Sustaining the Usefulness of eHealth Research Software

Lessons Learned in Action Design Research

Mudassir Imran Mustafa

UPPSALA
UNIVERSITET

Dissertation presented at Uppsala University to be publicly examined in Lecture Hall 2, Ekonomikum, Kyrkogårdsgatan 10 A, Uppsala, Monday, 10 June 2019 at 13:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Associate Professor Anders Hjalmarsson Jordanius (University of Borås).

Abstract
Mustafa, M. I. 2019. Sustaining the Usefulness of eHealth Research Software. Lessons Learned in Action Design Research. 316 pp. Uppsala: Department of Informatics and Media. ISBN 978-91-506-2758-9.

Research software is vital to advancement in the sciences, engineering, humanities, and all other fields. Scientific research is dependent on the quality of and accessibility to research software. Research software is often developed hastily to solve one-off problems, leading to flimsy code that is not sustainable or usable beyond the lifetime of a given research project and is difficult for researchers, outside of the original context, to use, reuse or extend. It is critical to address the many challenges related to the development, deployment, and maintenance of research software. Therefore, there is a growing concern in the scientific community regarding designing sustainable research software. The academic research context refers to the environment or community concerned with scientific research, sponsored by research grants and public funding. Despite the increasing dependence on research software, software development practices in academia lag far behind those in the commercial sector.

Health care relies on a very complex information technology architecture with many different IT components and also has a highly complex governance structure alongside the very rapid technology development. Additionally, there are ever-increasing demands and needs from health care users for more flexibility, more functionality and making the care transparent and patient-centred. Taken together, this poses significant challenges for eHealth and Information Systems researchers, as each artefact, depending on the context, has different quality characteristics to operationalise the requirements under consideration.

The research objective is to explore what Information Systems researchers and practitioners need to be aware of for sustaining the usefulness of eHealth research software, in the academic research context. This longitudinal action design research (ADR) project, with its three cases, was conducted in an eHealth research project over a period of six years. Contributions from this research include the identification of quality characteristics and their enactment in the actual organisational settings, as well as empirically grounded design principles and a typology for sustaining the usefulness of eHealth research software, based on a formalisation of learning in the three ADR cases. This dissertation also contributes to the method space with the introduction of the augmented action design research (AADR) method, an extension of ADR, on how to conduct multiple ADR projects that build towards an overarching knowledge aim.

Practice contributions are the design and development of internet-based eHealth research software to offer patients psychological treatment and support for issues resulting from physical illnesses, while also providing a chance for researchers to study the effectiveness of the aid provided. The dissertation also contributed in a broader sense to the research software development practice, as the findings extend to research areas in which research software is needed to read and interpret research data, and where software must continue to function so that it allows continued access and use of research data.

Keywords: research software, eHealth, sustaining usefulness, action design research, academic research context, quality characteristics, design principles, fitness-utility, data export, technology adaptation, mobile adaptation, design science research

Mudassir Imran Mustafa, Department of Informatics and Media, Kyrkogårdsg. 10, Uppsala University, SE-751 20 Uppsala, Sweden.

ISBN 978-91-506-2758-9
urn:nbn:se:uu:diva-381473 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-381473)

Dedicated to my parents, wife and daughters, as well as to the memory of my late grandfathers.

Acknowledgements

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

In the name of Allah, the Most Gracious, the Most Merciful

I praise and give thanks to Allah, the Almighty, for granting me the capability to proceed successfully to complete this dissertation.

First and foremost, I would like to express my sincere gratitude to my main supervisor Dr Jenny Eriksson Lundström (Uppsala University) who has supported me throughout this process. Thanks for being patient, believing in me and for helping me develop as a researcher and as a person. Your endless comments have improved my work immensely. I am grateful to my co-supervisor Dr Jonas Sjöström (Uppsala University) for considering me, seven years ago, as a master thesis student and for introducing me to U-CARE. Without that introduction, this experience would not have happened. I am grateful that you allowed me to follow wherever the research was leading. I am grateful to my co-supervisor Dr Helena Grönqvist (Uppsala University) for your feedback and support since my first day at U-CARE in January 2012. I am also grateful to my co-supervisor Associate Prof. Dr Owen Eriksson (Uppsala University) for your brilliant comments which have been invaluable to my research work.

Prof. Dr Pär Ågerfalk (Uppsala University) I am thankful to have had you as a co-supervisor, for considering me as a possible PhD candidate, giving me access to your network and guiding me.

I am particularly grateful to Prof. Dr Louise von Essen (Uppsala University) and my colleagues in the U-CARE research group, Clinical Psychology in Healthcare, for opening their doors and allowing me to conduct research and participate in this multi-disciplinary context. The time I spent at U-CARE was vital for conducting my research, thank you. I would also like to give credit to U-CARE for the financial support that made this dissertation possible.

I am grateful to Prof. Dr Tuure Tuunanen (University of Jyväskylä, Finland) for his critical but insightful comments on an earlier draft of this dissertation at my halftime seminar. The questions, feedback and perceptive ideas from Prof. Dr Maung Kyaw Sein (University of Agder, Norway) during the final seminar were endlessly valuable in finalising this dissertation. I would also like to gratefully acknowledge helpful comments received from anonymous reviewers within Design Science Research in Information Systems and

Technology (DESRIST) community on my conference publications and thesis proposals in the doctoral consortium. The feedback, guidance and support allowed me to expand and further develop this manuscript.

I would like to thank researchers from whom I have been fortunate enough to get guidance in one way or another. These people include Prof. emeritus Dr Lars Engwall (Uppsala University, Sweden), Prof. Dr Sabine Koch (Karolinska Institutet, Sweden), Prof. Dr Carole Goble (The University of Manchester, UK), Prof. Dr Pim Cuijpers (Vrije Universiteit Amsterdam, Netherlands), Prof. Dr Robbert Sanderman (University Twente, Netherlands), Prof. Dr Theo van Achterberg (KU Leuven, Belgium), Associate Prof. Dr Annika Lindahl Norberg (Karolinska Institutet, Sweden), Associate Prof. Dr John Venable (Curtin University, Australia), Prof. Dr Matti Rossi (Aalto University, Finland), Prof. Dr Sandeep Purao (Bentley University, USA), Prof. Dr Samir Chatterjee (Claremont Graduate University, USA), Prof. Dr Shirley Gregor (Australian National University, Australia), Prof. Dr Jan vom Brocke (University of Liechtenstein, Liechtenstein), Prof. Dr Mark Aakhus (Rutgers University, USA), Prof. Dr Sven Carlsson (Lund University, Sweden), Prof. Dr Erik Bongcam-Rudloff (Swedish University of Agricultural Sciences, Sweden), and last but not least Prof. Dr Alan R. Hevner (University of South Florida, USA). Hevner has been an essential source of inspiration and support for my research. Thank you all for your inspiration and support!

I would like to express my heartfelt thanks to everyone at the Department of Informatics and Media (Uppsala University, Sweden). Special thanks to teachers and colleagues: Prof. Dr Andreas Hamfelt, Prof. Dr Annika Waern, Prof. Dr Mats Edenius, Prof. Dr Darek Haftor, Associate Prof. Dr Steve McKeever, Dr Franck Tetard, Dr Claes Thorén, Dr Tomas Eklund, Dr Ylva Ekström, Dr Therese Monstad, Dr Henrik Åhman, Dr Göran Svensson, Dr David Johnson, Dr Simone Callegari, Associate Prof. Dr Lina Eklund, (late) Associate Prof. Dr Vladislav Valkovsky, Madelen Hermelin, Fredrik Bengtsson, Andreas Hedrén, Anton Backe, Christer Stuxberg, and Evelina Andersson. Thanks to all current and past Ph.D. students for making my years in the department pleasant Mohammad Hafijur Rahman, Asma Rafiq, Christopher Okhravi, Siddharth Chadha, Yiming Chen, Kirill Filimonov, Cristina Ghita, Paulina Rajkowska, Martin Stojanov, Laia Turmo Vidal, Thomas Ejnefjäll, Katya (Katerina) Linden, Dr Ruth Lochan, Dr Görkem Paçacı, Dr Daniel Lövgren, Dr Patrick Prax, Dr Mareike Glöss, Dr Sylvain Firer-Blaess, Dr Emma Svensson, Dr Stanislaw Zabramski, and Dr Elena Márquez. Thanks to the most supportive administrative and technical team: Tina Kekkonen, Deqa Farah-Asbury, Lotta Lundell, Eva Karlsson, Eva Enefjord, Christian Sandström, Anna Henriksson, Klara Runesson, Sophie Skogehall, Carina Boson, Lars-Göran Svensk and Pierre Hjälm for their help and support.

Dr Anneli Edman, you are the best teacher and one of my role models. Teaching a Master's course under your supervision for seven years consecutively was an honour. You have set the bar so high it will be a challenge to

keep it up for the rest of my professional life. Thank you for making me feel welcome here from the first day of the Master of Information Systems programme at Uppsala University and encouraging me to continue to the PhD. Thank you for being always available at a personal level and listening to me in difficult times.

I would also like to express my heartfelt thanks to colleagues at the Department of Women's and Children's Health (Uppsala University, Sweden). Especially, I want to thank Associate Prof. Dr Erik Olsson, Dr Martin Cernvall, Dr Sven Alfonsson, Dr Malin Ander, Dr Anders Brantnell, Dr Fredrika Norlund, Dr Mattias Öhman, Fabian Holmberg, Ian Horne, Ylva Hägg Sylvén, Laura Kukkola, John Wallert, and Teolinda Toft.

I would also like to remember my fellow PhD students in the Swedish National Research School of Management and IT (MIT) for making PhD courses cool: Dr Jason Crawford, Dr Fahd Omair Zaffar, Dr Christian Fischer, Dr Edward Gillmore, Dr Daniel Nylén, Dr Sofie Wass, Dr Parisa Aasi, and the list goes on. To MIT for providing a platform which enabled me to share ideas with other MIT researchers and also received constructive feedback from them, I am very grateful.

I would acknowledge the Pakistani community whose presence in Uppsala made me feel I am not too far away from my homeland Pakistan.

Finally, and most importantly, I would like to express my sincere gratitude to my parents Ghulam Mustafa and Riaz Begum, my in-laws Qazi Zahoor ul Haq and Tahira Parveen, my wife Dr Saima Zubair, and my daughters Emaan Mudassir, Fakiha Mudassir and Hamna Mudassir, my siblings Ghosia Tabassum, Tajammal Kamran, and Sofia Tabassum. I would acknowledge my family, relatives, friends and teachers for their prayers and support. May Allah give them long, happy, and healthy lives!

اِس قدر عشقِ نبی ہو کہ مٹا دوں خود کو
اِس قدر خوفِ خدا ہو کہ نڈر ہو جاؤں

ضرب دوں خود کو جو ان سے تو لگوں لاتعداد
وہ جو مجھ میں سے نکل جائیں صفر ہو جاؤں

(کلام: مظفر وارثی)

Uppsala, June 2019
Mudassir Imran Mustafa

Contents

Abbreviations

AADR	Augmented Action Design Research
ADR	Action Design Research
ARL	Augmented Reflection and Learning (a stage in AADR)
BIE	Building, Intervention, and Evaluation (a stage in ADR)
CSS	Cascading Style Sheets
CTMS	Clinical Trial Management System
DBMS	Database Management System
DSR	Design Science Research
eHealth	The use of ICT in health care
FEDS	Framework for Evaluation in Design Science research
FL	Formalisation of Learning (a stage in ADR)
FLV	Flash Live Video
GDPR	General Data Protection Regulation
HADS	Hospital Anxiety and Depression Scale
HTML	Hypertext Markup Language
ICBT	Internet-based Cognitive Behavioural Therapy
ICT	Information and Communication Technology
IM	Instant/Internal Message
IP	Intellectual Property
IS	Information Systems
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
PCs	Personal Computers
PL	Problem Formulation (a stage in ADR)
POC	Proof of Concept
RBL	Respondent Behaviour Logging
RCT	Randomised Controlled Trial
RL	Reflection and Learning (a stage in ADR)
RQ	Research Question
SAB	Scientific Advisory Board
SMS	Short Message Service

SSL	Secure Sockets Layer
SVN	Subversion or Apache Subversion
TDD	Test-driven Development
TFS	Team Foundation Server
U-CARE	Uppsala University Psychosocial Care programme
UI	User Interface
URL	Uniform Resource Locator
UU	Uppsala University

List of Definitions

Word or phrase	Meaning in the dissertation
Academic research context	Academic research context refers to the environment or community concerned with scientific research sponsored by research grants and public funding. Other researchers used similar terms, for example, academia or academic settings.
Artefact	The term artefact is used to describe something that is artificial or constructed by humans, usually for a practical purpose, in contrast to something that occurs naturally (Simon, 1996). Artefact denote the idea of ensemble IT artefacts, as proposed by Sein et al. (2011), recognising the technology as structure view of Orlikowski & Iacono (2001).
CBT	Aaron Beck developed cognitive behavioural therapy in the 1970s. CBT is an active, directive, time-limited, structured approach used to treat a variety of psychiatric disorders, for example, depression and anxiety.
Chronic disease	Long lasting disease that can be managed and may not affect a person's lifespan.
Design principles	Design principles are design decisions and design knowledge that is in-tended to be manifested or encapsulated in an artefact (Gregor, 2002). Design principles recommend how to address a specific class of problems or class of solutions in a range of settings (Markus *et al.*, 2002; Sein *et al.*, 2011; Mckenney & Reeves, 2012).
IS design	"IS design in general emphasises the process of defining, designing, implementing and evaluating architecture, components and features of an information system" (Sjöström *et al.*, 2016).
Psychosocial care	A term that is used interchangeable with psychosocial support, related to meeting the emotional, mental, spiritual and social needs of the individual.
Research participant	Research participant refers individuals who were taking part in the clinical research in the empirical context.
Research study	The research study refers to empirical context related eHealth interventions. In the empirical data sometimes referred to just study. Also, study, study protocol, study design and RCT terms refer to research studies.
Somatic disease	Disease affecting the physical body.
Sustainable	A literal definition of the term 'sustainable' means, "the ability of an activity to endure or function for a certain period of time or perhaps indefinitely".
Sustainable research software	Sustainable research software means that software functionality continues to be improved, supported and, available. Likewise, research software is sustainable when researchers – outside of the original design context – can use, reuse, or extend software with reasonable effort in future projects.

Usefulness	Usefulness is defined as the extent to which a system can be used to achieve specific goals. Usefulness and utility are treated as a synonym (i.e., the quality of being useful).
Utility	The term utility is referred to as a characteristic of the design artefact in addition to other characteristics (e.g., quality, efficacy). In the design science research context, the utility is applied as a utility of a tool, which generally means the usefulness of a tool (Gill & Hevner, 2013). Design artefact utility typically means usefulness (e.g., efficacy to perform task, ease of use, ease of learning, et cetera)
Utility function	Within economics the utility function refers to a function which ranks alternatives according to their utility to an individual in the context of decision-making (Gill & Hevner, 2013). Economic utility involves a complex utility function used to rank alternatives in order to maximise utility.

Note: British English used in this dissertation.

List of Tables

List of Figures

Part I: Inspiration

1 Introduction

This chapter defines design artefacts, research software, sustainable research software, and the academic research context. Further, it examines the challenges associated with the academic research context and eHealth research software design. The chapter concludes with a discussion of the research problem, research aim, research questions, and demarcation of the dissertation.

1.1 Research Software

The advance of information technology (IT) presents enormous opportunities in how research can be conducted. Researchers in all disciplines are increasingly adopting and adapting practices, techniques and digital tools. These new tools – particularly software[1] – allow for more creative and productive research as they facilitate the collection, manipulation, and dissemination of information. In other words, software is an integral part of today's research community, as contemporary research is not possible without it (Goble, 2014). Research software is vital to advancement in the sciences, engineering, humanities, and all other fields (Carver *et al.*, 2018). A portion of contemporary research data depends on, or is retrieved and manipulated by, research software. Differentiated from industrial software, research software is used "to explore, test, verify, disprove, or serve as an aid for academic research ideas rather than bring commercial value directly" (Liu *et al.*, 2008, p. 626). Research software ranges from small utility scripts written by researchers for their own use to highly developed applications (millions of lines of code) with significant user bases.

1.2 Sustainable Research Software

Sustainable research software means that software functionality continues to be improved, supported, and available. Sustainable research software is becoming increasingly important as researchers are gradually moving towards

[1] In this dissertation, *software* is used as an umbrella term for software, software systems, software-intensive systems, and software as a component of an information system.

open research, sometimes referred to as *open science*[2] (Nosek *et al.*, 2015; Aalst *et al.*, 2016; Munafò *et al.*, 2017). This *openness* trend means that research software needs to be kept and maintained as long as the data are relevant. Reproducibility[3] is a core principle of science. Researchers from various computational science disciplines have been calling for reproducibility or reproducible research (Schwab *et al.*, 2000; Laine *et al.*, 2007; Mesirov, 2010; Stodden, 2010; Peng, 2011; Morin *et al.*, 2012). Currently, a significant trend, particularly related to journals in the computational sciences, is that of demanding increased availability of research data and research software code, so researchers can attempt to reproduce published findings or extend the original findings and knowledge (Alsheikh-Ali *et al.*, 2011; Stodden *et al.*, 2013; Munafò *et al.*, 2017; Crick *et al.*, 2017).

Regardless of the context in which the software originated, the sustained availability of research software is essential to reproduce the software-dependent research results (Gentleman & Lang, 2004; Stodden *et al.*, 2013; Stodden *et al.*, 2015). However, research software is often developed hastily to solve one-off problems, leading to flimsy code that is unsustainable or not usable beyond the lifetime of a given research project (Baxter *et al.*, 2012). Furthermore, it is difficult for researchers, outside of the original context, to read and understand the code so as to use, reuse, or extend it. In this context, research software is sustainable when researchers – outside of the original design context – can use, reuse, or extend software with reasonable effort in future projects. Therefore, there is a growing concern in the scientific community related to the design of sustainable research software (Crouch *et al.*, 2013; Katz *et al.*, 2014; Downs *et al.*, 2015; Katz *et al.*, 2016a; Hettrick, 2016; Katz *et al.*, 2016b).

1.3 Design Artefacts

Design Science Research[4] (DSR) has become a recognised research paradigm in the Information Systems (IS) discipline (Gregor & Hevner, 2013), explor-

[2] "Open science is the practice of science in such a way that others can collaborate and contribute, where research data, lab notes and other research processes are freely available, under terms that enable reuse, redistribution and reproduction of the research and its underlying data and methods" (The European-funded project Facilitate Open Science Training for European Research (FOSTER), https://www.fosteropenscience.eu/, [accessed: July 11, 2017]).

[3] Reproducibility is the ability to duplicate an entire analysis of an experiment or study, either by the same researcher or by someone else working independently, whereas duplicating only an experiment is called replicating it [source: https://en.wikipedia.org/wiki/Reproducibility, accessed: July 11, 2017].

[4] DSR traces its root to the 1969 book *Science of the Artificial* by Herbert Simon. The purpose of design is to change existing situations into preferred ones (Simon, 1996). The term DSR was coined by Hevner *et al.* (2004). Multiple terms are used in literature to give the idea of *design as research*, for example, design science, design research and design-oriented research.

ing the relationship between Information Systems design activities and research. Design scientists create *design artefacts*[5] as part of their research (Hevner & Chatterjee, 2010). The term *artefact* is used to describe something that is artificial or constructed by humans, usually for a practical purpose, in contrast to something that occurs naturally (Simon, 1996). Baskerville *et al.* (2015) consolidated the number of ways that design artefacts have been defined in Information Systems literature: design theories (Walls *et al.*, 1992; Gregor & Jonas, 2007); design patterns (Gamma *et al.*, 1995); constructs, models, methods, and instantiations (March & Smith, 1995); design principles (Markus *et al.*, 2002; Sein *et al.*, 2011); design propositions (Romme, 2003); technological rules (van Aken, 2004); new properties of technical, social, or informational resources (Järvinen, 2007); and organisational designs and management practices (Niederman & March, 2012).

DSR focuses on solving problems that have unstable requirements and constraints, as well as complex interactions among their subcomponents (Hevner & Chatterjee, 2010). DSR encourages researchers to develop knowledge that the Information Systems practitioners can use in their professional domain (e.g., Routine Design) to design solutions to their problems (Hevner & Chatterjee, 2010; Alturki & Gable, 2014). A design activity involves a malleable process and artefacts, creativity, and teamwork to produce effective solutions (Hevner & Chatterjee, 2010, p. ix). DSR is not only research about design, but also research through design (Iivari, 2015). In DSR, an artefact is developed and rigorously evaluated to gain an understanding, to address an organisational problem, and to contribute to the knowledge base (Hevner *et al.*, 2004). In other words, DSR is a research approach that aims to integrate design and science (Baskerville *et al.*, 2015) and is therefore well-suited for the research in this dissertation (i.e., associating research software with an artefact and vice versa).

1.4 eHealth Research Software

eHealth involves the use of information and communication technology in health care (Eysenbach, 2001; Pagliari *et al.*, 2005). eHealth is a vital area of the Digital Agenda for Europe and the Europe 2020 strategy presented by the European Commission in 2010. Acknowledging the importance of eHealth, several countries have incorporated eHealth strategies into their national health strategies. eHealth is growing and has produced many innovative interventions (van Rooij & Marsh, 2016).

[5] As this dissertation is aimed at a multi-disciplinarily audience, I use the term *design artefact* for simplicity, to denote the idea of ensemble IT artefacts, as proposed by Sein *et al.* (2011), recognising the *technology as structure* view of Orlikowski & Iacono (2001).

Recently, the number of eHealth applications for patients and citizens has increased tremendously. Some projects have been running as part of Sweden's national strategy for eHealth (Ministry of Health and Social Affairs, 2010) – for example, Electronic Health Records, ePrescriptions, national eID for health professionals, information structure, and terminology (Snomed CT) (Doupi *et al.*, 2010). Sweden's eHealth strategy, which was adopted in 2006 and was updated in 2010, focuses on use, benefit, and deployment of technology for the development of health care and social services (Ministry of Health and Social Affairs, 2010). An issue of continuous importance, in 2016 the eHealth strategy was replaced by a vision for eHealth 2025 (Ministry of Health and Social Affairs, 2016). One of the challenges for the Swedish health care sector is how to provide psychosocial support and psychological help to people who need such help because of physical illnesses. In recent years, cognitive behavioural therapy (CBT) has been proven to be effective for a range of psychological disorders, including insomnia, depression, schizophrenia, eating disorders, anxiety disorders, personality disorders, chronic pain and fatigue, substance use disorders, post-traumatic stress disorders, and other psychotic disorders (Hofmann *et al.*, 2012).

