CoDisclose: An Approach to Disclosing Design Rationale

Research-in-Progress

Jonas Sjöström[1, 2] (jonas.sjostrom@im.uu.se)
Owen Eriksson[1] (owen.eriksson@im.uu.se)
Pär J. Ågerfalk[1] (par.agerfalk@im.uu.se)

[1] Department of Informatics and Media, Uppsala University
[2] Department of Public Health and Caring Sciences, Uppsala University

Abstract

In exploratory research, research questions may emerge at a late stage in design-oriented research projects. In such case, researchers are faced with the complex task of reconstructing design rationale, i.e. retrospectively scrutinizing certain aspects of the design process. While such a task may be challenging due to the human and technical complexities in design, we propose an approach to support the disclosing of design rationale. The approach presumes an iterative design process where stakeholders are continually exposed to new releases of software. The approach is comprised of three design science artifacts: A histogram, a workflow, and a software artifact. The histogram is a quantification of both design dialogue and actual changes in the source code. The workflow suggests an iterative process of quantitative and qualitative inquiry. While the approach remains to be rigorously evaluated, we discuss its background and use in an ongoing design project in the eHealth context. Implications for research are discussed.

Keywords: Design research, action research, exploratory, design rationale

1. Introduction

Design-oriented research is often exploratory (Goldkuhl and Lind 2010). At the inception phase of a research program, IS researchers may identify several interesting topics that are hard initially to phrase as distinct research questions. *Design* is often characterized as a search process (e.g. Hevner, March, Park, & Ram, 2004), a characterization that is true also of design-oriented *research* (Sjöström and Ågerfalk 2009). The latter suggests that although working on a reasonably structured design problem, the understanding and positioning of expected outcomes in terms of research contributions may be allowed to evolve over time as the understanding of the problem space increases.

Design-oriented research may be based on a design practice that makes continual releases of artifacts into an ongoing workpractice. A software release, whether into beta-testing or production practice, resembles an intervention in action research. The new release may be based on (i) requirements from stakeholders (possibly based on their assessment of the current version), (ii) theory proposed by researchers that ingrain the new release, or (iii) creative work by designers and/or researchers. Despite a considerable and long-term research interest in design rationale (e.g. Conklin and Yakemovic 1991; Horner and Atwood 2006; Maclean et al. 1996) , design decisions are often not well documented, especially not during the early stages. This may be due to the extra work involved in documenting design rationale, and to the fact that designers, just as any decision makers, may rely more on intuition than on formalized decision models (cf. Akinci and Sadler-Smith 2012). Relying on intuition does not mean that decisions are ill informed or irrational. On the contrary, in keeping with Weber's notion of practical rationality, such decisions are governed by norms and pre-understanding that serve as tacit grounds for design activities

(Ågerfalk and Fitzgerald, 2006). Another reason why it is difficult to document the design rationale in real-world design projects is that the design is often pursued in contexts where restrictions, e. g. time, makes it difficult to manually document the process in a proper way. In a situation where research questions are vague and the documentation of design rationale is missing or is inadequate, a series of naturalistic evaluation (Venable et al. 2012) activities between each release may also be enacted in order to capture design rationale incrementally.

Over time, initial research interests emerge into well-specified research questions (Sjöström 2010). When this 'shift' occurs, there may be a need for retrospective analysis to reconstruct the design process and the rationale behind design decisions. Such analysis may be required to understand the steps through which the design has evolved. In an action design research (ADR) context (Sein et al. 2011), for example, it would be desirable to account for the design cycles through which design principles have evolved. In an action research context, researchers would explain the cycles to show the rationale for each intervention, and how each intervention contributed to knowledge development (Järvinen 2007; Susman and Evered 1978) (Eriksson and Goldkuhl 2013; Järvinen 2007; Susman and Evered 1978). In a similar manner, the rigor and relevance of a design science research (DSR) endeavor would be partially evaluated based on the presentation of the iterative interplay between theory, practice, and design/evaluation.

