on the immediate programming work.

5.2 Case results

All the development experiments proceeded smoothly and painlessly, without significant deviations from original expectations, and were completed in a relatively stress-free way within the time constraints. All three applications worked as expected and passed their tests. All three development teams agreed that design science contributed significant process awareness to the development. The case 1 team note that "*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.*" Case 2 concludes that: "*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 - this has increased the quality of the developed application.*" The third team understand it slightly differently: "*the research-like approach makes the development very controlled and focused in an intuitive way.*" Another common reflection is that the iterative development that the DS frameworks encourage is a positive trait: "*the iterations also helped to keep development going...*" (case 2); "*the iteration process proved invaluable in extending the functionality of the program once the original goals had been met.*" The initial exploration phase, which, though not directly attributable to design science process models, seems at least to have been provoked by process reflection and consideration of design principles, was considered very useful: "*the task of formulating the initial theories ... helped identify the challenges of the problem.*" (case 2); "*another important point is the initial exploration, which contributes greatly to the development process.....the initial explorations from our model helped develop a good knowledge base, which eased the rest of the development.*" (case 1). 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.....what seems to make DS shine in this context is the knowledge base...*" Mobile development is less well understood than older programming forms and therefore riskier – team 3 comment that "*DS seems to be a good choice for developers who work with new and unfamiliar API's and technologies*." The generic nature of design science was appreciated: "*design science .....seems to be an optimization to any kind of development project*" (case 2). A final consideration is the role of meta-understandings of design work in the experiments. "*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*" (case 1).

5.3 Limitations of the experiment

Positive results from the experiments should be tempered by difficulties with validity and generalisation. Experiments with student programmers are common in the literature, but it cannot be assumed that more experienced developers would reach the same conclusions. Though the documentation and reflection demands of these experiments are high, the discovery experiment remains a loose form of investigation, without defined measurements or the normal controls of experimental research. There is probably some form of Hawthorne effect, where the focus on experimentation affects the results. Lastly the development projects undertaken are very small, and there should be no natural expectation that these results can scale up to even small commercial projects.

6 CONCLUSIONS

A certain scepticism should be registered in the question posed in the title of this article: can design science be used for design? For those of us used to working with software engineering techniques, design patterns, software support environments and development method, the simple prescriptions of design science appear both naive and superficial. The surrounding context seems abstract and theoretical. The expectation when embarking on these experiments was that the answer would be a fairly resounding 'no.' Instead the student experimenters, though experiencing a steep learning curve and some degree of culture shock, were able to adapt the simple precepts and process models naturally and effectively into their development world. Principles and processes seemed both to echo, and to structure what they had learned in their education, and what they had experienced (partly as confusion) in earlier development projects. It cannot be assumed that we experimenters had very thoroughly understood what we read; we made conscious adaptations to the models we found, and incorporated conscious and unconscious misunderstandings into frameworks for development efforts. The frameworks were followed in an improvisatory fashion, rather than a rigorous methodical way, in the spirit of professional engineers doing their best to solve non-trivial design problems. Nevertheless they were found to be useful, perhaps because they were able to provide structure at a meta-level to tie together the various disparate experiences of system development. Perhaps they operated as a kind highly level concept map, which allowed the experimenters to investigate different parts of the project, without losing sight of the hermeneutic whole.

Though these small experiments provide no basis for trying to operationalize design science in any commercial development environment, they do provide justification and motivation for two different kinds of further experiments. The first is that the experiments give some grounds for investigating whether system developers could benefit from studying design science as a routine part of their education. The second is that it becomes interesting to study whether design science can be used as a structuring factor in more realistic development efforts. To this end we have organized a larger scale (six-month) experiment, where we will work on a development of an innovative mobile system for supporting dyslexics. The development will take place in collaboration with an innovation greenhouse and a group of users (dyslexic children), where the intention will be to provide a prototype of a commercially marketable system.

Acknowledgements

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

Aaen, I. (2008). Essence: Facilitating Agile Innovation. Agile Processes in Software Engineering and Extreme Programming. P. Abrahamsson, R. Baskerville, K. Conboyet al. Berlin Heidelberg, Springer. 9: 1-10.

Alberts, D. S. and R. E. Hayes (2002). Code of Best Practice for Experimentation. DoD Command and Control Research Program, Office of the Assistant Secretary of Defense. Washington, D.C.

Alberts, D. S. and R. E. Hayes (2005). Campaigns of Experimentation: Pathways to Innovation and Transformation. Command and Control Research Program, Office of the Assistant Secretary of Defense Washinton. D.C.

Andersen, J. L. and K. Markfoged (2008). Design Science applied as a Development Framework in an Unfamiliar Programming Environment. unpublished https://services.cs.aau.dk/public/tools/library/files/rapbibfiles1/1212138697.pdf.

Arnott, D. (2006). "Cognitive biases and decision support systems development: a design science approach." Information Systems Journal 16(1): 55-78.

Carroll, J. M. (1993). "Creating a Design Science of Human-Computer Interaction." Interacting with Computers 5(1): 3-12.

Cross, N. (2001). "Designerly ways of knowing: Design discipline versus design science (The 1920s and the 1960s, two important periods in the modern history of design)." Design Issues 17(3): 49-55.

Ejlersen, A., M. S. Knudsen, et al. (2008). Using Design Science to Develop a Mobile Application. unpublished https://services.cs.aau.dk/public/tools/library/files/rapbibfiles1/1212143646.pdf.

Gregor, S. and D. Jones (2007). "The anatomy of a design theory." Journal of the Association for Information Systems 8(5): 312-335.

Haeslegrave, J., T. Justesen, et al. (2008). Design Science: Applying the Anatomy of a Design Theory to a Mobile Application. unpublished https://services.cs.aau.dk/public/tools/library/files/rapbibfiles1/1212344842.pdf.

Hevner, A. R., S. T. March, et al. (2004). "Design science in Information Systems research." Mis Quarterly 28(1): 75-105.

March, S. T. and G. Smith (1995). "Design and natural science research on information technology." Decision Support Systems 15(4): 251-266.

Marxt, C. and F. Hacklin (2005). "Design, product development, innovation: all the same in the end? A short discussion on terminology." Journal of Engineering Design 16(4): 413-421.

Mathiassen, L., A. Munk-Madsen, et al. (2000). Object-Oriented Analysis and Design. Aalborg, Marko Publishing ApS.

Purao, S. (2002). "Design Research in the Technology of Information Systems: Truth or Dare." Atlanta.

Vaishnavi, V. K. and W. Kuechler Jr (2008). Design Science Research Methods and Patterns. Boca Ranton, Fl, Auerbach Publications.

Walls, J. G., G. R. Widemeyer, et al. (2004). "Assessing information system design theory in perspective: How useful was our 1992 initial rendition?" Journal of Information Technology Theory and Application 6(2): 43-58.

Zachman, J. (1999). "A framework for information systems architecture." IBM systems journal 38(2/3): 454-470.

---

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

[ii] https://www.cs.aau.dk/~markus/safe_html/dat4/doku.php?id=start