The internet has become more assimilated into the daily lives of most of Sweden's population and offers new opportunities for internet-delivered psychological interventions. Internet-based CBT (ICBT) has several advantages over traditional face-to-face CBT regarding accessibility, timing, and pacing for the individual patient. ICBT reduces the amount of time a therapist needs to spend with a patient and maintain efficacy, as the extent of therapist involvement can be tailored to the actual needs of the patient (Cuijpers *et al.*, 2010; Andersson *et al.*, 2013). The evidence is accumulating that ICBT can achieve similar outcomes as traditional face-to-face CBT (Andersson, 2009; Cuijpers *et al.*, 2010; Hedman *et al.*, 2012; Andersson *et al.*, 2013). ICBT appears to work well in populations with various somatic disorders (Cuijpers *et al.*, 2008). Individually tailored interventions seem especially beneficial for patients with emotional distress associated with a somatic disease (Baasterlar *et al.*, 2011). There is a need for more, and large-scale, studies to determine the effectiveness of ICBT on different types of symptoms (Andersson & Titov, 2014), such as symptoms of anxiety or depression associated with a physical disease (Mattsson *et al.*, 2013; Norlund *et al.*, 2015; Ander *et al.*, 2017; Ternström *et al.*, 2017). However, these interventions are complex to design and evaluate due to legal, ethical, technical, organisational, and methodological challenges (Sjöström, von Essen, *et al.*, 2014).

Innovative use of IT holds enormous potential for health care (Lundberg *et al.*, 2013). However, since IT is no panacea, such interventions need to be researched for their clinical effect and cost-effectiveness. Randomised controlled trials (RCT) represent the gold standard in evaluating the safety and the efficacy of health care interventions. Conducting clinical trials is an important practice in medical research. A clinical trial management system

(CTMS) is crucial in conducting a clinical trial and managing trial data (Raptis *et al.*, 2014). eHealth interventions, typically behaviourally based, can be delivered via the internet (Eysenbach, 2011). eHealth RCTs require a system to deliver and evaluate internet-based interventions, as well as a CTMS. Such a specialised system can be characterised as an *eHealth research software,* considering that its purpose is to conduct academic research in eHealth.

1.5 Research Problem: Sustaining the Usefulness

Hevner *et al.* (2004), March & Smith (1995) and Walls *et al.* (1992) advocate the need for DSR in Information Systems. The purpose of DSR is to achieve knowledge and understanding of a problem domain, and its solution, in the building and application of the designed artefact. Evaluation has been highlighted as a fundamental activity in DSR (March & Smith, 1995; Hevner *et al.*, 2004; Vaishnavi & Kuechler, 2004; Venable, 2006; Venable *et al.*, 2016). Through evaluation, the researcher demonstrates the usefulness and efficacy of proposed artefacts (Hevner *et al.*, 2004). Design is inherently an iterative and incremental activity; evaluation provides feedback to the build phase for further artefact development and the continued design process. Hevner *et al.* describe artefact evaluation as follows:

> IT artefact can be evaluated in terms of functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organisation, and other relevant quality attributes. (2004, p. 85)

Following Hevner *et al.* (2004), many researchers (such as, Vaishnavi & Kuechler, 2004; Tremblay *et al.*, 2010; Gill & Hevner, 2013; Prat *et al.*, 2014; Prat *et al.*, 2015; Venable *et al.*, 2016) have argued for utility, quality, and efficacy of a design artefact as the essential criteria for evaluation methods. In the Information Systems literature, different terms are used in discussing the evaluation of an artefact, for example, characteristics, attributes, and properties. In this dissertation, such characteristics are labelled as quality characteristics following ISO/IEC Standard 25010:2011[6].

Quality characteristics are key factors in ensuring value to stakeholders and can be further used to determine requirements, their satisfaction criteria, and the corresponding measures (ISO/IEC Standard 25010: 2011). Stakeholders assign different weights[7] (or priorities or levels of importance) to different characteristics, depending on their subjective judgment and knowledge (ibid.).

[6] ISO/IEC Standard 25010:2011 provides a consistent terminology for specifying, measuring, and evaluating system and software product quality.

[7] An example of such a weight assignment is assignment of priority to product backlog items, based on different prioritisation factors, by the product owner in the Scrum agile development methodology.

Each artefact, depending on the context, has different quality characteristics concerning operationalisation of stakeholders' requirements (Sjöström, 2010; Helfert *et al.*, 2012). Given the importance of quality characteristics and their significant impact on artefact development, there is much interest regarding research into quality characteristics and how to evaluate artefacts (Hevner *et al.*, 2004, p. 83; Venable *et al.*, 2016).

Gill & Hevner (2013) point out that an artefact's usefulness[8] attribute has been widely used in Information Systems as the main evaluation criterion for its success (Delone & Mclean, 1992; Delone & Mclean, 2003). In this setting, utility[9] and usefulness are treated as synonyms. Gill & Hevner (2013) extended the ideas of evaluation of the artefacts from usefulness (i.e., the quality of being useful) to fitness (i.e., ability to sustain its usefulness over time). To sustain usefulness, artefacts require what Gill & Hevner (2013) refer to as a fitness characteristic (such as novelty, openness, and elegance) which involves the adaptation and evolution of the artefacts. Lakew (2013) further elaborated this as the end-users continuously defining the usefulness of the artefact (i.e., making contextual affordances using the artefact functions and properties), while the designer of the artefact insets fitness characteristics to sustain the usefulness of design artefact (i.e., perpetuates fitness and enables long-term use of artefact). So, in this dissertation, *sustaining usefulness* refers to the appropriation (estimation, inset, and enactment) of fitness characteristics in the design process by the designer(s) to ensure that the functionality of the artefact remains available – improved and supported – and used (*survives*), reused, or extended with reasonable efforts (*reproduced and evolved*) in the future. As fitness characteristics were only recently introduced for DSR, there is a lack of empirical studies that show how designers relate to the fitness characteristics in actual design practice. In fact, Gill & Hevner call for revision and clarification of fitness through future empirical and explanatory research (2013, p. 14).

As commented by Ambati and Kishore (2004), Hastings *et al.* (2014), and Störrle *et al.* (2016), the academic research context is different from the commercial. Hence, in this dissertation, it is postulated that the quality characteristics of design artefacts in academic settings will likely be different from those of other artefacts. The academic research context refers to the environment or community concerned with scientific research, sponsored by research grants and public funding. The academic research context is unique regarding its goals of exploring the unknown and its demands on quality assurance and reproducibility. Despite research software being used in most academic research settings and extensively so in eHealth, characteristics that impact on

[8] Usefulness can be defined as the extent to which a system can be used to achieve specific goals.

[9] In the design science research context, the term utility is applied as the utility of a tool, which generally means the usefulness of a tool. Utility is referred to as a characteristic of the design artefact in addition to other characteristics (e.g., quality, efficacy).

sustaining the usefulness of eHealth research software have not yet been studied in an academic research context or at least not for a prolonged period throughout all stages of its lifecycle (i.e., proof-of-concept, development, production, operational/maintenance, and phase-out or withdrawal). We recall that eHealth interventions are very complex to design and evaluate; consequently, the same applies to eHealth research software. There are additional stakeholders in the academic research context, including those who develop, implement, use/re-use, support, control, study, extend, make decisions, and provide funding. Stakeholders are likely to have differing views of the system and different criteria for success (Venable *et al.*, 2016). Consequently, in sustaining the usefulness of eHealth research software, the stakeholder perspective needs to be included, as the selection of the quality characteristics is also subject to the stakeholders' goals and objectives for the system (Cho *et al.*, 2012).

1.6 Research Aim

Based on the described problems and knowledge gaps, this dissertation aims to contribute with design knowledge on sustaining the usefulness of eHealth research software. The dissertation draws on six years' engagement and eight years of data from designing eHealth research software in an academic research context in Sweden. I seek to develop relevant and useful design knowledge (Hevner *et al.*, 2004). In doing so, I pursue the ongoing quest for answering Orlikowski & Iacono's (2001) call to theorise the artefact in a specific domain (i.e., eHealth research software) (Akhlaghpour *et al.*, 2013). Gill & Hevner (2011) suggested that "we need to move beyond just thinking about usefulness as the nature of utility of the artefact" and ask ourselves "How can I make that artefact sustainable? How can it adapt to change in an environment?". The overall research aim is to explore *what Information Systems researchers and practitioners need to be aware of for sustaining the usefulness of eHealth research software in the academic research context.* The primary research aim has been divided into three sub-questions.

> **RQ1:** Which quality characteristics of eHealth research software impact on sustaining its usefulness in the academic research context?

Researching this question would provide insights on the use of quality characteristics by Information Systems researchers and practitioners in sustaining the usefulness of eHealth research software in the academic research context. This is the main concern of the next question:

> **RQ2:** How do Information Systems researchers and practitioners approach quality characteristics for sustaining the usefulness of eHealth research software in the academic research context?

Further, this dissertation aims to move beyond mere descriptive and explanatory knowledge, into prescriptive knowledge. Design principles are design decisions and design knowledge that are intended to be manifested or encapsulated in an artefact (Gregor, 2002). Design principles recommend how to address a specific class of problems or class of solutions in a range of settings (Markus *et al.*, 2002; Sein *et al.*, 2011; Mckenney & Reeves, 2012). Answers to RQ2 would provide insights to generate and articulate prescriptive knowledge (i.e., design principles) to guide Information Systems researchers and practitioners in sustaining the usefulness of eHealth research software. The design principles are one of the effective ways to capture abstract knowledge about the design artefact in Information Systems (Chandra Kruse *et al.*, 2016). This leads us to the third question:

> **RQ3:** What design principles should guide Information Systems researchers and practitioners in sustaining the usefulness of eHealth research software in the academic research context?

The dissertation is aimed primarily at Information Systems researchers and practitioners. Considering the academic research context and research software, this dissertation might be of interest to researchers and practitioners in other fields. Following Baskerville & Myers' (2002) suggestion, the dissertation also aims to disseminate the research output to a broader audience. The dissertation targets practitioner communities, such as Information Systems, software engineering, health informatics, and research software engineering, providing quality characteristics, design principles, and lessons learned as regards designing and sustaining the usefulness of eHealth research software in an academic research context. The design principles are formulated as a ready guide for the practitioners to apply in their design contexts.

Another target audience is the research community (i.e., Information Systems researchers, design science researchers, and action design researchers) informing about the appropriation of the fitness-utility model and action design research (ADR) method, while presenting longitudinal and rich (in-depth contextualised) empirical cases to show how designers relate to the quality characteristics in actual design practice. One of the aims is to gain an in-depth understanding of the problem domain and how results that emanate from this understanding can be generalised (e.g., design principles), given that multiple cases of evidence emerge in a complex research context during a longitudinal study.

1.7 Demarcation

There are different types of design artefacts (e.g., a construct, a model, a method, an instantiation, or a design theory), but this dissertation's primary focus is on *instantiation* design artefacts (e.g., prototypes, software, and information systems). This research merely examines the design practice regarding Information Systems researchers' and practitioners' general attentiveness to the quality characteristics of eHealth research software, while assessment of usefulness and sustainability of the outcome of the research (efficacy of research), for example, the eHealth intervention itself, is beyond the scope of this research at this point. The research is focused on one eHealth research project in an academic research context that is sponsored through academic research grants and public funding. Thus, it excludes research contexts that are industry-funded, academic-industry collaborations or partnership, as they may differ from purely academic pursuits regarding research objectives, methodological rigour, transparency, duration, and anticipated outcomes (Pham *et al.*, 2016).

1.8 Dissertation Outline

This dissertation is written as a monograph with nine chapters. The chapters are ordered so as to optimise flow and make it easier for the reader to gain a progressive understanding of the research work. Therefore, it is suggested to read the text in sequential order. In some cases, readers may benefit from consulting the appendices for a more comprehensive understanding of the issues in focus, as some text, figures, and tables are presented therein for reasons of clarity. The structure of the dissertation is as follows:

Chapter 1 describes the research problem (class of problems). This chapter defines design artefacts, research software, sustainable research software, and the academic research context. Further, it examines the challenges associated with the academic research context and eHealth research software design. The chapter concludes with a discussion of the research problem, research aim, research questions, and demarcation of the dissertation.

Chapter 2 presents the knowledge base. This dissertation is positioned in the research domain of design-oriented Information Systems research and Information Systems Design. This chapter presents – and motivates the use of – essential theories and introduces the reader to some relevant concepts.

Chapter 3 discusses the empirical foundation. The organisational context is described in detail, considering the importance given thereto in the research method chosen (i.e., ADR). Relevance is a key attribute in DSR. This chapter makes the empirical context (i.e., relevance cycle) explicit.

Chapter 4 explain the research design (i.e., rigour cycle). The ADR method – a practice-inspired DSR approach – is presented. ADR allowed me to engage

directly and collaborate actively with stakeholders, in order to learn about sustaining the usefulness of eHealth research software.

Chapters 5–7 present three longitudinal ADR cases (i.e., design cycles) to provide rich insights (such as design principles) and the evaluation of the artefact in practice. The activities during the iterative Building, Intervention, and Evaluation cycles are described in detail to provide transparency in how the artefact was built, evaluated in practice, and the resulting changes that were made in the relevant context.

Chapter 8 synthesises learnings from the three ADR cases over an extended period (longitudinal) and presents retrospective analysis and reflections.

Chapter 9 discusses, concludes, and communicates the research contributions and present implications for research, practice, and future work.

2 Knowledge Base

This dissertation is positioned in the research domain of DSR and Information Systems Design. This chapter presents – and motivates the use of – essential theories and introduces the reader[10] to some relevant concepts. The first two sections (2.1 and 2.2) familiarise the reader with the DSR approach to researching and designing (including evaluating) research software, in particular to the fitness-utility model that serves as the point of departure for this research. The research software is part of an ecology of technology; Section 2.3 presents the ecology of artefacts theory that is used in this dissertation to acknowledge this phenomenon. The academic research context has many stakeholders and uncertain requirements that evolve over time; therefore, the agile software development methodology was used in the design of the research software in this dissertation. Therefore, a presentation of agile software development and refactoring is made in Section 2.4.

2.1 Design Science Research

Design science traces its roots to the 1969 book *Science of the Artificial* by Herbert Simon. The purpose of design is "to change existing situations into preferred ones" (Simon, 1996). Design scientists creates artefacts: "something created by humans usually for a practical purpose." Walls *et al.* (1992), March & Smith (1995), and Hevner *et al.* (2004) highlight the need for DSR in Information Systems. The purpose of DSR is to achieve knowledge and understanding of a problem domain, and its solution, in the building and application of the designed artefact. DSR encourages researchers to develop knowledge that Information Systems practitioners can use in their professional domain (e.g., Routine Design) to design solutions to their problems (Hevner & Chatterjee, 2010). In DSR, the artefact design cycle (building and evaluating) is informed by the rigour cycle (i.e., providing a grounding from and addition

[10] Information Systems researchers and practitioners are presumably knowledgeable about DSR and Information Systems development. The dissertation also targets to a broader academic research audience. Furthermore, the research presented in this dissertation was conducted in a multi-disciplinary research setting. It is necessary to explain the essential theories and relevant concepts considering the heterogeneous background of the target audience. If the reader is familiar with the topic presented, s/he can skip the section. Nonetheless, the following sections are essential in enabling the reader to understand the empirical context.

to the knowledge base) and the relevance cycle (i.e., inputting requirements and acceptance criteria from the relevant context and introducing the evaluation environment) (Hevner, 2007).

The knowledge contribution of a DSR project is difficult to identify, as one has to recognise the nature of the artefact being designed, the audience to communicate with, the publication outlet, and the state of the knowledge field (Gregor & Hevner, 2013). To facilitate and position DSR contributions, Gregor and Hevner (2013) provided a DSR Knowledge Contribution Framework (a 2 x 2 matrix). The framework consists of two dimensions, i.e., the application domain maturity (the opportunities or problems) and the solution maturity (the possibilities of existing artefacts). Based on these two dimensions, there are four quadrants.

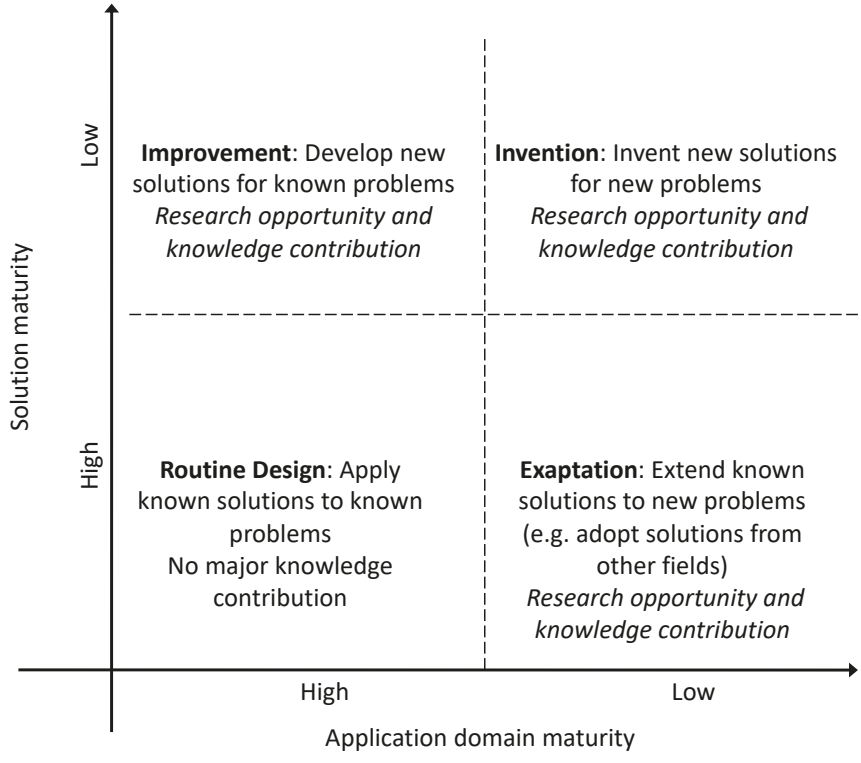


Figure 1. DSR knowledge contribution framework (Gregor & Hevner, 2013).

Figure 1 describes the four quadrants as invention, improvement, exaptation, and routine design. According to the framework, DSR contributions can fall into any one of these quadrants. This framework is used to position the maturity of the artefact of this dissertation (i.e., the U-CARE software system).

2.2 Evaluation

Evaluation has been highlighted as a fundamental activity in DSR (March & Smith, 1995; Hevner *et al.*, 2004; Vaishnavi & Kuechler, 2004; Venable, 2006). Through evaluation, the researcher demonstrates the usefulness and efficacy of proposed design artefacts (Hevner *et al.*, 2004). Evaluation provides feedback for further development and assures the rigour of the research. The extant DSR literature identifies a variety of different evaluation methods (March & Smith, 1995; Hevner *et al.*, 2004; Vaishnavi & Kuechler, 2004; Venable, 2006; Peffers *et al.*, 2007; Tremblay *et al.*, 2010; Sonnenberg & Vom Brocke, 2012; Gill & Hevner, 2013; Prat *et al.*, 2014; Venable *et al.*, 2016). Two methods are relevant to this dissertation regarding the quality characteristics: the framework for evaluation in design science research (FEDS) (Venable *et al.*, 2016) and the fitness-utility model (Gill & Hevner, 2013). Determining the properties to evaluate is an essential step in the FEDS (Venable *et al.*, 2016). During the tutorial at CAiSE 2015 conference, while discussing the FEDS, Venable described an explicit list of quality characteristics (he referred to them as properties) for evaluating design artefacts (Venable *et al.*, 2015). While the list provides an excellent starting point for reflecting and considering quality characteristics, it is not as explicit as the fitness-utility model (Gill & Hevner, 2013) regarding sustaining the usefulness of design artefacts. Gill & Hevner (2013) postulate that artefacts need to exhibit a certain number of fitness characteristics to be of long-term value in society. As long-term value is especially important in the eHealth domain, given the need to sustain eHealth research software, the fitness-utility model is considered as a point of departure for the evaluation method in this dissertation, and is explained in detail in the next section.

The Fitness-Utility Model

Gill & Hevner (2013) provided a view on design artefact quality that expands the prevailing focus on utility as usefulness in DSR. They proposed to consider artefacts' fitness aspects in addition to their usefulness. They understand fitness in an evolutionary sense in two ways: first, an abstract artefact can be adapted to (i.e., be useful in) different contexts (*reproduction*) and, second, an artefact at the instance level can provide sustained usefulness by adapting to a changing context (*survival*).

Gill & Hevner (ibid.) argued that the design system that supports the emergence of an artefact is important for the long-term evolution and diffusion of the artefact in the evolving design landscape. They proposed that designers (in practice) base their designs on more or less explicit utility functions. These choices of the designers evolve as the designers interact with various stakeholders in the design process. An artefact that does not continuously evolve through design actions is likely to become obsolete more quickly than one that

is frequently revised based on new requirements from stakeholders in its application environment. Over time, the evolutionary fitness of a design artefact becomes far more interesting than the use fitness at any particular point in time. Gill & Hevner operationalised the fitness concept by proposing a preliminary list of fitness characteristics to be included in the utility function of the artefact designer(s).

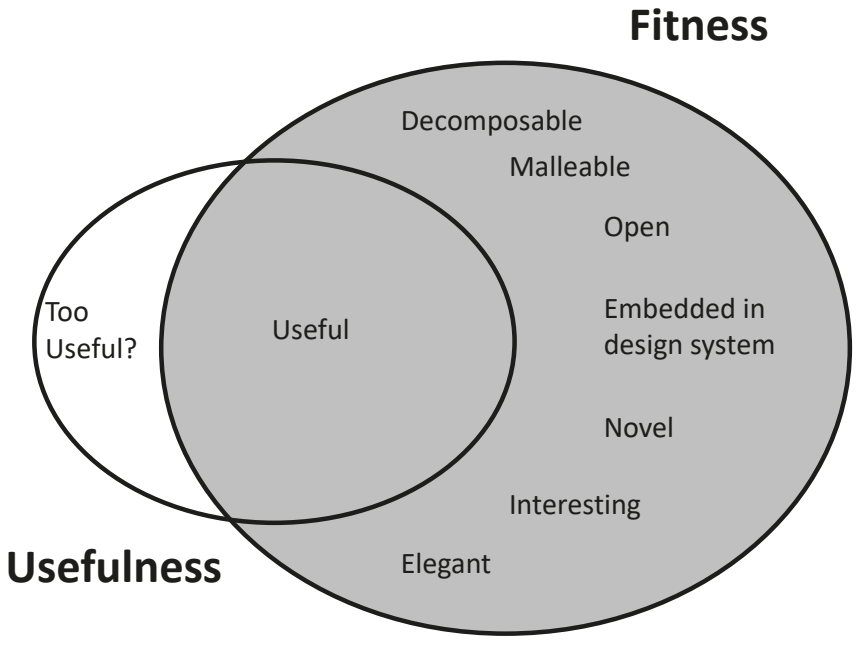


Figure 2. Design fitness characteristics and usefulness (according to Gill & Hevner, 2013).