As outlined above, design-oriented research highlights the cyclical character of research; how knowledge and its rationale emerge through iterations. As a design-oriented researcher, one must find ways to retrospectively scrutinize a design process. Although there may be existing field notes *et cetera*, it may still be complex to effectively and efficiently analyze – let's say – a three year long design research effort.

This paper elaborates on an approach to support retrospective analysis of design processes. We introduce the CoDisclose approach, which is a way to quantify aspects of a design process, and visualize the results on a timeline. The name refers to the idea that we should disclose the actual changes in software and its underlying rationale for scrutiny. CoDisclose is an inquiry into both (i) actual code changes in a software repository, and (ii) discussions among stakeholders about requirements, design rationale, *et cetera*. The fundamental idea in the approach is to visualize these two aspects of the design process. The approach presumes an iterative design process, in which stakeholders are exposed to (or appropriate) continuous releases of the software artifact. Although there are other approaches that can be used to do retrospective analyses of method rationale (Lee 1997), these typically rely on software that captures method rationale during the design process (Horner and Atwood 2006). An advantage of the CoDisclose approach is that no such, more or less intrusive, logging of design activities is required.

The paper proceeds as follows. Section 2 outlines the research approach adopted. Section 3 presents the CoDisclose approach and its evolution within a design-oriented research project. Finally, Section 4 presents a concluding discussion and sketches plans for future research.

2. Research Design

This work was conducted as part of the U-CARE research program at Uppsala University. Within the research program, a fairly complex piece of software was designed in the context of eHealth and eHealth research. The software measures to approximately 50 000 lines of code, and the underlying database has 100 tables. Development has been going on since mid 2010.

The overarching program is a multi-disciplinary effort including researchers and practitioners from psychology, medicine, information systems, caring sciences, and economics. The research program aims at supporting people with potentially lethal somatic diseases to cope with their situation. People with such diseases commonly develop depression and anxiety, which increases human suffering. It may also impact the treatment of somatic disease negatively, e.g. if a depressive state causes a patient to engage in less physical activity, develop sleeping problems, or forget to adhere to their medication subscriptions. Internet-based self help has proven effective for psychiatric disorders as well as for promotion of health behaviors (Barak et al. 2008; Riley and Veale 1999; Ström et al. 2000). It is promising both with regard to treatment efficacy and costs, by using less therapist time per effectively treated patient compared to face-to-face therapy (Tate and Finkelstein 2009; Warmerdam et al. 2010).

From an IS research point of view, the research program was clearly interesting. It provided an opportunity to participate as design researchers in the design and construction of a web-based software for Internet therapy. The societal relevance of the psychological research was well presented in the program. The researchers were not able to specify clear research questions at the inception of the design process, due to three factors. *First*, in order to phrase high quality research questions, a lot more knowledge was needed. That knowledge would emerge over time, through literature studies, experiences from the design process, and from the multi-disciplinary collaboration. *Second*, IS researchers entered the program in a late stage. The funding for research was already present, based on a grant awarded to the psychology department. That is; IS researchers were not part of the original planning of the program. *Third*, the multi-disciplinary environment essentially led to a clash of research ideals that – from an IS point of view – constrained innovation. Thus, the approach to IS research in the program was exploratory, and in the first 1-2 years, the IS staff in the program were highly involved in practical design and construction work, continually assessing how to turn that work into interesting IS research.

Doing exploratory design-oriented research is an investment with uncertain outcomes. By acting as designers and developers, the IS researchers gain first-hand experiences of all sorts of challenges related to development of eHealth work practices and software. In addition, IS researchers were in control of the technicalities of the development process, which provides access to people, design documentation, and the source code and its history. The design process was set up in accordance with agile values (Conboy 2009), characterized by sprint reviews approximately every two weeks. The review meetings had several recurring members representing different professions and academic disciplines. In addition, external specialists and patient groups were invited to explore the software, followed by workshops in which they provided feedback to the design team. In total, 80+ design workshops were organized, engaging a great variety of stakeholders. IS researchers contributed with knowledge from the IS field and related disciplines (primarily interaction design and software engineering). Over time, several research questions emerged, which to date have rendered several publications in IS conferences (Mustafa and Sjöström 2013; Sjöström and Alfonsson 2012; Sjöström et al. 2011, 2012; Sjöström, Rahman, et al. 2013).