The fitness-utility model also recognises that usefulness characteristics have a significant role in design fitness, as illustrated in Figure 2. Those characteristics that impact task performance directly can be classified under usefulness (e.g., usability). The area within the fitness ellipse outside of the intersection with the usefulness ellipse reflects other characteristics that impact on fitness (e.g., decomposability).

Table 1. Fitness characteristics

Characteristic	Definition
Decomposability	Artefacts that are decomposable into smaller units allow a redesign of singular units to cope with external changes, instead of requiring redesign of the entire artefact.
Malleability	Artefacts that are malleable can be adapted to cope with changing environments. They can also be adapted to be used for unintended purposes.
Openness	Artefacts that are open for inspection and change allow their end-users a rapid adaptation to changing environments. Malleability, decomposability, and openness are complementary and enhance each other.
Embedded in *design system*	When artefacts are part of systems where design and changes are common, it can be expected that they evolve more rapidly than when design and change are uncommon.
Novelty	Novel artefacts, provided they are viable, can trigger and lead a wave of innovation or change for an entire landscape of artefacts
Interestingness	Interesting artefacts may fascinate designers, researchers, or users and thus lead to a wave of change or innovation – especially when artefacts are novel and interesting at the same time.
Elegance	Artefacts perceived as elegant – in addition to being functional (useful) – may trigger positive reactions in users and therefore be adopted or used more often or have increased longevity.

Table 1 describes fitness characteristics (referred to as quality characteristics in this dissertation).

2.3 Ecology of Artefacts

"Ecology is a theory of how large numbers of species of organisms interact with one another and in that process feed on each other, reproduce, proliferate, find their niches, or die" (Krippendorff, 2006, p. 193). Like biological species, human-designed artefacts also interact with each other, both with their own kind, but also, more importantly, with artefacts of other species; the important difference is that the former interact on their own terms and the latter on human terms. Designers cannot ignore these interactions, as their designs will enter into relations with other artefacts and must be designed to survive such ecological interactions. Designers need to recognise the meaning of an *ecology of artefacts* to produce successful designs (Krippendorff, 2006). Krippendorff suggests that "designers who can handle the ecological meanings of their proposals have a better chance of keeping their designs alive" (2006, p. 202).

In the ecology of artefacts, the meanings attached to an artefact consist of its possible interactions with other artefacts. When designing a new artefact, designers need to build upon existing artefacts and relate their new artefact to the knowledge base (Sjöström *et al.*, 2012). For example, from a developer

perspective, a design artefact depends on software development tools, languages, frameworks, and design patterns; from an architect perspective, it depends on infrastructure; from a management perspective, it depends on processes, procedures, rules, and regulations in the organisation; and from other stakeholders' perspectives, the ecology may look different. From a technological point of view, ecology includes the dependencies between the artefact in focus and boundary objects (e.g., plug-ins, APIs, and frameworks). There is a need to continuously keep an artefact in sync with such boundary objects to promote artefact mutability.

In the academic research context, research software is similarly connected to an ecology of artefacts. Researcher/practitioners need to design new research software using multiple technologies[11] and relate them to the existing ecology of research software.

2.4 Agile Software Development

Globalised and uncertain business environments, combined with rapid advancements in technology, put new strains on software developers. This is especially true in the complex eHealth research environment and academic research context. The software developers have to face constant changes in the user requirements, the organisation, and the environment. Also, user requirements are often hard to define or visualise and can rarely be anticipated to be stable throughout a project. As a consequence, a software development methodology referred to as *agile* has emerged (Cockburn, 2001). In an agile process, software is developed and tested in short repeated cycles. The practitioners claim agile software development to be more responsive to changes in business needs than other, traditional methods (Beck *et al.*, 2001).

[11] I have chosen *technology* as an umbrella term for tools, programming languages, frameworks, libraries, scripts, plugin-ins, add-ons, templates, IDEs, compilers, interpreters, browsers, operating systems, software applications, web applications, web services, design patterns, design principles, design (best) practices, design methods, recommendations, specifications, standards, laws, protocols, and hardware related to Information Systems development.

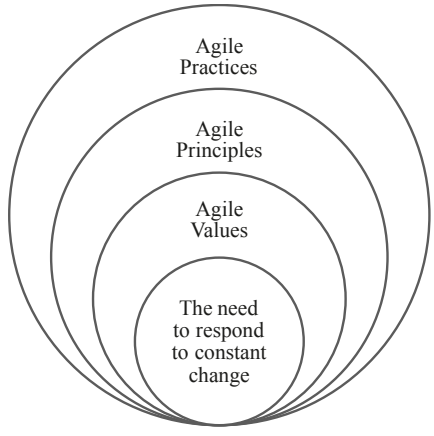


Figure 3. The relationship between agile values, principles, and practices.

Agile is not another software development process – it is more of a mind-set, a way of thinking about software development. The best way to illustrate our understanding of agile is through Figure 3. The agile software development lifecycle is flexible enough to enable organisations to have the ability to respond to constant change. Agile values attempt to focus on what adds value in a software development process. A set of 12 principles, defined in the agile manifesto[12], represents the characteristics of an agile process that welcome change and focus of work are on the customer. For example, the principle – *customer satisfaction through early and continuous delivery of valuable software* – emphasises provision of working software at regular intervals. Similarly, another principle – *regular reflections on how to become more effective* – emphasises the importance of reflecting on how to become more effective, and adjusting the process accordingly.

Agile practices are activities that are used to manifest or implement agile principles and values. The agile mind-set can be applied to any process using any set of practices. There are numerous agile practices, such as user stories, test-driven development (TDD), daily stand-up meetings, et cetera. Specific tools and techniques, such as automated unit testing, continuous integration, pair programming, design patterns, code refactoring, and other techniques are often used to improve quality, enhance project agility, and make it easier to build the product evolutionarily (Beck *et al.*, 2001).

Agile software development involves delivering working and tested software in two- to four-week iterations. As the iterations flow, software developers are coding more and modifying code more, while staying focused on deadlines. Agile teams have to maintain and extend their code a lot from one iteration to the next. Refactoring is used to keep the code easy to maintain and extend.

[12] https://agilemanifesto.org/principles.html [accessed: January 9, 2012].

Refactoring

Martin Fowler popularised the term refactoring (Fowler *et al.*, 1999; Fowler, 2018), defining it as "a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour." In the agile methodology, continuous evolution in software design requires regular refactoring. If not, the code will *rot*, to paraphrase Martin (2000). Every time developers change code without refactoring it, rot worsens and spreads. Code rot frustrates us, costs time, and ultimately shortens the lifespan of useful systems. Refactoring leads to two main benefits: maintainability (it is easier to fix bugs, as the source code is easy to read and easy to grasp); and extensibility (it is easier to extend the capabilities of the application by providing some flexibility where none may have existed before).

Almost invariably, in agile projects, software developers can be so focused on achieving the needed functionality that the software itself grows less comprehensible, more complex, and harder to change. Since such a deterioration of the system usually reflects a lack of refactoring, it can be viewed as a kind of *[technical] debt*[13] that software developers owe the system (Shull, 2011).

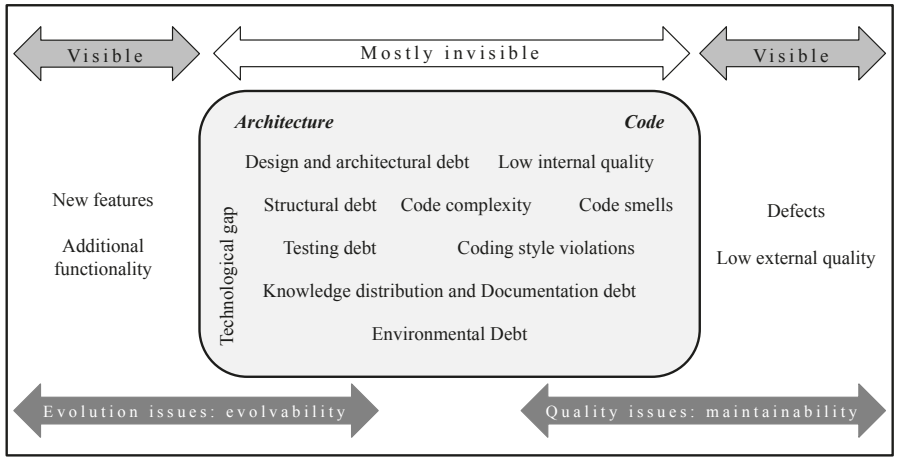


Figure 4. Technical debt landscape (adapted from Kruchten *et al.*, 2012; Tom *et al.*, 2013).

Kruchten *et al.* (2012) presented a possible structure of a technical debt landscape during system evolution. Figure 4 shows a likely technical debt landscape after incorporating constructs like knowledge distribution debt and environmental debt (Tom *et al.*, 2013). New functionalities to be added and defects to be fixed are visible elements, while technical debt is mostly invisible

[13] Technical debt is a metaphor coined by Ward Cunningham (in his experience report presented at the OOPSLA'92 conference) to help software developers think about this problem.

(or rather, visible only to software developers). The ideal situation is to minimise the technical debt in the rectangular box (Kruchten *et al.*, 2012). The left side of the landscape deals primarily with evolution or its challenges, whereas the right deals with internal and external quality issues.

Part II: Relevance and Rigour

3 Empirical Foundation

Relevance is a key attribute in DSR (Hevner, 2007). This chapter clarifies the empirical context (relevance cycle). Researchers need to explicitly consider the context of their research (Davison & Martinsons, 2016). The organisational context is described in detail, considering the importance it is given in the research method chosen (i.e., ADR). The explicit specification of the context in which the research was conducted will facilitate an understanding of the research design and the contexts in which the research results may be useful. The empirical context of the dissertation is the U-CARE research programme, hosted by the research group *Clinical Psychology in Healthcare* at Uppsala University; it has similar complexity and constraints that any academic research context might have. This chapter first presents the U-CARE case organisation. After that, a detailed description of the eHealth research software, the U-CARE software system (the artefact), and the U-CARE design process are given. Lastly, an account of design science research at U-CARE and the representative and unique characteristics of U-CARE as an academic research context are presented.

3.1 Academic Research Context Challenges

The academic research context is continually influenced by and adapted to the academic community through, for example, academic journals, funding agencies, government institutions, and research ethics boards (Sjöström, von Essen, *et al.*, 2014). Despite an increasing dependence on research software, software development practices[14] in academia lag far behind those in the commercial sector. One reason for this is that developing software in an academic research context is very challenging (Hastings *et al.*, 2014). For example, ac-

[14] For example, modularity, documentation, version control, shared code ownership, code comments, coding consistency (coding style), test coverage, code review, test driven development, automated builds and tests, continuous integration, continuous deployment, technical communications, group decision-making, et cetera.

ademic research projects largely depend on public funding and require transparency and openness. Störrle *et al.* summarise these challenges (in his talk[15]) as follows:

> i) a high degree of unknown unknowns, finding out about the requirements is an essential part of the journey; ii) diverse stakeholders with strong opinions, diverging priorities, limited availability, and no software engineering expertise; iii) long-lasting development on a shoestring budget, but with highly qualified personnel; iv) involvement of substantial numbers of junior subject matter experts (a.k.a., students), that are acting as programmers. (2016, p. 1)

Groen *et al.* (2015) argue that software development practices in academia differ from those in commercial development settings and described the reasons as follows:

> the relatively small and transient development teams found in academic settings, […the] project-focused development [… and the fact that] academic software development is rarely performed by full time code developers, and if so, these members are typically hired on short-term contracts with a focus on a specific subset of the code. Most scientific software is developed by researchers such as PhD students and [post-docs] who split their time between code development and research activity. This dichotomy has led to a some-what reluctant and heterogeneous adoption of rigorous software engineering practices in academic contexts. (2015, p. 2)

Groen *et al.* (2015) further argue that limited research has been conducted on assessing software development practices in academia. Liu *et al.* (2008) also point out that there is much less research on the research software development process in an academic context than on the industrial software development process. As the academic research context evolves from a simple to a complex or even a mission/life-critical context, the software can quickly become very complex (Sommerville, 2011). Hence, sustaining the usefulness of the research software (i.e., the design artefact) is important for the long-term evolution and diffusion of the research software in the changing design landscape (Gill & Hevner, 2013).

eHealth research contexts, due to the complexity of health care organisations, are particularly complex (Plsek, 2003; Lipsitz, 2012). Many eHealth innovations do not result in sustainable health care practices (van Gemert-Pijnen

[15] The talk was held at the first Research Software Engineers Conference in Manchester, UK, in September 2016. It is important to note that the working paper (an experience report) regarding the talk was received from Associate Professor Harald Störrle (Technical University of Denmark, hsto@dtu.dk) [February 26, 2017] on personal request, while the abstract of the talk is available at https://ukrse.github.io/conf2016_talks [accessed: September 16, 2016] and the presentation of itself is available at https://drive.google.com/open?id=0BxBBX2ag-SwZhcV9YTFl6VzdqZmM [accessed: September 16, 2016].

et al., 2011). Nijland (2011) identifies three types of difficulties with the uptake of eHealth: slow diffusion, low acceptance, and low adherence. Many eHealth projects fail to survive beyond the pilot phase, so questions remain about how eHealth innovations can be sustainable (Nijland, 2011). Therefore, there is a need for research to fully understand practices in an academic research context that enable sustaining the usefulness of the research software in the complex and evolving eHealth research context.

3.2 U-CARE

The Uppsala University Psychosocial Care Programme (U-CARE)[16] is one of the Swedish government's strategic research programmes, started in 2010 to:

> offer internet-based psychosocial support and psychological self-help to all those who need such help because of a physical illness, no matter where in Sweden they live and no matter what kind of psychosocial resources are available to them locally. (u-care.uu.se, 2012)

The key task of this research programme was to establish the foundation for a new research environment for psychological support and treatment via the internet. In the early stages of the process, the decision was made to develop a specific system to meet the requirements of the programme, since existing systems did not meet the diverse requirements. One way this was to be achieved was in the form of the U-CARE software system, which facilitates patients' psychological treatment and support for issues resulting from physical illnesses, while also providing a chance to study the effectiveness of the aid provided. The U-CARE specialises in the development, piloting, and evaluation of Mental Health interventions for people with somatic disease, and their significant others, for example e-Health interventions. In this ongoing research programme, researchers from caring sciences, clinical psychology, health economics, and Information Systems at Uppsala University collaborate closely to design, develop, and use the U-CARE software system to run randomised control trials (RCTs), deliver eHealth interventions, and study their effectiveness. Initially, the software was built to support three RCTs with their different programmes, to study the effects on anxiety and depression of self-help programmes, delivered via the internet, among adolescents with cancer, among adults with cancer, and among adults after myocardial infarction, respectively. Since its inception, many research groups have started collaborations with U-CARE to use the U-CARE software system to run their studies (referred to as associated studies in the U-CARE context).

[16] http://www.u-care.uu.se [accessed: January 9, 2012].

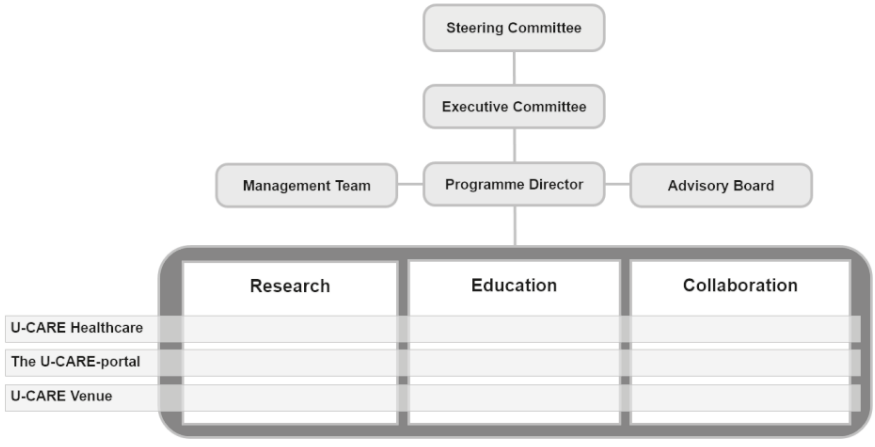


Figure 5. U-CARE organisational chart[17].

U-CARE has gone through some structural changes within its organisation over the years. Figure 5 gives an overview of the present organisation in U-CARE and categories of stakeholders. The steering committee decides on guidelines and establishes regulations for operations. The executive committee is responsible for the implementation of the project plan. The advisory board (also referred to as a scientific advisory board – SAB) consists of well-known expert researchers from various fields as well as people with lived experience of being a patient or relative, and being a research partner. The advisory board reviews the studies and proposes changes. The advisory board meets co-workers within the research programme at least once a year[18]. The programme director leads operations in U-CARE. The advisory board and management team support the programme director in executing research activities. The U-CARE is further divided into three areas:

i) Research – To carry out innovative research of high international quality with regard to the importance of psychological factors for somatic diseases, psychological and economic consequences of somatic diseases, as well as development, testing, and evaluation of psychological interventions.

ii) Education – To offer high-quality education at the undergraduate level, master's level, and postgraduate level in clinical psychology, eHealth and implementation.

[17] http://www.u-care.uu.se/vision-goals-organisation/organisation-and-documents/Organisation-of-U-CARE/ [accessed: November 13, 2017].
[18] SAB meetings were conducted once a year for five years (2011 to 2015). Now, these meetings have been moved to coincides with U-CARE Venue, held bi-annually.

iii) Collaboration – To contribute to interventions developed within U-CARE being implemented within health care.

U-CARE has three activities across three areas:

i) U-CARE Healthcare – provides care to patients (only within research).

ii) U-CARE Venue – a bi-annual interdisciplinary forum for research within the field of psychosocial care.

iii) U-CARE Portal – develops and continuously improves the U-CARE software system, through which self-help programmes are offered and their effects studied.

The treatment provided through U-CARE falls within the framework of research at Uppsala University. U-CARE complies with the Swedish laws and regulations that govern health care, which means that persons receiving care in U-CARE have the same protection and rights as they would when receiving care from any other Swedish caregiver. The treatment provided by U-CARE is free of charge, and it is offered to research participants in the target groups within research studies. As U-CARE provides the care to research participants, it has a health care provider responsibility as well (i.e., U-CARE Healthcare – a system/organisation for the caregiving part). The research participants (only the ones who actually receive care, not control or reference groups, nor in observational studies) are considered to be patients in U-CARE's health care provider context. U-CARE has treatment materials licensed under Creative Commons (CC BY 2.5), to facilitate efficient use/reuse of resources. Such a licence enables treatment material created across various research groups and research studies to be shared.

From the above overview of U-CARE, it is evident that this dissertation emanated from a multi-disciplinary research setting. U-CARE research studies follow scientific methods and standards in medical studies. An overview of what a study looks like in the U-CARE context is given in Appendix B.1; this may help a medical layperson to understand the research studies, empirical data, and quotes from the clinical researchers.

Table 2. Research studies using the U-CARE software system

Name	Description	Category	Inclusion duration	Participants[19]
U-CARE AdultCan	A randomised controlled trial to study the effect of a stepped care self-help programme via the internet on anxiety and depression among adults with cancer.	U-CARE	2013–2017	1,117
U-CARE Heart	A randomised controlled study of the effect of an internet-based psychological self-help programme on anxiety and depression in post-myocardial infarction (MI) adult patients. The study aims to examine the clinical efficacy and cost-effectiveness of internet-based cognitive behavioural therapy for post-MI patients.	U-CARE	2013–2016	1,052
U-CARE YoungCan	A feasibility study of an intervention including psychosocial support and psychological treatment via the internet to adolescents and young adults struck with cancer during adolescence.	U-CARE	2015–2016	6
U-CARE ParentsCan	A study focused on offering psychological help via the internet to parents of children with cancer. [ParentsCan has multiple phases. In the first phase, the study was designed. In the second phase, a sub-study (PUSSEL) was executed.]	U-CARE	2016–2017	121
U-CARE MAYA	A study to identify cancer-related emotional suffering in young people diagnosed with cancer during their teenage years and to develop psychological treatments to alleviate such suffering. [The study used the U-CARE software system only to collect research participants' consent.]	U-CARE	2015–2015	10
UPPS	An observational study, Uppsala Pelvic Pain Study, concerning pelvic pain arising during pregnancy.	Associated study	2014–2019	356
ISAK	A randomised controlled study of the effect of internet-based relapse prevention as an adjunct to anti-depressive medication.	Associated study	2014–2016	105
U-CARE Pregnant	A randomised controlled study of the effect of an internet-based intervention aimed at pregnant women with childbearing fears.	Associated study	2015–2016	270

[19] Participant counts updated on March 06, 2019.

JUNO	A randomised controlled study of the effect of an internet-based intervention for women with negative or traumatic experiences related to childbirth or abortion. This also involved the women's partners.	Associated study	2015–2018	401
AIDA-I & AIDA-II	A project aimed at experimentally investigating how people are reinforced in psychotherapy and how this affects adherence and drop-out. A further aim was to examine whether live and internet-based interventions differed in these aspects.	Associated study	2015–2016	185 and 100
ENGAGE (1000g)	An observational study aimed at getting an overall view of the physical and mental health in young adults born with an extremely low birth weight.	Associated study	2018–2019	170

A number of research studies are using, have used, or are planning to use the U-CARE software system (see Table 2 for a brief description). Although most of the studies focus on clinical psychology and particularly cognitive behavioural therapy (CBT) interventions, the target populations of patients (i.e., research participants) vary as regards cause of their need (e.g., cancer, myocardial infarction, pelvic pain, et cetera), age group, and gender. This shows how diverse the research studies' uses of U-CARE are (e.g., design and evaluation of CBT intervention using RCT, observational or experimental studies, or collection of online consent), yet all depend on the U-CARE software system to achieve their research goals.

From the perspective of this dissertation, the multiple research studies performed in the U-CARE context can be seen as multiple empirical contexts; also, U-CARE and the associated studies can be regarded as a variety of academic research contexts in eHealth. Regardless, the important thing here is that the U-CARE system has been developed to support various circumstances, and also been used in 'production mode' under various circumstances. The different cases have affected the study process and enriched it through the use by and feedback from a large number of stakeholders with differing views and empirical contexts. These include various stakeholders representing different research studies with diverging priorities, differing availabilities, and various expertise connected to clinical research, rather than to software engineering. Also, the table shows the total numbers of participants included in the research studies and inclusion duration. Inclusion duration refers to the length of time from addition of the first participant to addition of the last participant. This is a small part of the overall research study execution period, as it does not include study design and analysis duration. However, it is important to see that different studies have different durations and were started at different dates. Different starting dates and different durations are important for two reasons. Firstly, different starting dates means that different stakeholders' requirements came at different times and that the system evolved gradually. Studies were at different stages, exerting different pressures on the software development team concerning software development, deployment, and operation. This means there would be different consequences for stakeholders in case of any failure. Secondly, the durations show the length of time during which the software development team had to maintain and support consistent functionality. The object of study is thus "richer" than a single-use situation.

3.3 The U-CARE Software System – The Artefact

The U-CARE software system is referred to in different ways, for example, the U-CARE platform, the U-CARE Portal, the U-CARE software, the U-

CARE system, and the U-CARE infrastructure[20]. The U-CARE software system is the main artefact[21] discussed in this dissertation. This system is designed to enable delivery of psychological treatment interventions and data collection in all research projects undertaken within the framework of the research programme.

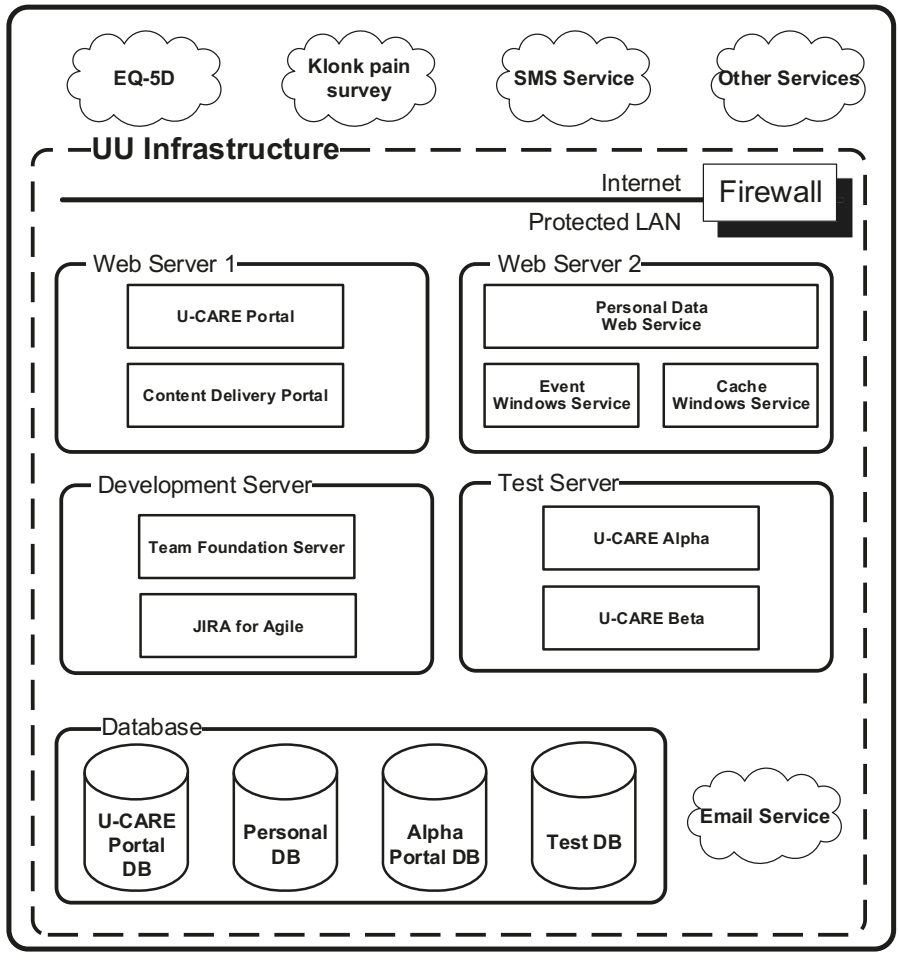


Figure 6. The U-CARE software system.

[20] Hence, any mention of 'U-CARE' with 'platform', 'portal', 'software, 'system', and 'infrastructure' in the text or empirical material of this dissertation refers to the U-CARE software system.

[21] In this dissertation, the U-CARE software system refers to an 'ensemble artefact' embedded/immersed in the U-CARE practical/social setting, not the software system or technology per se.

Figure 6, above, shows a visualisation of the U-CARE software system and its components. The U-CARE software system consists of various subsystems, such as i) the U-CARE portal (web-based software), ii) a personal data web service, iii) an event windows service, iv) a cache windows service, v) an email service, and vi) external services (SMS[22], EQ-5D[23], Klonk pain survey[24], et cetera). Additionally, there are a few desktop applications for data export, verification, analysis, and reporting. The Uppsala University[25] IT infrastructure is used for email, database, and web servers. The U-CARE software system uses four web servers. First, the production server hosts the U-CARE portal and the content delivery portal. Second, the personal server hosts the personal data web service and event and cache windows services. Third, the test server hosts the alpha and beta portals for testing purposes. Fourth, a development server hosts Git[26] to support source code version control and the team foundation server (TFS)[27] for automatic system builds and testing. Many database servers are used for data storage for different web services, for example, the U-CARE portal, personal web service, alpha, beta, and test portals. Access to the personal database is restricted, to ensure the security and privacy of research participants' personal data.

The U-CARE software system comprises ~200k lines of code and 100+ database tables. Several design patterns from object-oriented design have been applied, such as the factory design pattern (e.g., Gamma *et al.*, 1995). Interfaces[28] are used to achieve a weak coupling between classes (e.g., Parnas, 1972; Parnas, 1976; Canning *et al.*, 1989), and to facilitate unit testing[29] (Sjöström *et al.*, 2011). Software development using interfaces reduces dependency on implementation specifics and improves the code reusability (e.g., Gamma *et al.*, 1995). The U-CARE portal, the central web-based software, has a layered architecture which includes the foundation, data access, communication, business logic, and presentation layers. The presentation layer, following the Model-View-Controller (MVC) design pattern, has three further layers. The U-CARE software system supports various activities and functionalities regarding research and treatment delivery. Table 3, below, explains core activities and functionalities related to different user roles.

---

[22] It is offered by http://www.pixie.se/ [accessed: October 9, 2014].
[23] EQ-5D is a standardised questionnaire for use as a measure of health outcome. The questionnaire is copyright-protected, and requires a licence, meaning it can only can be used through a dedicated web service. http://www.euroqol.org/ [accessed: October 9, 2014].
[24] http://drawsurvey.com/ [accessed: October 9, 2014].
[25] http://www.uu.se [accessed: October 9, 2014].
[26] https://git-scm.com/ [accessed: January 16, 2017].
[27] https://www.visualstudio.com/tfs/ [accessed: October 9, 2014].
[28] Interfaces in object-oriented programming languages are used to define and abstract types that contains no data or code.
[29] Interfaces facilitate unit testing as they are used to create a mock (dummy) implementation (mocking frameworks are available to do this automatically) to run the tests, without having to use concrete (real) implementation.

Table 3. Activities and functionalities in the U-CARE software system according to user roles

User roles	Activities and functionalities
[Clinical] Researcher	Create and design research studies (according to protocol)
	Customises study-specific features (e.g., allow the user to access chat, forum, internal/instant messages (IM) and library)
	Manage research studies
	Schedule events (e.g., reminders)
	Design questionnaires (using generic questionnaire design tool)
	Design treatment content for psycho-education (i.e., audio, video, PDF, text, images, et cetera) using the library
	Design intervention treatment (e.g., ICBT modules which include treatment contents, questionnaires and homework)
	Manage research participants' consent
	Add a research participant to the research study and send login information
	Use decision support (i.e., dashboard which provides an overview of current state of activities)
	Create FAQs
	Choose staff to be shown on About us page
Therapist [i.e., psychologist]	Design questionnaires (using generic questionnaire design tool)
	Design treatment content for psycho-education (i.e., audio, video, PDF, text, images, et cetera) using the library
	Design intervention treatment (e.g., ICBT modules which include treatment contents, questionnaires and homework)
	Follow treatments in accordance with study protocol
	Approve ICBT modules
	Respond to homework tasks
	Communicate with research participants (e.g., using IM)
	Use decision support (using patient indicators framework)
	Moderate forum and chat
	Define flag words (i.e. suicide) that when uses in chat or forum will alert moderators that they need to pay attention to a particular conversation
	Answer and monitor FAQ
[Research] Participant	*Provide consent online
	*Fill in questionnaires
	*Choose nickname
	*Upload picture [if they want to upload]
	+Go through treatment by completing a list of ICBT modules
	+Access self-help and homework
	+Communicate with therapists (e.g., using IM)
	+Communicate with peers through chat and forum
	+ Choose to be visible or not in chat and forum
	+Write personal diary
	+Ask questions to health care professionals
	[*any participant in reference, control or treatment group] [+functions for treatment group only]
Health care professional	Get patient approval to participate in research or be contacted

	Add research participants (at various health care sites across Sweden)
	Design treatment content for psycho-education (i.e., audio, video, PDF, text, images, et cetera) using the library
	Moderate forum and chat
	Answer and monitor FAQ
Registrator	Add research participants
	Fill in participant-specific questionnaires, often regarding clinical data
	[a Registrator may be assigned to a specific health care site and can only register data for participants from their site]
U-CARE support	Add support issues (received through phone calls or support emails)
	View research participant and activity snapshots
	Reset research participant flow
[Research] Coordinator	Coordinate research groups and studies
[also, product owner]	Monitor research study events (reminder, role change, ICBT offer, et cetera)
	Audit privacy breaches
Any user (except research participants)	Translate text to research study-specific language (using in-place translation feature)

The above description of the U-CARE software system shows its complexity. The U-CARE software system is continuously evolving, expanding, and being extended, primarily due to new feature requirements from existing and associated studies. In addition, the system provides functionality such as (double) authentication (using dynamic authorisation management), system log for errors and exceptions, log user events and interactions (i.e., referred as respondent behavioural logging – RBL), event management (rule-based engine), data security and privacy, security/privacy breach monitoring, risk detection regarding suicidal patients, send reminders (messages that are scheduled within a specified timeframe after a specific event), and research participant stratification and randomisation.

3.4 Design Science Research at U-CARE

Information Systems researchers, being part of the U-CARE multi-disciplinary research environment, have ingrained the design process with knowledge from the Information Systems field and its sibling disciplines, for example, software engineering including design patterns (e.g., Gamma *et al.*, 1995), interaction design (e.g., Preece *et al.*, 2002), and internet-based psychosocial care and cognitive behavioural therapy (e.g., Kraft *et al.*, 2009). The Information Systems input is based on a stakeholder-centric (Sjöström & Goldkuhl, 2010; Sjöström, 2010) and iterative approach, promoting a focus on value creation, combined with ADR (Sein *et al.*, 2011) – a practice-inspired DSR approach – which allows for ideas and design principles to be gradually refined.

The multi-disciplinary approach in U-CARE resonates well with the ideals of rigorous evaluation in Information Systems design research, as put forth by Hevner *et al.* (2004). Contributions from psychology and economics, disciplines with a strong quantitative evaluation tradition, have ingrained the design research with evaluation methods. The relevance, design, and rigour cycles (Hevner, 2007) were used in the iterative development of the U-CARE software system.

The overarching ambition among Information Systems researchers, in the U-CARE context, was to employ a DSR approach to develop novel design knowledge drawing on their experiences in designing the U-CARE software system (ISR-1, 2010, IT meetings minutes). Over time, with an increased understanding of the problem domain, new knowledge interests emerged, for instance in the design process, relating design challenges that were not expected by U-CARE researchers at the inception of the research process. Such knowledge interests concern, among other things, decision support for therapists (Sjöström & Alfonsson, 2012), crowd translation (Sjöström & Hermelin, 2013), design principles for data export (Mustafa & Sjöström, 2013), online survey evaluation techniques (Sjöström, Rahman, *et al.*, 2013), technological-ecological adaptability (Mustafa *et al.*, 2014), design process exploration (Sjöström, 2017), privacy and accountability (Sjöström & Ågerfalk, 2013; Sjöström, Ågerfalk, *et al.*, 2014; Sjöström *et al.*, 2017), and software-embedded evaluation (Sjöström *et al.*, 2018). Several design features were built and evaluated, for example, a generic questionnaire design tool, an intervention design tool, dynamic actions, an authentication, authorisation and menu tool, in-place translation, and respondent behaviour logging (RBL). It could be noted that the U-CARE context is unique, as it enables for the study of a design process both as the creation and enactment of design research, as well as the appropriation of the design activities as conducted by Information Systems practitioners.

3.5 U-CARE Design Process

The U-CARE development team, consisting of Information Systems researchers and software developers, used Scrum, an interactive and incremental agile software development framework (Beck *et al.*, 2001). Scrum uses three roles (Development team, Scrum Master, Product owner), three artefacts (Product backlog, Sprint backlog, Burndown chart), and time-boxed sprints. Each sprint consists of four events (Sprint planning, Daily Scrum, Sprint reviews, Sprint retrospectives). In U-CARE, development sprints last two weeks. At the end of each sprint, a sprint review meeting is held where *customers* – various stakeholders (mainly researchers from multiple disciplines) – are shown the latest version of the system. These meetings are followed by design meetings, in which stakeholders provided feedback to the development team, who

use the feedback in the continued development efforts. In total, 250+ design meetings have been organised thus far, engaging a wide variety of stakeholders, including representatives from the different academic disciplines, such as the clinical researchers, Information Systems researchers, economists, health care professionals, and psychologists. Also, external specialists and patient groups have been invited to explore the system. The stakeholder-centric (Sjöström & Goldkuhl, 2010) and iterative approach promotes a focus on value creation – a continuous assessment of the U-CARE software system as a means to contribute to the overarching goals of the U-CARE programme.

Table 4. U-CARE stakeholders in the design process and their relevance

Stakeholder	Relevance in U-CARE context	Relevance in this dissertation
Clinical researchers	The clinical researchers are conducting clinical trials to study various emotional and mental health problems that may arise due to physical illnesses. They are usually psychologists, nurses, and cognitive scientists.	They are the customers for whom the U-CARE software system was designed, as well as being users of the system.
Psychologists	A psychologist is a mental health professional who develops and delivers psychosocial interventions, for example, CBT.	Psychologists, who have the role of therapists, design CBT contents, manage these contents in the U-CARE software system, design CBT treatments, manage CBT treatments in the system, and interact with research participants.
Health economists	Health economists are studying and evaluating the cost of U-CARE interventions.	They provide feedback on the U-CARE software system design at various stages.
Associated researchers	The researchers from associated research groups run associated studies in the U-CARE context.	They provide diverse requirements for the U-CARE software system, influence the design choices, and evaluate it.
Health care professionals	Physicians, nurses, and hospital staff add research participants to the U-CARE software system. They also moderate discussion forums and answer FAQs.	They also give feedback on their task-related issues.
Research participants	People taking part in research studies and using the U-CARE software system. They are from various groups, for example patients, relatives of patients, or healthy persons.	They provide indirect feedback on the U-CARE software system.
Information systems researchers	The Information Systems researchers design the system and abstract novel design knowledge drawing on their experiences of designing online psychosocial care. Information Systems re-	Information Systems researchers are also part of the ADR team.

	searchers are part of the development team. They work alongside the software developers with the goal of shortening the development time and overall cost for the U-CARE software system.	
Software developers	They develop and maintain the U-CARE software system. They are part of an in-house development team which includes both software developers and Information Systems researchers.	They are also part of ADR team.

U-CARE Stakeholders

Since the empirical setting – the academic research context – and the stakeholders in the context are an integral part in designing eHealth research software, in the research method, and thus also in this dissertation, it is important to describe the stakeholders and their involvement in the performance of this research. This will promote the understanding of the empirical data and their interpretations.

Table 4 describes the list of U-CARE stakeholders, as found in the U-CARE software system and IT meeting minutes, and their relevance in the U-CARE context, as well as in this dissertation. A few groups of stakeholders consist of two or more stakeholder categories mentioned above, for example, development team, U-CARE management, U-CARE healthcare, U-CARE support, and external stakeholders. For the purpose of this dissertation, when working together, the Information Systems researchers and Information Systems practitioners (i.e., software developers) are denoted in the following as the ADR team.

The next section gives an example of the design process and stakeholders, and its relevance to U-CARE context, as well as to this dissertation. It is essential to understand the design process in order to understand later sections about data collection, research methods, and ADR cases. One of the critical points to note is the evolution of the design process, which has implications for data collection over time concerning data type, quantity, quality, and format.

An Example of the Stakeholder-centric Evolving Design Process

At a very early stage in the design process, it was realised that receiving and managing continuous feedback from a diverse and large number of stakeholders was a challenge. At this point, software feature requests were managed in text to-do lists using a Microsoft Word document and feedback was shared

through various communication channels, for example, face-to-face, via email, phone, or Skype[30].

Participant view

Hur ska första sidan för deltagarna se ut? Detta inbegriper flikarna, över- och vänstermenyer och informationstexter, autotexter och länkar på förstasidan. Hur ska denna sida fungera? Vad ska hända när man klickar på olika saker.

Who	What	Prio	Status
IS	"Home" should be placed to the very left and "Intervention" should be deleted.	HIGH	
IS	At the top: Participants name is mentioned twice. That may be unnecessary. The Study description should be visible all time without click. The study description should be able to use "fancy text" e.g. bold and bullet lists.		Done
IS	The first box (Intervention description) beneath "My program" should be erased.		Done
IS	Discuss with psychologist should be moved to the left menu communication box and be renamed "Send message to therapist" SEE NEW COMMENT UNDER COMMUNICATION		Not yet
IS	"Activate new module" should only be visible if you can choose a new module		Done
IS	In "My program" all ongoing items (al items in the ongoing step/folder) should be listed in order with their alias-name but without description. It might be steps from two modules at a time. There should be different colors surrounding the items in each module.		Done
IS	Erase "Show whole module" and "show latest item"		Done
IS	When the participant logs in at the first time and if it is only one possible module available this should be opened automatically.	MEDIUM	Not yet
IS	The second tab next to "My program" should be named "Extra material". We would like to link a division or subsection from the library and perhaps threads from the forum to this place. In the "intervention three-view" it would be nice if you could "add link" and then choose from divisions in the library and threads in the forum. This will off course be lost if the Intervention is used in another study.	Low	Not yet
IS	The light bulb should be deleted (but at a late point after testing is finished).	LOW	Later
IS	The search doesn't work right now. Delete until it works. And then translate the alternatives	LOW	Not yet
IS	In the pop up calendar particular, after clicking the calendar icon, known dates should be marked. For example: times for observation points, when a module started, if there are any time limits elapsing and manually put reminders. The calendar has an English view with weeks starting with Sundays.	MEDIUM	Partially
	There should be a separator between each step in the program (step 2). The finished step or items should be obviously finished (other color or something).	HIGH	Not yet
IS	Participants should not be able to add reminders to others than themselves, and they should not be able to tag other users. (at least in WP1). If possible Therapists should be able to tag users.	HIGH	Not yet
Ps	Everybody can write help texts on the different pages. We should all help out with this.		
IS	In forum and Library there is an "Alternative box" to the left. This is only for the therapist. Delete. COMMENT 2012-01-10 In the library, the participant can choose to go the the library in the left menu under Alternatives. This is still an unnecessary box. Remove.	MEDIUM	Not yet

Figure 7. Text to-do list – product backlog and feedback.

Figure 7 is a screenshot of part of the document that described feedback/user stories related to the 'participant view' feature, user story priority, user story

[30] https://www.skype.com/ [accessed: October 9, 2014].

status, and the research group responsible (i.e., Information Systems). The feedback process was later improved by sharing the to-do list document using file synchronisation and a cloud storage service (i.e., Dropbox[31]). This enabled the use of a single central document for managing the information and feedback from the stakeholders.

Figure 8. The 'lightbulb' to get feedback from stakeholders.

In November 2011, a feedback feature was built into the system to allow any user (excluding research participants) to provide direct feedback about the system. Figure 8 (intentionally blurred) shows a screenshot from an arbitrary web page in the system. A click on the *lightbulb* icon opens a dialogue window, in which a free text comment can be written. This allows users to report problems and easily give feedback on the system design.

[31] https://www.dropbox.com [accessed: October 9, 2014].

A	B	C	D	E	F	G	H
Where	ID	What	Prio	Status	Doer?	Comments	Owner
automatic message	64	The automated message a participant recieves when an intervention is added is in english, needs to be translated	1	done	JS	(Dynamic message definitions). No IM is sent to participant... Should there be one?	
Questionnaire	71	Add a new catergory of question types when at "add question". It should be possible to choose numbered radio button. It should also be possible to choose the numbers on the buttons. I.e. if you	1	done	HR	Should not replace any existing, this should be added as a new category	HG
Alias namn bug	78	In homework it is possible to choose alias name to show for particiant. However it is not what is shown, it is the Item name. Also the general desciption should not be shown. See screen dump #78. The title should not be all upper	1	done	MI		HG
Participant ID	79	Look into the possibility to hav studyspecific IDs i.e. starting with a letter. So that numbering within a study is consecutive	1	done			HG
Bug	83	A link dissaper during translation, we should not make this possible to happen? Have a picture : # 83	1	done		Prio 1 on the link that dissaper	MH

Figure 9. Spreadsheet to-do list – product backlog and feedback.