Whenever the design rationale for some subset of the design was to be scrutinized, it was complex to identify important design episodes that had taken place for that particular subset. One example of such a subset is the design of an indicator dashboard to support online therapists (Sjöström and Alfonsson 2012). It was time-consuming to reconstruct the design rationale for that particular feature in the software, since research on the topic was not planned in advance. Similar situations have occurred, e.g. concerning design of translation functionality (Sjöström and Hermelin 2012), data export design (Mustafa and Sjöström 2013), privacy and accountability issues (Sjöström, Ågerfalk, et al. 2013) *et cetera*.

A search for software tools to support empirical mining was conducted. Several products were identified, but none that provided the functionality that was desired. Existing tools either focused analysis of documents or software engineering-oriented queries into code repositories. The need in this situation was to find a combined view that revealed both how software *and* stakeholder discussions emerged over time. In order to support researchers in this type of situation, a set of design science artifacts (Hevner et al. 2004) was devised to facilitate a 'filtered' retrospective analysis of the design process. A piece of software was built to support analysis. The software has so far been applied by IS researchers within the program to support their retrospective design process analysis. There is little experience to demonstrate the qualities of the artifact outside the initial research context, but we do make a tentative assessment of the approach through an informed argument. The construction of the analysis software is in itself a design-oriented research effort. The development and use of evaluation methods and new evaluation metrics provide design-science research contributions (Hevner et al. 2004). In this paper, we outline the characteristics of the artifact as proposed by Gregor and Hevner (2013), followed by an informed argument (Hevner et al. 2004) to assess the qualities of the proposed artifacts.

3. CoDisclose: An Approach to Disclosing Design Rationale

The main artifact in the approach is the Design Cycle Histogram (Figure 1) that visualizes code and talk on the same timeline. The *dotted* line in the histogram – representing requirements discussions – was produced by querying for keyword frequency in a document repository containing notes from sprint meetings. The *solid* line represents actual software revisions. It shows how revisions in the code repository contain changes to files matching keywords related to the development of peer interaction

features such as online forum, chat, internal messaging between peers, and features to support staff in analyzing peer interaction.