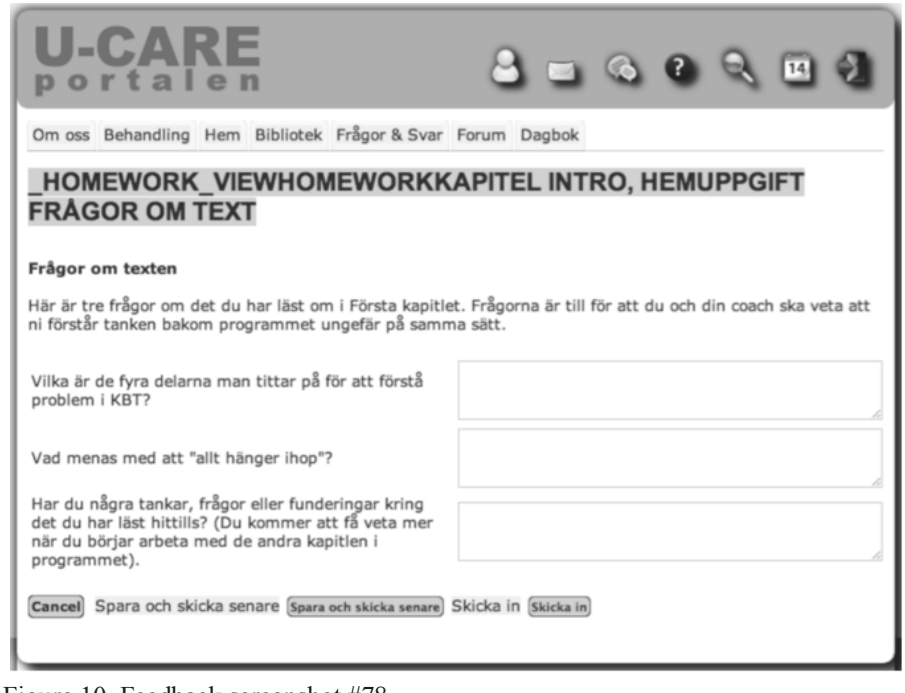


Figure 10. Feedback screenshot #78.

In August 2012, the text to-do list was moved to a Microsoft Excel spreadsheet. This was more efficient, as it made sorting and filtering easier. Additional information was included in the form of columns like Where (the fea-

ture), ID (user story identification number), What (user story text), Prio (priority like 1, 2, 3), Status (done), Doer (responsible developer), Comments (comments on or related to the user story), and Owner (stakeholder who initiated or was concerned with the user story). The design process was also improved through the addition of a screenshot folder, where stakeholders could add a screenshot (that showed and helped explain requirements, changes, or problems in the system). The screenshots could be connected to the user story by tagging them #ID, where ID was the user story identification number. For example, Issue #78 in Figure 9 and the screenshot related to Issue #78 in Figure 10 provide the information about a bug in the item name in the heading of the web page.

Table 5. Feedback categories and description

Category	**Description**
Aesthetics	The aesthetics (colours and layout) do not appeal to me!
Bug	I found an error!
Affordance	It is unclear how (or if) I can do something!
Consistency	The design of this page is not consistent with other pages.
Context	I do not understand what this page is about.
Feedback	It is hard to understand what just happened!
Ideas	I have got a great idea!
Performance	Everything is too slow!
Relevance and clarity	Information is missing, too extensive, or unclear!
Reply	This is a response to a previous comment.
Social	It is unclear who submitted something, or who can access what I submitted!
Unspecified	I just want to express something.

In March 2013, the feedback feature of the U-CARE software system was enhanced to act as a product backlog. The existing spreadsheet to-do list was imported into the database. The new routine was introduced for issue reporting, feedback, and feature management. Now, the clinical researchers could not only report bugs, they could also request new features and changes in existing features through the U-CARE software system. When the clinical researchers clicked on the lightbulb and enter comments, these were stored in the product backlog, along with a screenshot of the page they were on and contextual information derived from the logged-in user's context. They could also select a comment category. The categories reflected system usability needs and made feedback interpretation easier for the development team (see Table 5).

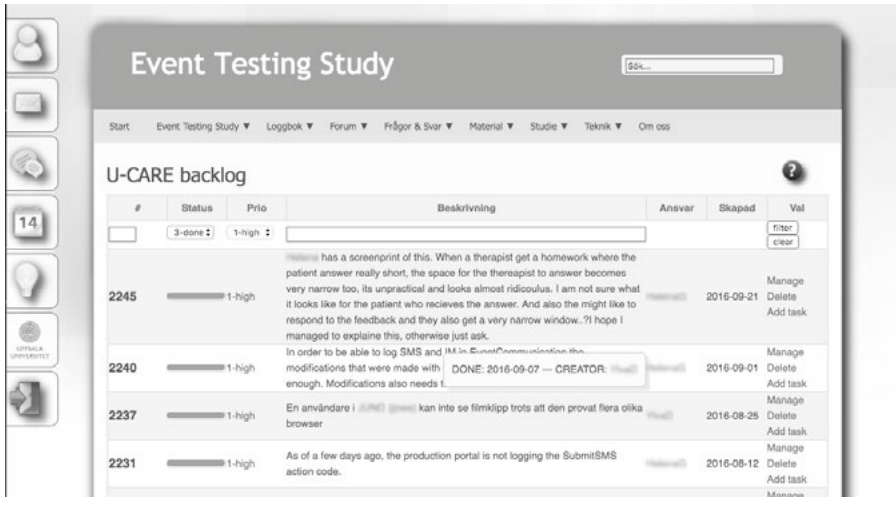


Figure 11. U-CARE product backlog and feedback feature.

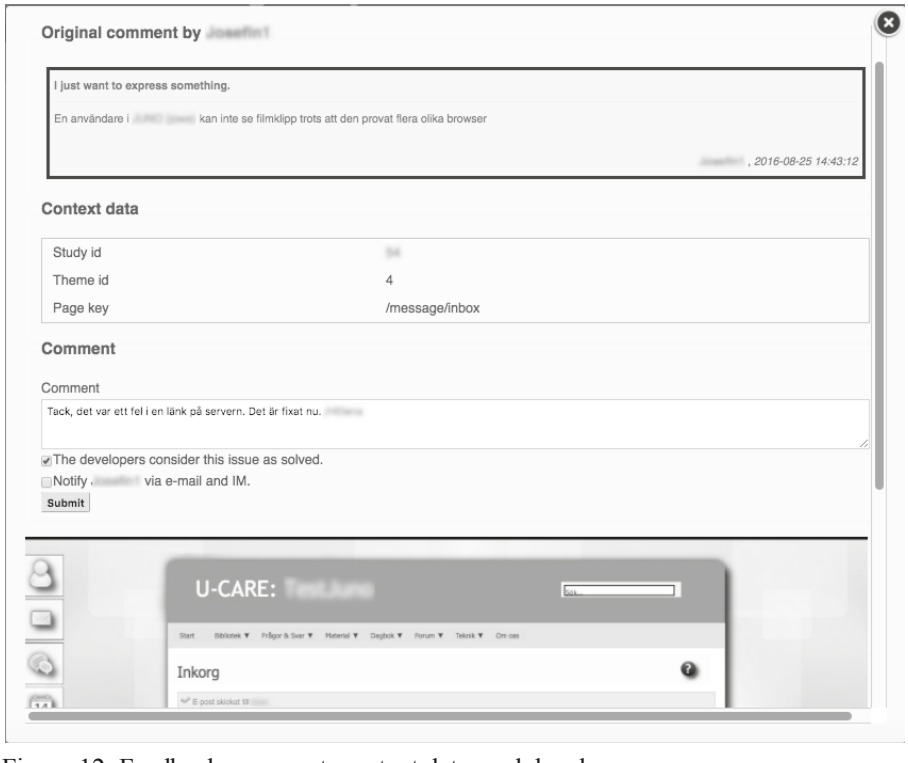


Figure 12. Feedback comment, context data, and developer response.

This new product backlog feature was further enhanced over time to include a full development cycle, for example, issue priority, sprint planning, splitting user stories into manageable tasks, and assigning complexity points to tasks. In a way, the product backlog feature became a software development tool for

an agile team, where the product owner could manage the product backlog (e.g., set user story priority) and the development team could manage issues (i.e., user stories in product backlog), sprints, tasks, and release info. Figure 11 is a screenshot of the product backlog including issue ID, status, priority, creation date, and – if resolved – who resolved the issue and when. Developers can directly comment and give updates directly in ongoing issues. The researchers receive an update on their reported issues either upon logging in to the system or by email. Figure 12 displays the feedback comment, screenshot, and use context of a clinical researcher, with developers' comment and issue status. As all stakeholders had access to the product backlog, they could see the status of development work, which enabled even further transparency. All other communication (e.g., research-related) was directed to the research coordinator (who was also the product owner). In this way, the development team did not receive information that was not related to or would not be handled by them.

Over time, the simple feedback feature became a rich design process management feature. The development history – a subpart of the product backlog feature – allowed addition of an annotation by the design researcher for a retrospective analysis of the process. The development history also integrated the product backlog and source code changes related to the backlog issue. This increased transparency enabled discoverability, from the issue creation through to the final changes in the source code. The backlog feature was a comprehensive repository, encompassing stakeholders' impressions of system qualities. This repository was one of the rich sources used in this dissertation to answer the research questions (this feature is no longer in use since it required to much attention to keep "fit").

Another Example of the Stakeholder-centric Evolving Design Process

Once the system was in the live mode, running real studies, developers had to provide support and system performance monitoring. This was done by creating a support role.

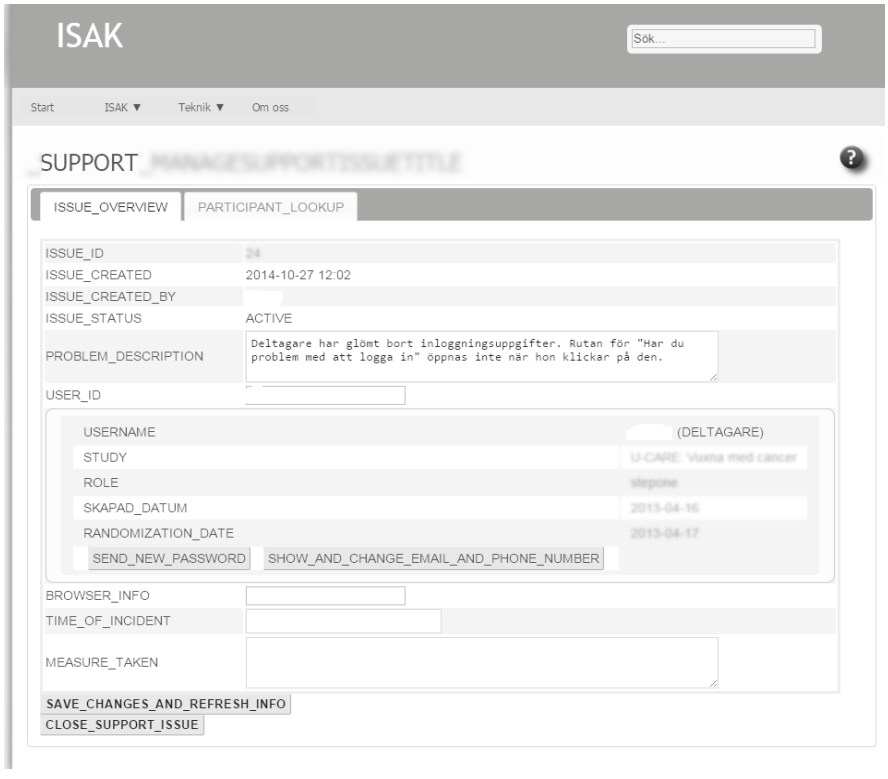


Figure 13. Support feature screenshot.

The support role served to handle immediate support issues from the U-CARE support staff (e.g., research assistants and psychologists) who were responsible for handling research participant issues, for example, related to system login, following intervention steps, and completing the questionnaires (see Figure 13). Over time, a support feature was introduced which provided the U-CARE support staff with the information necessary to handle support issue themselves in so far as possible. This also resulted in a rich repository of research participant issues and feedback on the system. However, it is important to note that the developers received some feedback from research participants (solely) indirectly through clinical researchers, support staff, and system logs. Similarly, due to privacy concerns and the requirement for separate approval from the ethical review board, the research participant issues and their feedback-related data have not been accessed or used directly in this dissertation. My role in the U-CARE context is explained in Section 4.3.

3.6 eHealth Challenges and U-CARE Research Context

The U-CARE research encompasses some chronic diseases and clinical issues that may cause symptoms of depression and anxiety; thus, the research results may have an impact on a large part of society. Multiple studies are running at the same time and are at various stages. The research participant enrolment methods differ between studies, for example, in clinics by routine health care professionals, through advertisements, and Sweden's national patient registers. There are various external stakeholders, for example, health care and patient organisations, research funding and monitoring bodies, et cetera. The research group has international collaborations, but also competes with research groups in various countries (e.g., see Appendix B.2). The group is unique in that it supports therapist-guided internet-based cognitive behavioural therapy.

Current openness trends mean that researchers have more open, transparent, and accountable research processes, for example, sharing research data for meta-analyses and results' validation, better utilisation of taxpayers' money, and the greater good of society. U-CARE promotes transparency of all activities to promote (open) research, while protecting research participants' data and privacy. Last, but not least, the potential issues related to patient safety, security and ethics are of great importance as compared with in other research projects due to the research participants' conditions. Thus, the open and accessible U-CARE context provides a unique opportunity for researching eHealth research software as it has the following properties:

- Multi-disciplinarity.
- Being at the intersection of health care, social care and self-care.
- Multiple target end-users, for example, patients, citizens, health care professional, researchers, and service recipients.
- Providing services to associated health care research groups.
- Acting as a health care provider and running a care ward.
- Engaging in providing education at various levels.

The U-CARE software system is developed, maintained, and managed by an in-house development team. The system is the central point, and the U-CARE research programme functions around it. The development team has a very central role in this complex environment, as it has the crucial responsibility of keeping the system functioning. On the one hand, the development team has very diverse issues, like changing requirements, goals, laws and regulations, and technological landscape, while, on the other hand, it has limited resources (e.g., team members). Since 2013, the U-CARE software system has continuously been running (RCT) studies, meaning that new releases of the system are required to be stable, to prevent damage to ongoing health care and research activities. The Information Systems researchers use DSR to understand this problem domain and design innovative artefacts as solutions. Moreover,

when finding out about the requirements for eHealth research software (the essential part of the design journey), the U-CARE context exhibits a high degree of unknown unknowns, diverse stakeholders (several without previous software engineering expertise) that are engaged in many other academic activities, and long-lasting development based on public funding. Consequently, the U-CARE context gives a unique opportunity for studying how to guide researchers and practitioners in sustaining the usefulness of eHealth research software, as it has the complexity and constraints of an academic research context and, as shown in this chapter, fits well with the research objectives of this dissertation.

4 Research Design

In this chapter, the research design employed in this dissertation is presented in detail. First, in Section 4.1, the design science research methods used are presented. The research method ADR is presented in Section 4.2. Lastly, in Section 4.3, the appropriation of ADR is presented, including the timeline, the author's role(s), data collection, data representation, data interpretation and analysis, ethical considerations, method limitation, and ADR case selection and criteria.

4.1 Design Research Methods

Design research encourages researchers to design novel solutions and then systematically study them, to build up the scientific knowledge about new designs. Particularly: what works, what fails, and why? Without such knowledge, we will not be able to understand the large-scale systems of today (Hevner & Chatterjee, 2010). Design research helps us understand complex systems (i.e., research domains) by participating in their settings, rather than observing them. In other words, design researchers learn more about design practice by doing design, as explained by Baskerville *et al.* (2011). This view resonates with the pragmatic idea that inquiry into a situation leads to an in-depth understanding of it (Dewey, 1938), and that attempts to change a situation will disclose forces that prevent or support the attempted change. In this dissertation, the goal was to understand how to *sustain the usefulness of eHealth research software*. It was significant to understand the thoughts and actions of stakeholders while engaging with the artefact within its actual empirical context. Hence, research in this dissertation was conducted using ADR – a practice-inspired DSR approach (Sein *et al.*, 2011), combined with the evaluation ideals of Hevner *et al.* (2004), Venable *et al.* (2016) and Gill & Hevner (2013), in multiple ADR cases. First of all, this enabled me to gain a holistic view of the academic research context. Secondly, it was useful in capturing the emergent properties of an unstable empirical context, like U-CARE, that was always changing and evolving. Thirdly, this was suitable in an exploratory analysis, when the goal was to provide an answer to a question of *how*, which aims to explain a certain phenomenon, such as RQ2: "How do Information Systems researchers and practitioners approach the quality characteristics …". In the next section, ADR is briefly presented.

4.2 Action Design Research

ADR is an interpretive research method, based on the sociomaterial recognition that human practice emerges (Orlikowski, 2010; Leonardi, 2012). Sein *et al.* define ADR as:

> A method for generating prescriptive design knowledge through the building and evaluating ensemble IT artefacts in an organisational setting. It deals with two seemingly disparate challenges: 1) addressing a problem situation encountered in a specific organisational setting by intervening and evaluating; and 2) constructing and evaluating an IT artefact that addresses the class of problems typified by the encountered situation. (2011, p. 40)

In other words, Sein *et al.* (ibid.) propose that artefacts should be designed in the organisational setting in which they would be used, with extensive participation from key stakeholders, using a structured and predefined process and learning from the intervention while addressing a problem situation.

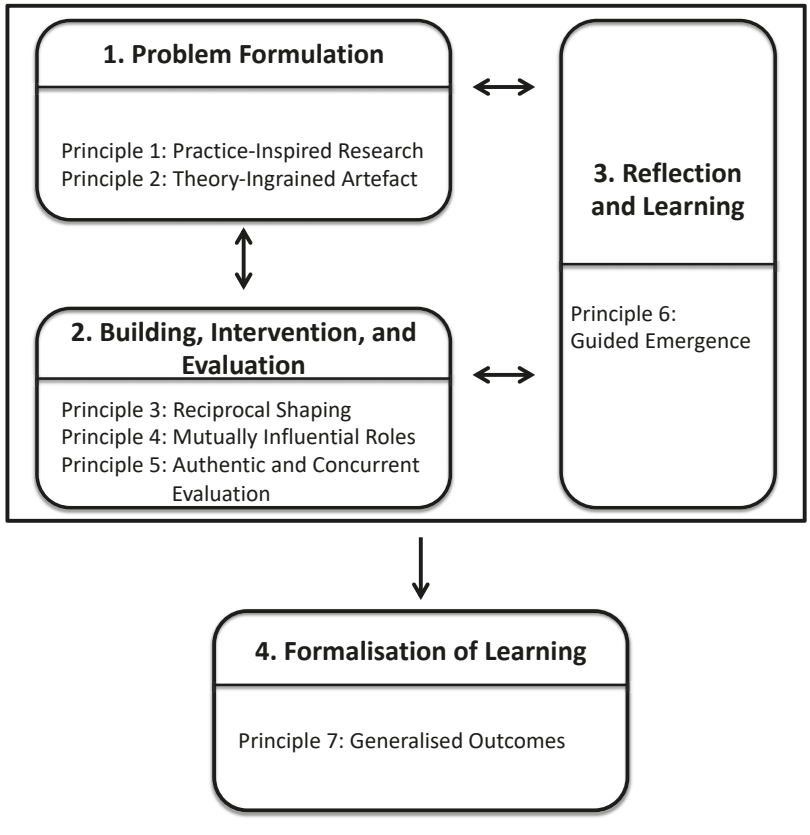


Figure 14. ADR method: Stages and principles (from Sein *et al.*, 2011).

Table 6. ADR stages and principles

Stages and principles	Description
1: Problem Formulation (PF)	In the first stage, the research questions are formulated based on problem perceived in practice, anticipated by researchers or practitioners, encountered in existing technologies, or identified in prior research. The problem is regarded as an instance of a class of problems for which the research aims to generate knowledge. This stage also includes determining the initial scope, and deciding on the roles and scope of practitioner participation. A critical issue at this stage is getting the long-term commitment of the organisation.
Principle 1: Practice-Inspired Research	Emphasises viewing field problems as knowledge-creation opportunities.
Principle 2: Theory-Ingrained Artefact	Emphasises that theories inform the artefact created and evaluated.
2: Building, Intervention, and Evaluation (BIE)	Problem framing and theoretical premises adopted in the first stage are used to carry out a change in the target organisation. During the BIE stage, the artefact is developed (B) and put into the organisational situation (I). As the artefact is used in the organisational context, it is continuously assessed and refined (E) to meet the needs of the end-users. BIE activities are simultaneous.
Principle 3: Reciprocal Shaping	Means that the artefact and the organisation shape one another.
Principle 4: Mutually Influential Roles	Means that practitioners and researchers influence one another.
Principle 5: Authentic and Concurrent Evaluation	Emphasises that evaluation needs to be continuous and organisationally situated.
3: Reflection and Learning (RL)	Deals with the experiences and insights from the BIE stage, with respect to the problem formulated in the first stage.
Principle 6: Guided emergence	Emphasises that the design of the artefact will emerge through its ongoing shaping by organisational use and by concurrent evaluation during repeated cycles of BIE.
4: Formalisation of Learning (FL)	The solutions are formalised as design principles to address the class of problems derived from learning during the organisationally situated intervention and artefact building.
Principle 7: Generalised Outcomes	Means that the situated learning in the organisational context should be further developed into general solutions for a class of similar problems.

The ADR research method consists of four different stages, i.e., Problem Formulation (PF), Building, Intervention, and Evaluation (BIE), Reflection and Learning (RL) and Formalisation of Learning (FL), incorporated with guiding principles (shown in Figure 14). Table 6 summarises activities within ADR stages and explains the guiding principles. The ADR method has been elaborated by scholars, for example, Haj-Bolouri *et al.* (2016) incorporated

knowledge from related approaches (such as participatory action research and participatory design), and Mullarkey & Hevner (2015; 2018) identified four distinct types of ADR cycles and expanded with multiple activities in each BIE cycle.

Haj-Bolouri *et al.* (2016) emphasised the importance of engaging stakeholders through a participatory approach in their participatory action design research (PADRE) method. They proposed four components (i.e., plan, implement, evaluate, reflect) centred around a learning nexus with activities to perform reflection and learning (RL) collaboratively with stakeholders in three stages of ADR (i.e., PF, BIE, and FL). They argued that the learning nexus would serves as a repository to be filled with accumulated knowledge from the continuous iterative cycle of activities in the RL stage of ADR.

Mullarkey & Hevner (2018) proposed an elaborated action design research (eADR[32]) process model identifying four distinct types of ADR cycles (i.e., diagnosis, design, implementation, and evolution). They expanded ADR with multiple activities (i.e., problem formulation, artefact creation, evaluation, reflection, and learning) in each BIE cycle and argued that the formalisation of learning could occur as a result of each stage. They proposed multiple entry points for conducting ADR at various levels of engagement (i.e., problem-centred, objective-centred, development-centred and observation-centred), combining the DSR process described by Peffers, Tuunanen, Rothenberger, and Chatterjee (2007) with ADR. Sein & Rossi (2019) disagreed with Mullarkey & Hevner (2018), arguing for ADR's single entry and inductive epistemology. Mullarkey & Hevner responded to this critique primarily based on empirical grounding and proof-of-use (Ågerfalk, 2019). This dialogue inspired my appropriation of the ADR method.