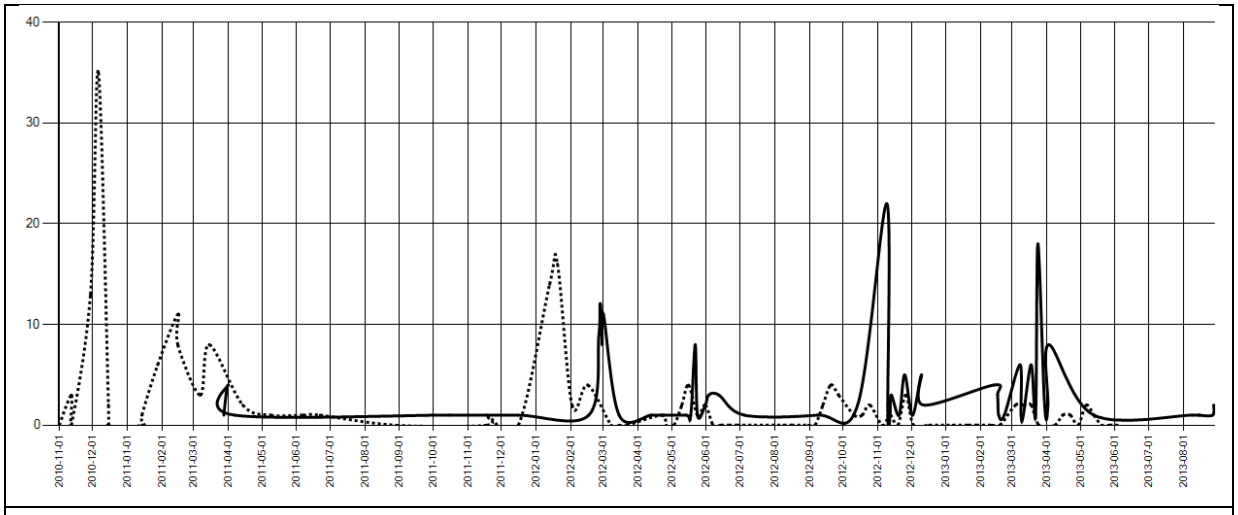

Figure 1 - Design Cycle Histogram

Through the visualization, we can intuitively identify the design cycles through which a subset of the software evolved. Essentially, the histogram is a quantification of design and construction activity mapping to a research interest in design rationale. The production of the histogram is performed during the first step of the CoDisclose Approach. A workflow model of the approach is described in Figure 2. The workflow phases are explained in the following sections.

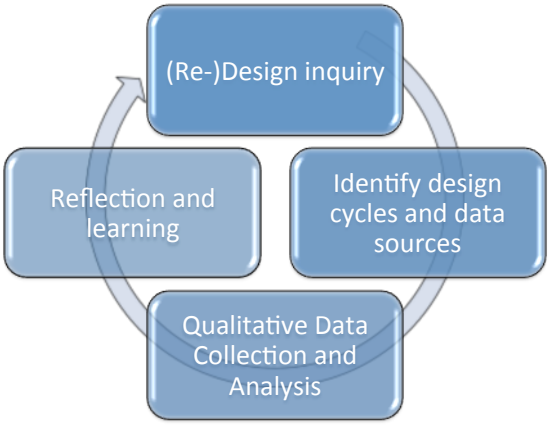

Figure 2 - Workflow Model

3.1. (Re-)Design Inquiry

In order to produce the histogram, we need to setup an inquiry by identifying keywords related to the current research interest. The histogram in Figure 1 is based on an inquiry focusing the design of peer interaction features. The keywords for the inquiry are shown in Table 2. Identifying appropriate

Repository keywords definition requires either technical insight into the solution, or a dialogue with knowledgeable developers. Keywords to search the document collection were quite similar but not identical. Those keywords would be identified in collaboration with those who frequently participated meetings, i.e. knowledgeable about how various stakeholders would refer to design related to peer interaction.

Table 1 – Example of keywords used to inquire into the design process

Keyword	Explanation	Keyword type
IUcareUtterance	Interface in moderation design	Code Revision
Monitor	Source code related to monitoring/moderation	Code Revision
ForumPost.cs	Partial implementation of ForumPost class	Code Revision
Chat.cs	Partial implementation of Chat class	Code Revision
ForumIndex.cs	Partial implementation of ForumIndex class	Code Revision
Diary.cs	Partial implementation of Diary class	Code Revision
Message.cs	Partial implementation of Message class	Code Revision
Moderation	Used when discussion peer interaction monitoring	Discussion
Chat	Used in discussions about the chat feature	Discussion
Forum	Used to discuss forums in general	Discussion
Abuse	Used in discussions about improper peer behavior	Discussion

In order to render the histogram, there is a need to mine and quantify both project documentation and the code repository. The CoDisclose approach is empowered by a software artifact dedicated to such mining and quantification. The software consists of three packages. *First*, the import package, which encapsulates logic to analyze external data sources and index them in a local database. The import function scans for changes in the data sources and updates the local database when needed. Storage in the local database increases analysis performance. *Second*, the analysis package, which queries into the local database to extract relevant data. *Third*, the visualization package, which renders the actual histogram based on analysis results.

Table 2 - Data sources

Type of data	Source	Example
Code Revision	Code repository	Code changes committed by a developer to a Subversion repository.
Discussion	Documentation	A Pdf document with notes from a development sprint meeting.

Table 2 shows the data sources needed for analysis. To analyze project documentation, data is imported from the document repository, including timestamps and the text as such. A link to the original document is also stored, to facilitate quick access. Concerning code revisions, we import among other things revision numbers, name of the programmer who committed the code, a revision timestamp, as well as all files that were modified in the revision (metadata and full text).

By indexing documents and code revisions in the local database (Figure 3), we facilitate SQL queries into the design process history. The CoDisclose histogram utilizes this possibility. To produce a histogram, we need to define an inquiry. An inquiry is defined by pointing out a document collection and a code repository, and a set of keywords mapping to each collection (as shown in Table 1). The actual analysis – at this point – creates a keyword count for each document. This is used to generate the Y-axis for the document series in the histogram. The Y-axis for he repository series in the histogram is the sum of the number of files changed (modified/added/deleted) matching the keyword set in the inquiry. Once the

keywords are defined, i.e. the when the inquiry has been **setup**, the software is executed to **perform the analysis** and render the histogram. This is represented by the ChangePath class in Figure 3.

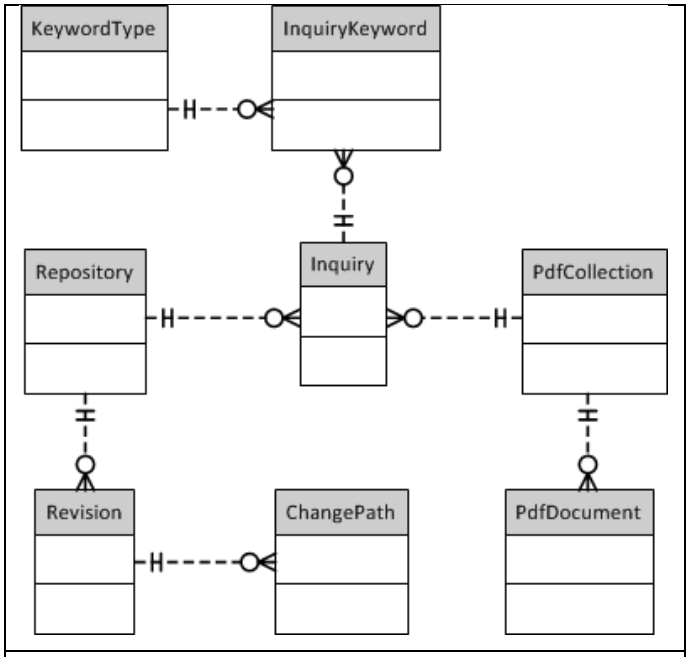

Figure 3 – The CoDisclose Conceptual Model

Figure 3 shows a static conceptual model corresponding to the setup inquiry phase. The model illustrates all the key constructs and their relations. It also reflects the schema in the local database referred to above. The model – and the storage of repository and discussion data in the database - opens up the possibility to appropriate SQL queries to extract and aggregate information from the design process.

3.2. Identify design cycles and data sources

At this point, the histogram may be rendered and needs to be interpreted. Each identified design cycle is subject to further inquiry to reconstruct important design decisions and their rationale. In this stage the software may additionally support us through queries into the local database, helping us identify relevant information sources that could be used in the further analysis of the design process. In abstract, we have access to meeting protocols and code. These are the starting point for further inquiry as outlined below. The outcome of the previous phase (i.e. the design cycles in the histogram) supports us in finding a relevant subset of meeting protocols and code revisions.

Meeting protocols. The meeting protocols reveal parts of the design rationale for the cycle, but they also reveal important actors in the design process (presuming that such information is included in the protocols). The meeting protocols give valuable information about important issues that have been discussed and decisions that have been made. The commitments made by the developers are especially interesting when validated against how these commitments have actually been realized in the source code. The information about actors is useful in order to get in contact with them for clarifications and interviews.