Nonetheless, neither ADR nor the two extended ADR methods focus on how to synthesise learning across multiple ADR cases to generate design knowledge through reflection and abstraction. To the best of my knowledge, most, if not all, DSR methods describe knowledge abstraction from a single case of artefact instantiation at a time. In the course of my research, while adopting stages and principles of ADR method, I have encountered multiple ADR intervention cases. Also, each case had a different priority, design pace, design duration, and design stage at any particular time. I found that the reflection and learnings of an individual case, in my research context, led to knowledge abstraction (i.e., design principles) for a class of problems; across multiple cases, the abstraction could generate design knowledge and thus apply to an even broader class of problems. Hence, the ADR method is supplemented with an additional stage of *augmented reflection and learning* to accumulate incremental prescriptive knowledge based on multiple ADR over an extended period (longitudinal) in multiple cases. In a similar vein,

[32] This is the latest version of eADR, which is an improved version of an early publication by Mullarkey & Hevner (2015).

Haj-Bolouri *et al.* (2017) suggested refinements to ADR, engaging in reflection and deriving outcomes.

The next section aims to give the reader information about the appropriation of the ADR method, and different orientations and choices made to conduct the research. While Sein *et al.* (2011) defined different stages of ADR, they did not go into details, leaving it up to the researchers who apply ADR to provide more details on how they use the method (Sein & Rossi, 2019). Therefore, the adaptation of the ADR method and how it has emerged in use in the U-CARE empirical context is explained. Lastly, a timeline of the research, the role of the researcher, data collection, data representation, data interpretation and analysis, ethical considerations, and method limitations are described.

4.3 Appropriation of ADR

The first stage in ADR is *problem formulation,* which emphasises the view of field problems as knowledge-creation opportunities. Problem formulation is the entry point to the BIE cycles in ADR (Sein & Rossi, 2019). In the U-CARE context, new requirements were continuously added during the development of the artefact (i.e., the U-CARE software system); either clinical researchers needed them or they were required by various other stakeholders. The need to modify the artefact to achieve the desired state provided ample opportunities for the ADR researcher to generate design knowledge by bridging this gap (Sein & Rossi, 2019). Hence, multiple ADR cases were initiated, each with its own specific problem. It was challenging to identify the class of problems when there were multi-disciplinary stakeholders in the U-CARE context and the problems evolved. Haj-Bolouri *et al.* (2017) have also pointed out that "new and interesting research problems continue to crop up as the [ADR] team engages in the research life cycle." Engaging the stakeholders in this stage was a challenge, due to time priorities and lack of knowledge of the ADR method in general.

The second stage in ADR is *building, intervention, and evaluation* (BIE), during which the artefact is (re-)built and put into the organisational situation. The artefact was assessed and refined continuously to meet the needs of the stakeholders during its use in the U-CARE context. Following the agile approach of continuously delivering artefact increments (see Conboy *et al.*, 2015) and demonstrating them in periodic IT meetings (a.k.a., sprint reviews) was useful in engaging and motivating the stakeholders. The engagement and motivation increased over time as stakeholders interacted with the early beta versions. It is evident that in the U-CARE context the ADR team followed the

Scandinavian approach to Participatory Design (Gregory *et al.*, 2003), as Information Systems researchers worked in close co-operation (co-designing[33]) with the software developers, clinical researchers and other stakeholders.

The BIE cycles in ADR are iterative loops where "the artefact and the designer's and user's understanding of the artefact evolve through a series of trials and their evaluation" (Sein & Rossi, 2019, p. 3). The activities during the iterative BIE cycles are described in detail to make it transparent how the artefact was built and evaluated in the practice, and the resulting changes that were made in the U-CARE context. The opportunity of continuous and longitudinal observation and actively participating in the artefact design process in the empirical settings has enabled me to communicate explicitly regarding how the artefact evolved through BIE cycles.

The third stage in ADR is *reflection and learning* (RL). Sein *et al.* (2011) framed RL as running in parallel with PF and BIE stages. Although Sein *et al.* (ibid.) postulate that reflection and learning occur continuously during ADR research, they conceptualise *formalisation of learning* (FL) as an activity of its own. In practice, formalisation of learning took place at the end of each BIE cycle (e.g., different versions of the set of design principles). Haj-Bolouri *et al.* (2016, p. 19) argued that "the ADR method can benefit from incorporating learning within and across each and every stage iteratively." Sein & Rossi (2019) have also suggested, while agreeing with Mullarkey & Hevner (2018) regarding publishing results from various phases of an ADR project, that new knowledge can be formalised and that generalised outcomes emerge throughout a project. An explicit and transparent formalisation of learning in the form of a set of design principles was achieved by providing details on how the design principles emerged (i.e., initial and multiple revisions). The continuous formalisation was useful in disseminating intermediate results through publications, such as Mustafa and Sjöström (2013) and Mustafa *et al.* (2014). This facilitated early feedback on research from the design researcher community. For example, subsequent revisions of the design principles were formulated using the effective and actionable structure proposed by Chandra *et al.* to make design principles clearer and more precise:

> Provide the system with [**material property –** in terms of form and function] in order for users to [**activity of user/group of users –** in terms of action], given that [**boundary conditions –** user group's characteristics or implementation settings] (2015, p. 4045)

[33] In this dissertation, 'co-design' is used in a broad sense to refer to actively involving all stakeholders in the design process to help ensure the result meets their needs and is usable. The designer (who may be the researcher or practitioner) takes on the role of a facilitator. The stakeholders in co-design are the people who are likely to be impacted by or will benefit from the process and the outcome, either directly or indirectly.

As argued above, in this dissertation, the formalisation of learning was continuous during the BIE cycles. Hence, the fourth stage in ADR, the *formalisation of learning* (FL), simply entails presentation of a summary of the learnings, and a final version of the design principles, as the incremental and continuous formalisation of learning is part of the preceding RL stage. It is important to note that while the ADR cases, stages, and BIE cycles are presented as being linear, there were, in practice, multiple iterations within and between different cases, stages, and cycles.

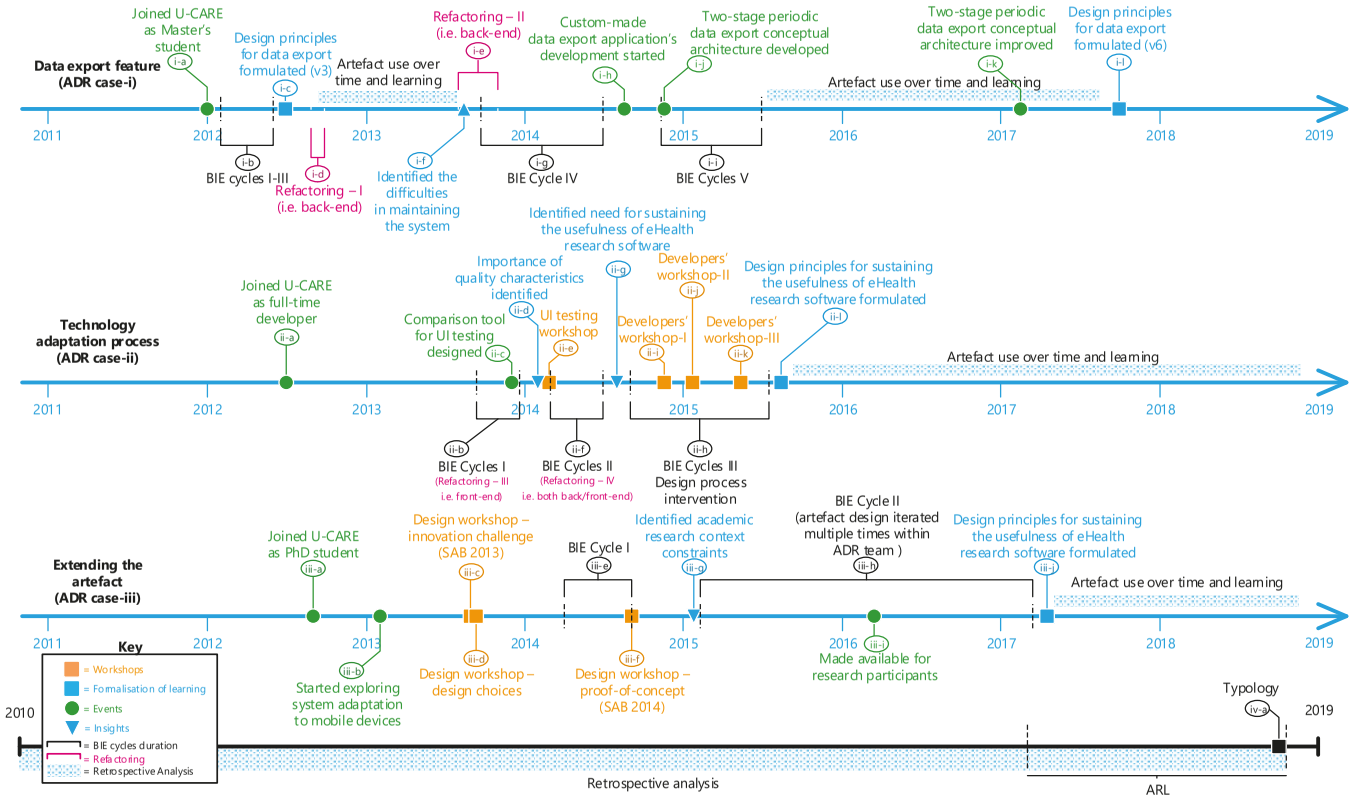


Figure 15. Timeline of the author's engagement and the ADR appropriation in the U-CARE context.

Timeline

The aim of the timeline is to provide an illustration of the research process, which consists of many overlapping research activities from the multi-year engagement in U-CARE. Figure 15 shows research activities in multiple ADR cases ordered in a timeline with the main activities keyed: events, insights, workshops, formalisation of learnings, and BIE cycles' duration. In narrating the story of the ADR cases, this figure is referred to, along with specific keys. Figure 15 also enlarged and provided at the (folded) back cover of this dissertation to make it easier to read.

The Author's Role(s)

In this section, my rationale and engagement in the empirical context and the gradual evolution of my research interests related to sustaining the usefulness of eHealth research software in the academic research context are presented. Likewise, my engagement as an ADR researcher and how I actively participated (and intervened), together with Information Systems practitioners and various stakeholders at all levels in the U-CARE context, are discussed.

The practical relevance of the U-CARE research programme and its impact on society[34], described above, inspired me to join U-CARE. ADR focuses on practical problems with theoretical relevance. The opportunity for continuous and longitudinal observation, while taking part in the software development process in the empirical settings (being deeply immersed in the organisation myself), enabled me to understand problems better, intervene with the design artefact, and reflect on the design processes. The ADR approach promotes participation and mutual learning through the iterative design cycles, involving both the researcher and practitioners. The hands-on experience of designing artefacts in the context facilitated an in-depth understanding of the context and design problems. I firmly believe that my value as an ADR researcher lies not only in the artefact I designed or the research I published, but also in what I learned while designing and publishing. This has had an influence on my writing style in describing my experiences and letting the readers know why things are the way they are; helping them learn about the context and its intricacies, and eventually, enabling them to appropriate learning to their context(s). I argue that I, as a result of this, am able to make the research material accessible to the reader.

[34] The application of research results has an impact beyond academia and could have a real influence on society.

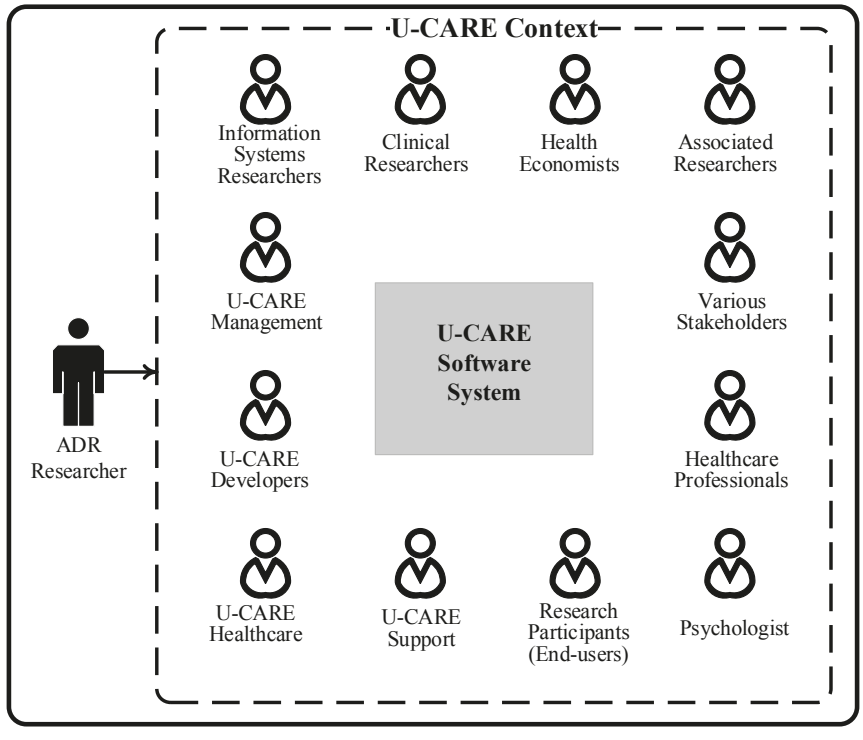


Figure 16. Different roles in the U-CARE context.

In the U-CARE context, I as an ADR researcher, in the ADR team, engaged in building, intervention, and evaluation of the U-CARE software system. Figure 16 provides an overview of different roles in the U-CARE context. My first engagement in U-CARE was as a Master's student in January 2012 (Figure 15, pt. i-a). I was involved in designing the data export feature for the U-CARE software system. At the time, I had more than ten years' experience in web development in both the Microsoft .Net platform and open source technologies, with an academic background in computer science, technology management, and Information Systems. Most of my professional experience was in the academic context. This enabled me to understand the complex U-CARE software system and design a generic data export feature for it (case i). During January to June 2012, when I was using the ADR method in my Master thesis, I consider my involvement in the context to be that of an ADR researcher (Figure 15, pt. i-b). For a brief time, from July to August 2012, I worked as a full-time developer on some features of the U-CARE software system (Figure 15, pt. ii-a). During the period January to August 2012, I took an active part, as member of the development team, in all workshops organised in U-CARE. My research interest in the Information Systems field led me to apply and subsequently enrol as a PhD student in the U-CARE research programme.

When I joined U-CARE as a PhD student in September 2012, the decision was made that PhD students would dedicate 10% of their time to the development of the U-CARE software system, primarily to conduct their research (Figure 15, pt. iii-a). My previous insider experience as a Master's student and software developer enabled me to understand and reflect on the problems in the U-CARE context. During the first year of my PhD studies, I was mainly focused on learning research methods in Information Systems, interdisciplinary research, and clinical research. During this time, I observed that the U-CARE software system faced problems on mobile devices, because the system was not built to accommodate them; that had been an early design decision made. However, there was a growing trend in Sweden of accessing the internet through mobile devices. Eventually, many users (i.e., research participants) of the U-CARE software system were using or wanted to use mobile devices to access it. This led to my initial research interest in the U-CARE software system's adaptation to mobile devices (Figure 15, pt. iii-b). In parallel, I was able to reflect on the actual use of the generic data export feature and proposed data export design principles in the U-CARE context. I learned that there were a number of difficulties in maintaining the U-CARE software system (Figure 15, pt. i-f). I was specifically interested in sustaining the data export feature and adoption of data export design principles over time. I found out that designing innovative artefacts was interesting, but that looking at our design artefact's actual use and appropriation by the developers in routine design was much more interesting. In September 2013, the outcome of the mobile adaptation[35] exploration resulted in an understanding of the need for updating technologies used in the U-CARE software system or software development process (Figure 15, pt. ii-b). The technology upgrade and subsequent improvement in the design process became another interesting research case (case ii). In 2015, an actual adaptation of the U-CARE software system began (case iii).

Being an ADR researcher, I was guiding the emergence of design artefact and design process. Based on learnings from the technology upgrade (case II) and prototype development (case III, BIE cycle I), I took a step back and let the development team take initiatives and make design decisions. While facilitating stakeholders through the design process, I leveraged my expertise of design and research and helped the stakeholders to develop their ideas.

Inspired by the ADR approach, with continuous reflection and learning during build-intervention-evaluate cycles across different cases, I began to consider a broader class of problems. In the middle phase of my work with U-CARE, my research interest developed around the need for sustaining the usefulness of eHealth research software in the academic research context (Figure

[35] *Mobile adaptation, adaptation to mobile devices and extending the artefact* are used interchangeably to refer to the same third ADR case, i.e., adapting the U-CARE software system for better user experience on mobile devices such as mobile phones and tablets.

15, pt. ii-d and ii-g). This increase in my understanding of the research problem was not a linear process and included multiple inputs from the stakeholders in the described context. The evolved research interest (i.e., sustaining the usefulness of eHealth research software in an academic research context) is summarised in the introduction chapter of this dissertation. Agile software development facilitated the BIE cycles and continuous engagement with stakeholders. The first-hand and intimate knowledge of the empirical context and data, and participation in dedicated design workshops enabled reflection and learning. I used a personal research log to write on-the-spot reflections and take field notes.

I collected the empirical data for research while developing the U-CARE software system and observed the stakeholders in action in the context. The intention was to gain in-depth understanding of how researchers and practitioners approached sustaining the usefulness of eHealth research software throughout the lifecycle of the U-CARE software system and how they addressed issues that arose over time. The various roles of the author facilitated the collection of in-depth views from stakeholders, who participated in or were observed during the research in this dissertation. In accordance with the iterative nature of ADR, the collaboration within the development team not only resulted in knowledge acquisition, but also gave the team members the opportunity to reflect. Also, working together with a team, I was able to continuously observe and write down on-the-spot reflections about the artefact and the design process. Thus, a critical aspect of the research was achieved: access to and understanding of the problem domain. The U-CARE context provided me with opportunities to come close to the object of study and to gain access to what was happening. This resulted in in-depth knowledge and insights that would have been very difficult for a non-participating researcher to gain. Still, it should be acknowledged that the empirical context is too rich to be comprehensively captured.

Data Collection

The research process was non-linear, longitudinal in nature, and iterative, and therefore demanded different data collection methods, instruments, and tools. Thus, to seek answers to the research questions, I collected and analysed primary data via interviews, direct participatory observations, workshop sessions, seminars, brainstorming sessions, focus group and field notes. Additionally, U-CARE software system documentation, IT meeting minutes, product and sprint backlogs, and code repository comments were collected (from the existing data repositories) and analysed. Also, e-mail correspondence, brainstorming diagrams, and developers' self-notes have been collected.

The audio-recorded semi-structured interviews were transcribed and formatted as pdf documents. Field notes were formatted as a pdf document. The product backlog was exported from the database to a Microsoft Excel file and

formatted as a pdf document. The source code repository's commit history was exported and formatted as pdf a document. IT meeting minutes, developer notes and emails, and U-CARE documentations, such as SAB reports, data extraction guidelines, et cetera were formatted as pdf documents. All collected empirical data (2010–2019) were then imported into Atlas.ti[36] for coding and to perform a qualitative content analysis.

In this dissertation, I report findings based on an analysis of multiple sources of evidence. This reflects positively on the overall quality and validity of qualitative data and also enhances the trustworthiness in the research approach used. Data collection was an iterative process based on unfolding events, the author's maturity as a researcher, and an evolving understanding of the U-CARE context. It was also based on my understanding of the research process and the maturity of the research design of this dissertation. For example, the first case, (Ch. 5) *the data export feature*, has less empirical material than the third case, (Ch. 7) *extending the artefact*, which was more fully documented. This is in part due to how the software development process evolved and changes in the stakeholders' interest over time. The software development experience enabled me to extract and understand heterogeneous data.

Table 7. Data collected during the research, extent or quantity, and duration

Type of source	Description	Amount/occurrence	Period
Participatory observations	In-office, co-located with software developers, participation/observation in stand-ups, planning and retrospective meetings, research meetings, presentations, workshops, discussions, and informal chats	Five years/2–3 days a week	January 2012 to January 2017
Interviews	Audio-recorded semi-structured interviews with software developers and researchers	12 (275 minutes)	June 2012 to January 2017
Workshops/seminars	Audio-recorded workshops and seminars	13 (1,652 minutes)	February 2014 to October 2015
Field notes	Recording of day-to-day events, participatory observations and personal reflections	1 (175 pages)	June 2013 to January 2017
IT meeting minutes	Contain information about the bi-weekly progress of software development, stakeholders' feedback and design decisions	172 doc. (every two weeks)	November 2010 to December 2018
Product backlog and sprint log	Product features, their prioritisation and completion. [Used in the analysis of the U-CARE software system design process]	1,873 entries	November 2010 to February 2016
[Technology] Log	Log technologies in end-users' environment. Used to design and evaluate during mobile adaptation	20,000 entries (when a user logged in)	June 2013 to November 2017

[36] http://atlasti.com/ [accessed: September 29, 2015].

Exception log	Errors, their occurrences, and resolution. [Technical debt extraction]	16,695 entries	May 2012 to November 2017
Code repository	U-CARE code repository (through SVN version control), used to review code revisions history and developers' annotations for code commits. [A type of self-reflection done by the development team]	5,072 entries (almost every day)	February 2011 to March 2017

Table 7 contains a comprehensive description of data collected from the U-CARE context. The periods reflected in the above table represent the first and last date of data collection for each respective data source.

Data Presentation

In this dissertation, due to the small numbers of stakeholders, I do not report frequencies, relying instead on real quotes from the stakeholders and interpretations. The representation of these quotes and diverse empirical data needs to include information to facilitate readers' understanding and interpretation. The empirical data is in the form of: 1) stakeholder codes, based on their roles (e.g., clinical researcher) in the U-CARE context, combined with a number (e.g., 1,2,3) assigned as a unique identifier; 2) timeline (e.g., year, period); and 3) source (e.g., interview, focus group, observation, IT meeting minutes, workshop, seminar, repository, backlog, and so forth). Here is an example of a quote:

> it is quite impressive, it is good, I think it is more customisable and user-friendly than I thought it would be. I thought it would be a lot difficult; that we are would go running to [specific] developer every time we needed some data. (CR-5, 2012, Interview)

Any text in the quotes enclosed in square brackets ([]) is additional information added by the author for completing the sentences, to increase readability, to clarify, or to give context. With the stakeholders' consent, I do not use the stakeholders' names; the stakeholder codes are used instead. In some cases, when the quotes may be of a sensitive nature for the stakeholder, the unique number in the stakeholder code is removed to preserve privacy; for example, Abc-1 would be written as Abc-#.