Code. Similarly, based on the code that has been changed we can track the developers that checked in code changes, or interpret the character of the changes. The information about the developers is useful in order to get in contact with them for clarifications and interviews. It is also of interest to see how these changes could be traced back to the commitments made and documented in the meeting protocols. It is also of interest to notice how and when the developers have informed the rest of the project group about

changes of the software (e.g. through revision reports or in meetings). Another potential useful source to study is source code comments.

3.3. Qualitative data collection and analyses

We need to employ qualitative analysis techniques for the continued inquiry, e.g. interpreting text, analyzing changes in the code and its comments, studying systems documentation, conducting interviews with identified stakeholders, analyzing e-mail communication from the time period, *et cetera*. Our approach does not in detail advocate how to address the qualitative analysis. As an example interviews, prject documentation (e.g. meeting protocols) and code may be subject to qualitative anaysis. Relevant code and documents could be used for phrasing interview questions and interview topics. The meeting protocols and the code could also be interpreted side-by-side, in order to analyze the interaction between elicited requirements, and how and when these requirements have been implemented in the code. Our view is that source code (and revisions to source code) may be qualitatively interpreted (like any other text) in order to reconstruct design rationale. In design-oriented research, the actual code of the implemented artifact may be a key source for qualitative data analyses. The artifact has to be 'read' as a dynamic document – continuously subject to change – that partially explains the design process. Based on such a perspective of code as a source for qualitative analysis, it is important to analyze changes in the source code and scrutinize the reasons behind those changes.

3.4. Reflection and Learning

The last phase, inspired by action design research (Sein et al. 2011), includes that the histogram and the qualitative data are analyzed together. The purpose is to support the understanding of the dialectics between design dialogue and actual changes to the IT artifact. Using such an approach we can analyze how design activities, (the spikes in the diagram) have emerged during the design process. We could also explain why these spikes have occurred, and the interaction between the elicitation of the requirements and developed features in the artifact. The purpose of this dialectic approach is to reconstruct design rationale – both the rationale behind the design process and the developed artifact. This means that the design process could be interpreted in the light of the artifact actually emerged. Drawing on the knowledge of the design rationale, there is a foundation to abstract knowledge from the current design situation to theory (Lee and Baskerville 2003).

The reflection and learning phase aims at learning from the design process, but it may also lead to an increased understanding of appropriate keywords to inquire into the design documentation and the code repository. In such cases, there is a need to iterate the process. A re-definition of keywords leads to a re-iteration of the approach.

4. Concluding Discussion

We demonstrate the qualities of the approach using an informed argument (Hevner et al. 2004). In the introduction, we outlined a problem in design-oriented research: In exploratory research, we may end up in a situation where we need reconstruct the design rationale from a complex design process. We have proposed an approach to support researchers in such situations. At this stage the approach should be seen as a tentative methodology (Hevner et al., 2004) that could be used to reconstruct and evaluate the cyclical design process in action design research, and to help disclose its hidden rationale (Ehn and Bannon 2012).

. The approach consists of three artifacts:

- A visualization technique (design cycle histogram)
    - Quick entry to complex reconstruction of events
- An IT artifact that indexes design documentation and code
    - The IT artifact brings order and facilitates inquiry into design history
    - The IT artifact renders the histogram based on existing design process data

- A workflow model, outlining the relation between the dialectics between the quantitative analysis performed using the software and qualitative inquiry into the design process

First, the software tool as such is a design science artifact. The software is an instantiation serving as a 'proof-of-concept', demonstrating that the concept of performing this type of analysis is feasible to implement into software.

Second, our experiences so far show that the histogram is easy to render using the software. It is possible at any time to get a snapshot of (filtered) design activity. Regarding the quality of the histogram, two important reflections need to be made: (1) It is important to carefully reflect about how to define keywords for the document collection and the code repository, and (2) the metrics shown in the histogram at this point are rather rough. Improved metrics – based on more sophisticated algorithms – would be valuable to more accurately identify activity. While the quality of the metrics is an open issue, we still argue that the histogram adds value to a researcher who needs to identify episodes (design cycles) that need further scrutiny. It provides an entry point for further analysis.