Table 8. List of stakeholder codes

Stakeholder role	**Code**	**Profile**
Clinical researcher	CR-{1-9}	The clinical researcher role includes the programme director, psychologists, oncology nurses, PhD students and post-doctoral researchers. They had additional roles, for example, one clinical researcher was the principal investigator. Another clinical researcher was the study coordinator, product owner, and later become a team leader as

		well (Jan 2015 onward). Some clinical researchers also acted as therapists in some studies. Some clinical researchers joined the project as PhD students and later become post-doctoral researchers.
Information Systems researcher	ISR-{1-4}	Information Systems researchers include PhD students and Master's thesis students participating as members of the software development team. One of the Information Systems researchers acted as team leader, technical lead and Scrum Master as well (until Dec 2014). Information Systems researchers also switched from part-time to the full-time developers and vice versa.
Software developer	Dev-{1-7}	Developers worked part-time or full-time. They also had different additional roles over time, for example, one of the developers become a technical lead (Jan 2015 onward), another acted as Scrum Master (from August 2014 onward). One developer joined as a Master's thesis student and later become a full-time developer.
Research assistant	RA-{1,2}	This role includes research assistants, support staff, and project coordinators.

Table 8 shows a list of the stakeholder codes used in this research with comments/reflections about the stakeholders' profiles within each role.

Data Interpretation and Analysis

The interpretive research perspective is used to facilitate the process of understanding and to produce profound insights into the studied phenomenon (Klein & Myers, 1999). Interpretive research is a well-established part of the Information Systems field (Walsham, 2006). As outlined by Orlikowski & Baroudi (1991), interpretive research aims at clarifying phenomena by attempting to enable understanding of the meaning that participants assigned to them, which fits the focus of this research well. In the data interpretation and analysis, an interpretive approach was used that was inspired by Klein & Myers' (1999) principles for conducting and evaluating interpretive field research (see Table 9).

Table 9. Klein & Myers' principles for interpretive field research

Principle and description*
1. The fundamental principle of the hermeneutic circle "This principle suggests that all human understanding is achieved by iterating between considering the interdependent meaning of parts and the whole that they form. This principle of human understanding is fundamental to all the other principles."
2. The principle of contextualisation "Requires critical reflection of the social and historical background of the research setting, so that the intended audience can see how the current situation under investigation emerged."
3. The principle of interaction between the researchers and the subjects

"Requires critical reflection on how the research materials (or 'data') were socially constructed through the interaction between the researchers and participants."

4. The principle of abstraction and generalisation

"Requires relating the idiographic details revealed by the data interpretation through the application of principles one and two to theoretical, general concepts that describe the nature of human understanding and social action."

5. The principle of dialogical reasoning

"Requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings ('the story which the data tell') with subsequent cycles of revision."

6. The principle of multiple interpretations

"Requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the same sequence of events under study. Similar to multiple witness accounts even if all tell it as they saw it."

7. The principle of suspicion

"Requires sensitivity to possible 'biases' and systematic 'distortions' in the narratives collected from the participants."

* The description of principles is presented as-is from Klein & Myers (1999).

Being an insider and involved action design researcher, it was important to consider my subjectivity regarding the collection and analysis of data (Walsham, 1995). There is a need to mitigate the *mediation of language and preconception* associated with understanding reality, to paraphrase Orlikowski & Baroudi (1991). A triangulation approach was employed to analyse data, motivated by the author's subjectivity and bias. The triangulation approach is beneficial in addressing the principle of interaction between the researchers and their subjects (Klein & Myers, 1999). I initially approached the process of interpretation independently. Then, I discussed my interpretation with an independent Information Systems researcher, who was not directly involved in the U-CARE context, to validate the interpretation against empirical data.

As a final step, I sent the outcome of the analysis (in the form of a draft version of this dissertation) to the key stakeholders for *member checking* (Creswell & Miller, 2000). Acquiring data from multiple sources led to data triangulation. There was various software used in the qualitative content analysis, for example, Atlas.ti, SVNStat[37], Microsoft Excel, MS SQL Server Management Studio, and mobile-usage[38]. In addition, the CoDisclose[39] (Sjöström, Eriksson, *et al.*, 2013) tool was used for retrospective analysis.

[37] http://svnstat.sourceforge.net/ [accessed: August 18, 2015].
[38] https://github.com/hgoebl/mobile-usage [accessed: February 24, 2016].
[39] Renamed and improved as DeProX – A Design Process Exploration Tool (Sjöström 2017).

Ethical Considerations

Myers & Venable (2014) proposed an ethical principle for DSR such as i) the public interest, ii) informed consent, iii) privacy, iv) honesty and accuracy, v) property, and vi) quality of the artefact. The privacy of stakeholders who contributed in this research (e.g., software developers, clinical researchers, and focus group participants) has been protected by de-identifying their responses, interview transcripts, and other sources (e.g., documents, source code, et cetera). Some figures have been intentionally blurred to protect the privacy of stakeholders. The software development team was observed directly, for which reason its members were informed of and consented to participate in the research via a written consent form. The research design has been adapted to maximise privacy and eliminate unnecessary risks (e.g., the views of software developers were not directly revealed to other stakeholders). Data were not revealed directly to stakeholders working in the U-CARE setting; instead, the interpretation of the data was presented. If the stakeholders explicitly consented to them being referred to in the empirical material, they might not have revealed their real views. On the other hand, removing stakeholder-related information might lower the transparency of the research. The original data was triangulated through another researcher's interpretation who was not directly involved in the U-CARE context. Still, it is possible that the stakeholders might disagree with the interpretations made.

Method Limitations

Clearly the *real world* is more complex than what can be captured by a researcher. Despite the research in this dissertation being conducted in a rigorous and longitudinal study, the contextual nature of the design poses a challenge for how the research methodology can be operationalised in practice (Haj-Bolouri *et al.*, 2017). The current design research project was performed in a specific application context, and the resulting designs and design research contributions are influenced by the opportunities and constraints of the application domain. For example, it was neither possible nor would it have been ethically correct to take part in all workshops or access all documents and data. My interpretation of the empirical context can only paint a limited picture. Furthermore, interpretation and contextualisation are inherently subjective. Moreover, despite all efforts for rigorous research design, the stakeholders' views may have evolved, as this was a longitudinal study of an ongoing research project. I consider this a challenge for ADR researchers.

ADR Case Selection Rationale

Multiple ADR cases were used to increase the in-depth understanding and learnings, as well as the analytical generalisability of the research results. The

iterative character of ADR, the multiple designs and the use contexts (U-CARE and associated studies) can be seen as comparable to more than ten different empirical contexts. Also, the use of multiple ADR cases and several revisions of the artefact design and development are in line with the idea of abstracting this work from one case to another.

The primary objectives of this dissertation were to highlight the practices of sustaining the usefulness of eHealth research software in an academic research context from each case and across all cases.

> A paradoxical, but perhaps realistic, view of design goals is that their function is to motivate activity which in turn will generate new goals […] Each step of implementation created a new situation; and the new situation provided a starting point for fresh design activity. (Simon, 1996, p. 162–163)

This description of Simon (1996) aligns well with the U-CARE context and in some sense also with the research design of this dissertation as well. As the research progressed through the different ADR cases, the research aims, questions, and objectives evolved and were adapted to new insights. The research benefited from feedback obtained continuously from U-CARE stakeholders. The research design was continuously refined through engagement and interaction with the academic research context.

Stakeholders

As stated above, stakeholders are those who are impacted by (or have an impact on) the project and their perspectives need to be considered for a project to be successful. From a systems perspective, stakeholders are individuals or groups of individuals who stand to gain or lose from the success or failure of a system. eHealth research software, in the academic research context, has multiple and varied stakeholders, including researchers (who need the research software and whose research depends on it), developers (who design and maintain the software), research participants (who use the software during the research), decision-makers (i.e., implementers, funding agencies, ethical approval boards, government agencies, research community, and – not least – citizens), and end-users.

Stakeholders are likely to have different viewpoints of the system and different criteria for success (Venable *et al.*, 2016). Stakeholders assign different weights (or priorities or levels of importance) to different characteristics, depending on their subjective judgment and knowledge (ISO/IEC Standard 25010: 2011, p. 5). Consequently, the relevance of the quality characteristics is also subject to the stakeholders' goals and objectives for the eHealth research software (Cho *et al.*, 2012). At the same time, quality characteristics are critical factors in ensuring value to stakeholders and can be further used to determine requirements and their criteria for satisfaction.

Selected cases

As stated above, the cases were selected to increase the in-depth understanding and learnings, as well as analytical generalisability of the research results. Hence, three cases were selected to presenting different key stakeholder's perspectives, different objects of study (feature, design process, design product), different states of maturity of the U-CARE context (forming, maturing, matured), and to include instances of both success and failure. All cases provided accessibility and closeness, availability of rich data, and the opportunity for involvement and participation.

Table 10. Selected cases

#	Case	Key stakeholder	Object of study	Maturity of context
1	Data export feature	Clinical researchers	Feature	Forming
2	Technology adaptation process	Software developers	Design process	Maturing
3	Extending the artefact	Research participants	Design product	Matured

Each case addresses a slightly different aspect of the U-CARE software system and its academic research context. The key stakeholders of the ADR cases are mentioned in Table 10, based on the U-CARE empirical context. For example, the clinical researchers were key stakeholders in the data export feature case, as they needed to export the research data from the software for analysis. A data export feature is a must-have requirement in eHealth research software. The data export feature design started at the U-CARE forming stage. The software developers were key stakeholders in the technology adaptation process case, as they had to learn, implement, and maintain the technologies in the software system. The technological-ecological changes posed challenges for software developers and required attention in the design process through the U-CARE maturing stage. In the matured stage of U-CARE, several research studies were ongoing. The research participants varied in age and tech-savviness. They accessed the software system through different mobile devices, and they were key stakeholders impacted by the adaptation in the mobile devices case. Extending the artefact to mobile devices increased the U-CARE software system's reach. Some wanted to access and were accessing the U-CARE software system on mobile devices, and others are comfortable with the desktop product.

Table 11. Motivation and aim of three ADR cases

#	Case	Description
1	**Data export feature**	
	Motivation	Data export is a crucial functionality for eHealth research software in an academic research context. The clinical researchers need to export the data for analysis, to interpret research results, and draw conclusions. The data export feature design started at the U-CARE forming stage.

	Aim	The case aimed to develop design principles and quality characteristics for data export in eHealth research software in an academic research context.
2	**Technology adaptation process**	
	Motivation	Software developers have to cope with a continuously changing design landscape, due to changes in user requirements, the organisation and the environment, while designing eHealth research software in an academic research context with limited resources. The technological-ecological changes posed challenges for software developers and required attention in the design process through the U-CARE maturing stage.
	Aim	The case aimed to develop design principles and quality characteristics, to guide the design process, and to support a continuously changing design landscape in an academic research context.
3	**Extending the artefact**	
	Motivation	Technological innovations in the surrounding environment can affect the usefulness of eHealth research software. eHealth research software's access/availability is essential for the end-users (i.e., research participants in a research context). In the matured stage of U-CARE, several research studies were ongoing. Extending the artefact to mobile devices increased reach toward research participants.
	Aim	The case aimed to develop design principles and quality characteristics, to guide the design process, and to support a continuously changing design landscape in an academic research context.

Each ADR case was used to investigate the quality characteristics of eHealth research software, how researchers and practitioners approached these quality characteristics, and design principles for sustaining research software usefulness. The case results have served as lessons learned. The ADR cases were neither sequential nor independent of one another; rather, they progressed simultaneously, albeit at different stages. The motivation and aims of each case are presented in Table 11. See Chapters 5–7 for a detailed account of the cases.

Part III: Design in Three Cases

5 Case I: The Data Export Feature – the U-CARE Formation Phase

This chapter describes the design and evaluation of the data export feature, a crucial functionality for eHealth research software in an academic research context. The functionality is highly stakeholder-centric, as it is used for exporting research data from the U-CARE software system. This case highlights quality characteristics that impact on sustaining the usefulness of the eHealth research software studied and the need for parallel and continuous adaptation of the eHealth research software throughout its lifecycle. The initial BIE cycles (i–iii, presented in Section 5.2) helped the author to understand and engage further in the empirical context. The evolution of the data export feature served as a springboard to exploring the design process for sustaining the usefulness of the artefact over time during the subsequent BIE cycles (iv–v, presented in Section 5.2). This ADR case resulted in proposing a set of design principles expressing key aspects that needed to be addressed when designing a data export feature in an eHealth research software in an academic research context. This was the first ADR case, and it has continued throughout the research period. Section 5.1 explains the design context and problem relevance. The iterative building, intervention and evaluation cycles are presented in Section 5.2. Formalisation of learning is presented in Section 5.3 as a (final) set of design principles which emerged during the iterative BIE cycles

5.1 Problem Formulation

Information technology allows for large-scale data collection and data analysis, for example, data collection through online surveys (Lumsden & Morgan, 2005) and data analysis of user behaviour logs (Sjöström, Rahman, *et al.*, 2013). While the issue of data access, sharing, reuse, and reproducibility is extremely important in an academic research context (Murray-Rust, 2008; Peters *et al.*, 2012; Dallmeier-Tiessen *et al.*, 2014; Hettrick, 2016), previous research has not sufficiently emphasised the design of data export for research purposes. Data extraction, data migration, data mining, and data analysis are well-explored topics in Information Systems and related disciplines. In most cases, researchers only mentioned the availability of a data export feature, without detailing design issues related to such feature. There is an increased

interest among researchers, funding agencies, and policymakers to make data open (i.e., accessible by peers) for validation or further analysis (Arzberger *et al.*, 2004; Peters *et al.*, 2012; Ross *et al.*, 2012; EU, 2016), to support innovation (Nature, 2008) as well as to create additional value for the scientific community (Murray-Rust, 2008). Researchers must be empowered to export data flexibly from eHealth research software to make data accessible.

The U-CARE clinical researchers' primary objective was to develop and test eHealth interventions. They achieved this objective by designing RCTs, running RCTs, collecting data, and providing psychosocial care using the U-CARE software system. At the end of RCTs, they needed to export data from the U-CARE software system for analysis, to interpret and draw conclusions from the RCT results regarding whether an intervention or treatment had any effect. Based on its significance, data export was one of the required features listed early on in the U-CARE software system product backlog. U-CARE stakeholders also emphasised that the availability of a research data export feature was important:

> it [data export] is extremely important as we have so much data a lot of data which is unusual for us. We [Clinical researchers] often have quite a few data, but now on this platform [the U-CARE software system] with the internet and so on [… we get] large amounts of data. And it has to have a smart and well-functioning way to export data and to be able to use it, so I think [data export] is very important. (CR-5, 2012, Interview)

> it is [necessary] because you need some material to do your analysis […] so you can calculate the effect of the treatment. (CR-3, 2012, Interview)

> it is an important part of the research process. (CR-6, 2012, Interview)

Information Systems researchers, in U-CARE, also initiated the development of a data export feature. During the initial investigation, the U-CARE software system was studied in detail to identify challenges in designing a data export feature. The U-CARE software system at the time was designed with data collection in mind. While designing the data export feature, the development team anticipated and handled many challenges, such as a non-normalised database and continuously changing data models[40]. However, not all problems were foreseen during the initial problem formulation stage. Additional problems and challenges were identified through iterative BIE cycles. The U-CARE software system was a large, generic, and flexible system based on a dynamic web application. Mostly, features were designed using a flexible database structure to store data and metadata regarding the U-CARE RCT studies. This provided a generic and flexible application, but the extraction of data

[40] Data models are fundamental entities to introduce abstraction in a DBMS. Data models define how data are connected to each other and how they are processed and stored inside the system.

was a challenge. The data export process took time and was fraught with errors or defects. Continuous changes in the requirements from the U-CARE stakeholders forced changes in the system design and the database schema was another challenge in designing the data export feature.

The Information Systems researchers and practitioners of U-CARE could export data directly from the database using SQL queries or the built-in data export functionality of the database management system (DBMS), while the clinical researchers in U-CARE had little or no knowledge of SQL and DBMS. The Information Systems researchers and practitioners required an easy and efficient solution to export data with less effort in a suitable format compatible with data analysis tools. Thus, the fundamental challenge was to find an easy, effective, and efficient way to export data that satisfied all stakeholders in U-CARE. A data-centric clinical research software, such as the U-CARE software system, holds data from many research studies or clinical trials. Typically, the database is designed to manage data related to research studies. For that reason, the data collected by eHealth research software needed to be interoperable between different applications.

The design problem mentioned above about the U-CARE software system can be taken as an instance of a class of problems, (i.e., *designing a data export feature*) faced during data export in any eHealth research software. Following the ADR method, the ADR team wanted to consider the design principles of the data export feature that would apply to a class of similar problems. Therefore, the initial case-specific research questions were: RQ1) *What principles should guide data export design?* and RQ2) *What are the implications of RQ1 for software design?*

5.2 Building, Intervention and Evaluation Cycles

The data export feature went through multiple BIE cycles, i.e., i–iii (*generic*), and iv–v (*one-click*). The ‘generic’ and ‘one-click’ denote different functionalities that were used to build the data export feature in the U-CARE software system.

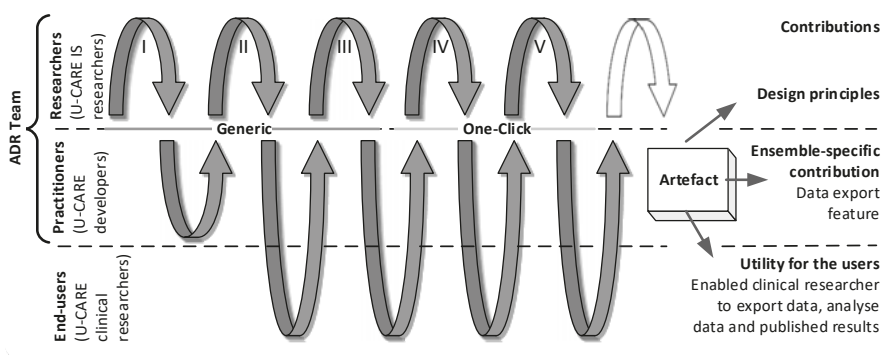


Figure 17. The BIE cycles of the data export feature including contributions and stakeholders involved in the design.

Figure 17 shows the BIE cycles, in which the data export feature was built in the U-CARE software system, put into the organisational situation, and formatively evaluated to meet the stakeholders' needs. The ADR team consisted of the Information Systems researchers and Information Systems practitioners (i.e., software developers) in the U-CARE context.

Figure 18. The timeline of the BIE cycles of the data export feature.

Figure 18 shows the timeline of the BIE cycles. Each BIE cycle is detailed in the following subsections: the build activities, intervention activities, and evaluation activities are presented. At the end of each cycle, an account is provided of lessons learned. I built most of the *generic* data export feature during the BIE cycles i–iii (Figure 15, pt. i-b). The design and evaluation of the BIE cycles i–iii have previously been published in Mustafa & Sjöström (2013), here they are summarised for the subsequent BIE cycles (i.e., iv and v), to accommodate the readers (Figure 15, pt. i-g and i-i, respectively). The *one-click* data export feature was built jointly by the ADR team.

BIE Cycle I

Build

The development of the generic data export feature's first version was profoundly influenced by the characteristics of the U-CARE software system, and the challenges it implied. As described in Chapter 3, the U-CARE software system is a relatively large and complex system, built to collect a variety

of research data, for example, research study events (inclusion date, randomisation date, CBT intervention version, and assignment date, et cetera), questionnaire responses from different observation points, treatment consumption (i.e., treatment modules, steps, items, and homework), research participants' activities (number of logins, library items visits, forum visits, FAQ visits, et cetera), communication (i.e., chat, forum, IMs), et cetera. Therefore, the decision was made that the first version would only support the export of questionnaire responses (a small subset of the research data).

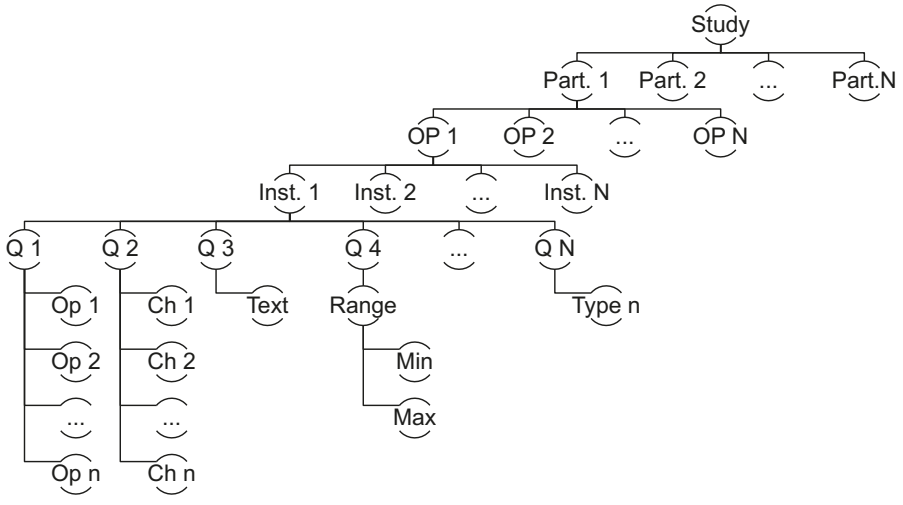


Figure 19. The data hierarchy in the research study questionnaires.

There was a hierarchical structure to the data. Figure 19 visualises this hierarchical structure. Each research participant [Part.] goes through several observation points [OP], which in turn includes several questionnaires (a.k.a., research instruments) [Inst.], consisting of multiple questions [Q]. Each question has multiple options [Op], choices [Ch], Text, Range with minimum values [Min] and maximum values [Max], et cetera. The clinical researchers were used to analysing data using cross-tabulations, i.e., getting all the data about a research participant in one row, where the answers to questions would be put in columns. This forced the design to include a pivoting functionality (row-to-column transformation), which entailed converting the hierarchical structure to a linear (flat) structure. It is important to note that different research participants can be at different observation points at any given time and may respond to different questionnaires or skip questions in any questionnaire. This had implications on data export, meaning that it had to align/map different research participants' data to an exact, unique sequence of columns.