Third, a key issue in this approach is the interpretation of the histogram. When first exposed to a histogram, a lot of questions are raised, such as "what do these spikes mean in the repository series?" Clearly, a number of factors could render such spikes in the histogram. They could imply that there is actually a lot of design activity going on, and that further scrutiny of the design cycle may be an important empirical activity. However, a spike could also indicate that there has been large refactoring period initiated by developers. We argue that this is where extra value is added by the other series in the histogram, i.e. the one showing design discussion activity. When there is activity both in discussion and in code changes, it is fair to assume that there further (qualitative) scrutiny of that design episode is highly relevant for research. This is also where the novelty in the approach resides: *In the integrated and automated analysis of code and design discussion*. The histogram provides an interesting potential to study the relations between stakeholder discussions and actual software changes. In addition to supporting design-oriented researchers 'navigate' design history, it also opens up the possibility to do research on the interplay between domain experts and developers. In essence, the histogram is a novel lens to support a post-hoc view of the dialectics in the design process.

At this point, there is little experience in large-scale appropriation of the approach. It is being tested in two ongoing studies, one concerned with eHealth accountability and one with community translation. The experiences so far show that the approach is supporting in navigating a complex design history and making sense of past events and design decisions. Given the experience so far, we have abstracted the approach as outlined in he paper. Due to the early stage of research on the proposed approach, we may only speculate about the usefulness of the approach in other settings (Lee and Baskerville 2003). However, due to the standard formats in use, we may conclude that the CoDisclose software is applicable also in other contexts. The approach is based on mining data sources that are de facto-standards (Subversion repositories and Pdf documents). These common formats can be found in many commercial and open source projects. Any project that is based on those standards would be possible to study post-hoc. In studies where the researchers have not been involved from the inception of the design project, there is arguably an even greater need for 'design history navigation support'. It is, however, an open issue to what extent other researchers will find the approach valuable and relevant. Continued work will investigate the usefulness of the approach through continued case studies in the current context. The approach also needs to be evaluated further in academic discourse. Peer-review and other feedback mechanisms (such as this workshop) will serve as means for scholarly evaluation (Sjöström et al. 2012) and re-design of the approach.

5. References

Akinci, C., and Sadler-Smith, E. 2012. "Intuition in Management Research: A Historical Review," *Journal of Management Reviews* (14:1), pp. 104–122.

Barak, A., Hen, L., Boniel-Nissim, M., and Shapira, N. 2008. "A Comprehensive Review and a Meta-Analysis of the Effectiveness of Internet-Based Psychotherapeutic Interventions," *Journal of Technology in Human Services* (26:2/4), pp. 109–160.

Conboy, K. 2009. "Agility from first principles: Reconstructing the concept of agility in information systems development," *Information Systems Research* (20:3), pp. 329–354.

Conklin, J., and Yakemovic, K. 1991. "No TitleA Process-Oriented Approach to Design Rationale," *Human-Computer Interaction* (6:3 & 4), pp. 357–391.
Ehn, P., and Bannon, L. 2012. "Design Matters in Participatory Design," in *Routledge Handbook of Participatory Design*, J. Simonsen and T. Robertson (eds.), Routledge, pp. 37–61.
Eriksson, O., and Goldkuhl, G. 2013. "Preconditions for Public Sector e-Infrastructure Development," *Information and Organization* (23:3), pp. 149–176.
Goldkuhl, G., and Lind, M. 2010. "A multi-grounded design research process," *Global Perspectives on Design Science Research* , pp. 45–60.
Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *Mis Quarterly* (37:2), pp. 337–355.
Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design science in Information Systems research," *Mis Quarterly* (28:1), pp. 75–105.
Horner, J., and Atwood, M. E. 2006. "Effective Design Rationale: Understanding the Barriers," in *Rationale Management in Software Engineering*, A. H. Dutoit, R. McCall, I. Mistrík, and E. Al (eds.), Berlin Heidelberg: Springer, pp. 73–90.
Järvinen, P. 2007. "Action research is similar to design science," *Quality & Quantity* (41:1), pp. 37–54.
Lee, A. S., and Baskerville, R. 2003. "Generalizing Generalizability in Information Systems Research," *Information Systems Research* (14:3), pp. 221–243.
Lee, J. 1997. "Design Rationale Systems: Understanding the Issues," *IEEE Expert* (12:3), pp. 78–85.
Maclean, A., Young, R. M., Bellotti, V. M. E., and Moran, T. 1996. "Questions, Options, and Criteria: Elements of Design Space Analysis," in *Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll (eds.), Lawrence Erlbaum Associates, pp. 53–106.
Mustafa, M. I., and Sjöström, J. 2013. "Design principles for research data export: lessons learned in e-health design research," in *Design Science at the Intersection of Physical and Virtual Design*, Springer, pp. 34–49.
Riley, S., and Veale, D. 1999. "The Internet & its Relevance to Cognitive Behavioural Psychotherapists," *Behavioural and Cognitive Psychotherapy* (27:1), pp. 37–46.
Sein, M., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. 2011. "Action design research," (35:1), pp. 37 – 56.
Sjöström, J. 2010. "Designing Information Systems," Uppsala University.
Sjöström, J., and Ågerfalk, P. J. 2009. "An Analytic Framework for Design-Oriented Research Concepts," in *Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009),* , pp. 302–310.
Sjöström, J., Ågerfalk, P. J., and Hevner, A. R. 2013. "Privacy and Accountability in Online Communities: Towards a Theory of Scrutiny," in *Proceedings of European Design Science Symposium 2013*, M. Helfert and B. Donnellan (eds.), To appear in Communications in Computer and Information Science. Springer.
Sjöström, J., Ågerfalk, P. J., and Lochan, R. 2011. "Mutability Matters: Baselining the Consequences of Design," in *MCIS 2011 Proceedings*, Limassol, Cyprus.
Sjöström, J., and Alfonsson, S. 2012. "Supporting the Therapist in Online Therapy," in *ECIS 2012 Proceedings*, Barcelona, Spain.
Sjöström, J., Donnellan, B., and Helfert, M. 2012. "Product Semantics in Design Research Practice," in *Future of ICT Research, IFIP AICT 389*, A. Bhattacherjee and B. Fitzgerald (eds.), Springer, pp. 35–48.
Sjöström, J., and Hermelin, M. 2012. "In-place Translation in Information Systems Development," in *Proceedings of the European Design Science Symposium 2012*, M. Helfert and B. Donnellan (eds.), Switzerland, pp. 88–98.
Sjöström, J., Rahman, M. H., Rafiq, A., Lochan, R., and Ågerfalk, P. J. 2013. "Respondent behavior logging: an opportunity for online survey design," in *Design Science at the Intersection of Physical and Virtual Design*, Springer, pp. 511–518.
Ström, L., Pettersson, R., and Andersson, G. 2000. "A controlled trial of self-help treatment of recurrent headache conducted via the Internet," *Journal of consulting and …* .
Susman, G. I., and Evered, R. D. 1978. "An assessment of the scientific merits of action research," *Administrative science quarterly* JSTOR, pp. 582–603.
Tate, D., and Finkelstein, E. 2009. "Cost effectiveness of internet interventions: review and recommendations," *Annals of Behavioral Medicine* (38:1), pp. 40–45.

Venable, J., Pries-heje, J., and Baskerville, R. 2012. "A Comprehensive Framework for Evaluation in Design Science Research," in *DESRIST 2012*, K. Peffers, M. Rothenberger, and B. Kuechler (eds.), Berlin Heidelberg: Springer Verlag, pp. 423–438.

Warmerdam, L., Smit, F., van Straten, A., Riper, H., and Cuijpers, P. 2010. "Cost-utility and cost-effectiveness of internet-based treatment for adults with depressive symptoms: randomized trial," *Journal of Medical Internet Research* (12:5).