Several features of the U-CARE software system were designed for malleability and offered configuration possibilities for U-CARE stakeholders, for example, software developers and clinical researchers. It made sense that the

data export feature should also be configurable by the clinical researchers themselves. The main idea was that the software as a whole should be useful in a new context as-is, without further software development involved. The data export feature was loosely coupled to the existing software, to achieve this generic property, through the use of data objects (DO), so as to minimise dependencies between the export feature and other parts of the software. The reflection utility was based on the reflection design pattern (see Appendix C.1). It inspects assemblies, types, and members of the U-CARE data models, and creates instances of them, and populates them with data. The reflection and data objects enabled easy adaptation of the data export feature to the changing U-CARE data models.

The reflection design pattern is used in other U-CARE features as well, for example, the authorisation feature (also referred to as the *action framework* in various publications by U-CARE Information Systems researchers) (Sjöström *et al.* (2011), Sjöström and Ågerfalk (2013) and Sjöström *et al.* (2017)). The generic data export feature's design is partially based on the authorisation feature[41]. The authorisation feature is briefly described in Appendix C.2.

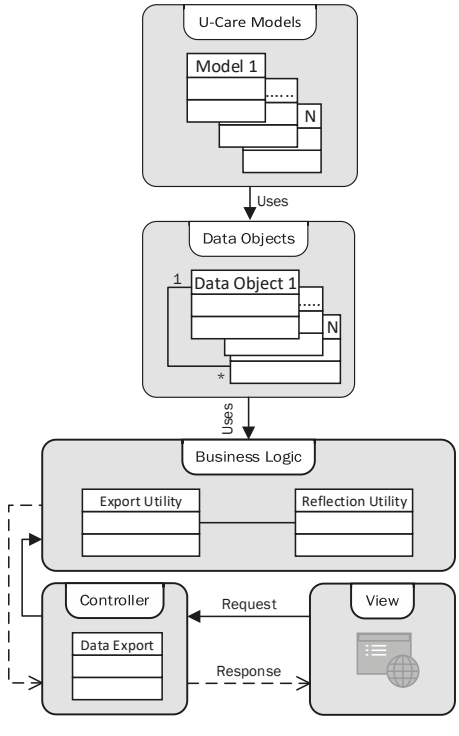


Figure 20. Architecture of the generic data export feature.

[41] The authorisation feature is referred to multiple times in the dissertation, for example, when data access is discussed.

Figure 20 illustrates the conceptual architecture for the generic data export feature that emerged during the design process. The export utility acted in its own layer, which facilitated data export into different formats, as well as persistent storage of user-defined export templates.

Intervention and evaluation

The intervention and formative evaluation sessions were performed as a part of weekly IT meetings between the clinical researchers and the development team. As stated by Venable *et al.* (2012), formative evaluation is such "in which an artefact still under development is evaluated to determine areas for improvement and refinement" (p. 426). The evaluation process showed that the clinical researchers appreciated the flexibility of the data export feature, which allowed them to extract data without needing the help of software developers. The software developers suggested a series of performance improvements including that a) user selections could be saved, and that the same selection of fields or small variation should be exported efficiently with fewer clicks during subsequent data export; and b) metadata should be generated whenever the data objects were changed.

Reflection and learning

In this BIE cycle, the clinical researchers were at an early stage of maturity as regards research study design. The research study design was directly related to data export, as a change in the study design can change the hierarchical structure of data or how research participants might respond to questionnaires and questions. There were multiple changes in study design requested by clinical researchers. As a result, the developers had to change the data models of the system. It should be noted that the reflection utility of the data export feature performed satisfactorily, as both the data objects and the user interface (UI) automatically adapted as expected whenever the developers modified the U-CARE software system's models. This highlights the empirical enactment of malleability as a key quality characteristic in the U-CARE software system, as this automatic adaptation enabled developers to focus on core features of the system without being concerned with dependencies in the data export functionality. Not everything in the data export feature could be adapted automatically. The addition of new models required minor adaptations of the data object classes. The first draft of the design principles was formulated based on the learnings during this cycle (see Table 12).

Table 12. Design principles for data export in eHealth research software (version 1)

Design principle	Specification
The principle of simplicity	Data export feature should be easy to use.
The principle of developer independency	Data export feature should be designed for end-users (researchers) and should not require deep technical knowledge.

The principle of mutability (emergence)	Data export should adapt itself to new data export requirements or changes in the software that hosts data.
The principle of compatibility	Data export should render output in standardised or de facto formats (such as CSV, XML, or spreadsheet), to facilitate import into data analysis applications and statistical applications.
The principle of easy development	The API for data export should be built in a way that minimises the development resources required to add new export functionality. Developers should be able to develop core system features without continuously addressing data export issues.
The principle of export as a separate concern	Data export should not require development for every data export request. The data export software should be prepared for present and future data export requirements and adapt automatically to changes in the underlying data model.

BIE Cycle II

Build

The lessons learned from the first BIE cycle were the basis for improving the data export feature and implementing additional functionalities. Reflection is a costly operation, performance-wise, and especially if the database is hit each time an object or collection of objects is retrieved from the production code. The reflection utility was revised to improve the performance of the data export feature, so that it ran only when a change was detected in the underlying U-CARE software system's data model. This was addressed by storing a persistent structure of metadata in the database and reusing it in subsequent data export requests. New functionality was added to manage templates (i.e., data export packages), so as to improve the efficiency for clinical researchers, who could create a template for data export and could then export data by reusing saved templates. The second version of the data export feature was integrated with the U-CARE software system and released on the U-CARE production server.

Intervention and evaluation

Initially, access to the data export feature was given only to the software developers. At that time, the U-CARE software system had a substantive amount of test data available (provided by various stakeholders during beta-testing throughout the design process). The test data were used to evaluate the data export feature in a production environment. Furthermore, a scenario-based evaluation (Hevner *et al.*, 2004) was conducted at an IT meeting where U-CARE management, clinical researchers and other stakeholders were present. The second version of the data export feature was explained and demonstrated using a test study with test data. The overall interpretation of evaluation was that all stakeholders appreciated the data export feature. However, questions were raised about the data export feature's access authorisations.

Reflection and learning

The key lesson learned from the evaluation was that the data export module should have access restrictions regarding who was allowed to export data, which data, and when. The anonymity of U-CARE research participants needed to be preserved. Another lesson learned during the cycle was the need for user-controlled row filtering, as in the existing data export feature only columns could be filtered. Based on these lessons, the design principles were revised. Table 13 lists design principles and any changes from the previous version, with additions highlighted in bold.

Table 13. Design principles for data export in eHealth research software (version 2)

Design principle	Specification
The principle of simplicity	Data export feature should be easy to use. **Preferably, data export should be triggered by a single click, and the users should be guided to filter data based on their needs.**
The principle of developer independency	Data export feature should be designed for end-users (researchers) and should not require deep technical knowledge. **End-users should be empowered to design their own export templates based on access to a view of exportable items.**
The principle of mutability (emergence)	Data export should adapt itself **or be easily adaptable** to new data export requirements or changes in the software that hosts data.
The principle of compatibility	Data export should render output in standardised or de facto formats (such as CSV, XML, or spreadsheet), to facilitate import into data analysis applications and statistical applications.
The principle of easy development	The API for data export should be built in a way that minimises the development resources required to add new export functionality. Developers should be able to develop core system features without continuously addressing data export issues.
The principle of export as a separate concern	Data export should **require a minimum of** development for every data export request. **Insofar as possible,** the data export software should be prepared for present and future data export requirements and adapt automatically to changes in the underlying data model. **The data export layer should depend on the core features, but not the other way around.**
The principle of restricted data access	**The access to the data export feature should be restricted to those who are allowed to export data. Authentication and authorisation schemes should be applied so that only those who are permitted to retrieve data for analysis can access them. Time restrictions may also apply to data export (e.g., only permitted after the trial is closed).**
The principle of anonymity	**All data must be de-identified so that a re-identification by use of different sets of published data or by linking exported data with other publicly available data sources is impossible. The information needed to re-identify individuals should be separated from collected data insofar as possible.**

BIE Cycle III

Build

The data export feature's second version was further improved based on learning from BIE cycle ii. New functionalities were added regarding data access issues and row filtering. The U-CARE software system already had support for role-based authentication and authorisation. The data access issue was handled merely by configuring access to the data export feature in a production environment. The U-CARE software system was designed to store data related to the U-CARE research participants in two separate databases. One database kept the research participants' identifiable data, while the other database kept all other study-related data (e.g., answers to questionnaires). Access to both databases was required to identify research participants, while the data export feature only gave access to the study-related de-identified data. In this way, data de-identification was achieved for the data export feature. In summary, there were trivial changes made to the data export feature to satisfy the data access and privacy requirements. However, it was clear that in other contexts there might be a need to adapt the data export feature to manage data access and privacy issues.

Figure 21. Generic data export UI.

New row filtering options were designed to filter a number of records before export, for example a filter that allowed exporting data of specific observation point(s). Figure 21 shows a screenshot of the *generic* data export feature's UI, how it adapts dynamically according to the data object structure and allows granular-level control over data export by letting researchers select individual fields to export both metadata from questionnaires and data collected through questionnaires.

Intervention and evaluation

In this cycle, the data export feature was evaluated based on feedback collected in four semi-structured interviews with open-ended questions, followed by a walkthrough demonstration. Three respondents were clinical researchers, and the fourth respondent was a software developer with an interest in the technical and conceptual qualities of the data export feature. In each interview, the data export feature, its intended use, and its UI were presented. This was followed by a demonstration of how to use the different functionalities in the data export feature, using a test dataset. The software developer was asked questions focused on software design. In general, responses were positive:

> the data export [feature] is very good as it follows design patterns […] one of the major concerns regarding data export performance is when we have huge [amounts of] data accumulated […] the good thing about data objects is that when you need to export another type of data you only [need to] work on the export engine, and rest [of the data export feature] will work […] the data export can work independently [from other features] […] this very fantastic work actually […] the system is growing very fast and there is always a challenge due to the growing needs for features, so extensibility is important, which is good […] Different design aspects are well implemented. (ISR-3, 2012, Interview)

The clinical researchers were asked about both benefits and drawbacks, as well as usability-oriented issues (e.g., ease-of-use). The clinical researchers expressed generally positive views about *generic* data export features:

> I think some training is required to use the tree view […] there should be a Save as function for packages […] the system should provide a facility for choosing headers from selected fields when creating a data export package […] the CSV data export format looks OK after importing to the statistical tool […] I am looking forward to exploiting the possibilities of the data export. (CR-3, 2012, Interview)

However, the data export feature's UI now showed many fields, including unimportant ones (e.g., fields to manage the business logic, such as timestamps, status flags, et cetera). Due to this, the UI was perceived as complicated. Two respondents expressed this as follows:

> I think data export is not complicated when using the system, but I am interested in understanding the system [better] [… the] data export interface is absolutely ok and understandable […] I think it looks more complicated than actually it is, and the first impression is *Oh my God! This cannot work!* but once it runs you [begin] to understand how it works [you would] need documentation or help to understand what fields [you do not need to worry] about while exporting […] [overall] it is quite impressive, it is good, I think actually it is more customisable and user-friendly than I thought it would be. I thought it would be [a lot more difficult; that] we [would] go running to [specific developer] every time we needed some data. (CR-5, 2012, Interview)

> it is very good that we have these extra functions [pivoting] [… the data export feature] is straightforward […]. it seems very flexible and has a lot of options […] especially like the option to choose conversion of columns into rows […] it is very good […] I guess the interface is a bit complex, I mean if you are not used to working with this type of tree structure […] but with some practice, it can be done, when you know what these items [in tree structure] correspond to […] I guess there will be a learning curve, but it should be manageable for everyone […] I can extract data right now based on what is shown to me […] it is quite straightforward I think. (CR-6, 2012, Interview)

Another concern was that the feature exported the metadata of questionnaires along with the research participants' responses. This led to more columns than required by the clinical researchers for analysis. The researchers stated that they did not need metadata for analysis and that in a case where they did need it, they could access it through the U-CARE software system. One clinical researcher stressed the potential for improvement of the data export feature by providing the possibility of labelling columns and highlighted the importance of a minimal number of clicks for exporting data.

The main output of the intervention and evaluation was that there should not be too much customisation and that the UI should not present irrelevant data. Still, most of the end-users were satisfied with this version of the data export feature.

Reflection and learning

The data export feature displayed all the fields/properties of the data models. Due to this, the UI became complicated and as a result the clinical researchers required training and information about all the fields while exporting data. The data export feature faced the design challenge that Sjöström *et al.* (2011) explained as "a trade-off situation between simplicity and mutability-in-use" due to its generic design. This cycle resulted in the addition of another design principle regarding UI (highlighted with bold letters in Table 14 below) (Figure 15, pt. i-c).

Table 14. Design principles for data export in eHealth research software (version 3)

Design principle	Specification
The principle of simplicity	Data export feature should be easy to use. Preferably, data export should be triggered by a single click, and the users should be guided to filter data based on their needs.
The principle of developer independency	Data export feature should be designed for end-users (researchers) and should not require deep technical knowledge. End-users should be empowered to design their own export templates based on access to a view of exportable items.
The principle of mutability (emergence)	Data export should adapt itself or be easily adaptable to new data export requirements or changes in the software that hosts data.
The principle of compatibility	Data export should render output in standardised or de facto formats (such as CSV, XML, or spreadsheet), to facilitate import into data analysis applications and statistical applications.
The principle of easy development	The API for data export should be built in a way that minimises the development resources required to add new export functionality. Developers should be able to develop core system features without continuously addressing data export issues.
The principle of export as a separate concern	Data export should require a minimum of development for every data export request. Insofar as possible, the data export software should be prepared for present and future data export requirements and adapt automatically to changes in the underlying data model. The data export layer should depend on the core features, but not the other way around.
The principle of restricted data access	The access to the data export feature should be restricted to those who are allowed to export data. Authentication and authorisation schemes should be applied so that only those who are permitted to retrieve data for analysis can access them. Time restrictions may also apply to data export (e.g., only permitted after the trial is closed).
The principle of anonymity	All data must be de-identified so that a re-identification by use of different sets of published data or by linking exported data with other publicly available data sources is impossible. The information needed to re-identify individuals should be separated from collected data insofar as possible.
The principle of adequate customisation	**Customisation should be afforded for end-users, and should not be too difficult. Only relevant information should be displayed. Customisation options should be presented in semantic layers to allow basic features to all, while allowing advanced users more advanced options.**

Artefact Use Over Time and Learning

The generic data export feature was evaluated using test studies with test data, and U-CARE stakeholders were satisfied with its functionality. Over time, the stakeholders experienced various problems while exporting data from real (pilot) research studies. For example, empty responses from research participants

were not handled properly. They resulted in misaligned row data from corresponding column headers. The development team also faced problems in maintaining the data export feature. Generic features increased the complexity of the software (Sjöström *et al.*, 2011). Although the *generic* data export feature was supposed to function even if the U-CARE software system changed, the feature stopped working when a big refactoring was performed.

During the refactoring (Figure 15, pt. i-d), the source code of the U-CARE software system changed from being very feature-oriented to being more activity-oriented, as it was assumed this would "be a better long-term solution [… which] will also make testing easier when new functions are added [… and] will also make data export easier" (ISR-1, 2012, IT meeting minutes). However, during this process, the middle layer (i.e., data objects) integrating the data export feature and the U-CARE software system became incompatible with the U-CARE data models, because this layer was not refactored along with the rest of the system (ISR-4, Research log, 2014-12-03).

Maintaining the data export feature was also difficult as the original developer of the data export feature (the author of this dissertation) began working on other development tasks (i.e., mobile adaptation and technology upgrade). Although the number of developers increased in the development team, new members were not yet acquainted with the existing software and application domain. Knowledge of the *generic* data export design was not adequately transferred to all members of the development team. Another source of complexity was the use of composition, reflection, and recursion design patterns in the data export feature. In the U-CARE context, composition, reflection, and recursion made the feature design easier or perhaps more elegant in cases of design problems like tree traversal and whole-part hierarchies' data representation. However, in cases of failure, it was hard to follow the code during debugging. Specifically, this meant that the input data had different hierarchical structures.

The failure in maintaining the *generic* data export feature made one thing very evident: there was a need for continuous adaptation of the data export feature due to changes in data collection, business rules, and components of the U-CARE software system. The experiences related to the *generic* data export feature's design led the ADR team to consider a design that acknowledged the quality characteristic of simplicity as a key to enabling a maintainable solution over time.

BIE Cycle IV

The *generic* data export feature was designed in accordance with the clinical researchers' initial data export requirements. User needs, and the users' abilities to express their needs, were honed over time (Mustafa & Sjöström, 2013). This was the case with the data export feature requirements as well. The data export feature was re-designed based on more elaborated requirements and

lessons learned during BIE cycles i–iii. The re-designed and improved feature was called the *one-click* data export feature (Figure 15, pt. i-g).

Build

The initial requirements' elicitation started with a data export request from the AdultCan pilot study (2013, IT meeting minutes). The data export requirements were discussed during a design workshop dedicated to this topic (2013, design workshop). Initially, due to the failure of the *generic* data export feature, data were exported directly from the DBMS. However, when similar data export requests were received from other clinical researchers (e.g., JUNO, an associated research study, 2013, IT meeting minutes), the ADR team decided to build a *one-click* data export feature instead of providing data export directly from the DBMS. During the development process of this data export feature, the output, data filtering, and variables' labelling were discussed with clinical researchers from the JUNO study.

The modular design of the *generic* data export feature allowed the ADR team to use part of the existing source code (i.e., export utility) to develop a simple and improved feature. The simple data export design was achieved by simplifying a complex design problem to the most fundamental unit of the research data, the RCT questionnaire in U-CARE. Thus, the *one-click* data export feature was designed to export questionnaire data only. Consequently, errors could easily be rectified and the code could easily be debugged and improved by narrowing down data related to a particular questionnaire.

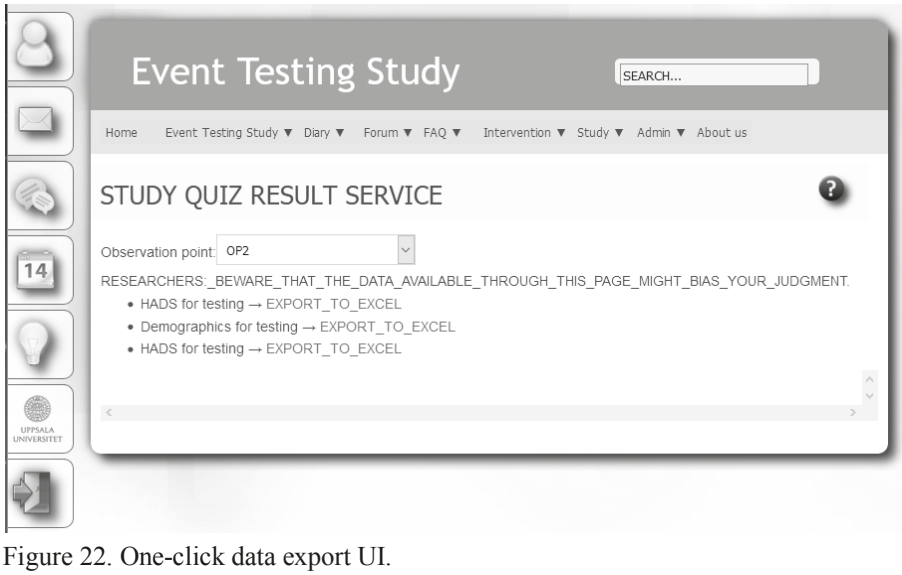


Figure 22. One-click data export UI.

The *one-click* data export feature's UI was also very simple. When a clinical researcher logged in to the U-CARE software system, a dropdown list was

shown with observation points (OP) based on his/her selected default research study. By default, the first OP (e.g., baseline) was selected in the dropdown list. Depending on the selected OP, each questionnaire was listed as a link for downloading respondents' data from that particular questionnaire. The clinical researchers could download a Microsoft Excel spreadsheet containing data from one questionnaire at a time with just one click (see Figure 22). The questionnaire data was exported in the clinical researchers' desired pre-defined format (i.e., question number as the column heading and one row per research participant, with responses to questions in the corresponding column). The clinical researchers could import data into their choice of data analysis software and perform data manipulation themselves; for example, filtering data by research participant groups, removing unwanted columns or rows, et cetera.

Intervention and evaluation

The *one-click* data export feature was designed to allow clinical researchers to export questionnaire data themselves, instead of having developers export data. However, only developers had access to the *one-click* data export feature initially. During multiple evaluation sessions (December 2013–May 2014), the ADR team received feedback from various stakeholders. For example, there was a problem with exporting a questionnaire (the PBQ questionnaire) in the JUNO study. While fixing this bug, the ADR team also identified another questionnaire (the WCQ questionnaire) that did not export correctly. HTML elements in the data affected data export while generating the XML document.

After some use, the clinical researchers stated that "it would be good to have a link to a [one-click data export] page somewhere in the [clinical] researcher view [in the U-CARE software system]" (2014, Product backlog, #1076). The ADR team configured a menu that allowed the clinical researchers to export data. At this stage, the one-click data export was not in compliance with *the principle of restricted data access*. The Information Systems researcher pointed out the importance of considering data export design principles while commenting on the development task: "an important issue to address is [the use of] policies for what, when and by whom data may be exported" (ISR-1, 2014, Product backlog, #1076). At a later stage, access to the menu was reconfigured to allow access only for the clinical researchers, thus limiting access to data to emphasise accountability as a key quality characteristic for eHealth research software (2014, Product backlog, #1152).

Reflection and learning

As a consequence of the one-click data export feature and the clinical researchers' need to export data themselves, the U-CARE management prepared guidelines for the clinical researchers regarding data extraction before the research study was completed. The guidelines consisted of two main

points, to emphasise the need for careful reflection before extracting data from an ongoing study as:

> First, it is possible to extract some data before the study is completed, as long as variables that are primary outcome measure and variables that compare the study groups are not analysed. Second, variables or analysis of variables that may be important in the final analysis of the study should not be used. (2013, U-CARE data extraction guidelines)

In this state, the one-click data export allowed any researcher (with a researcher user role) to log in and export data at any point in time in the research study. This was not only contrary to the data export policies, regarding when, by whom, and what data may be exported, but also contrary to *the principle of restricted data access*. As an afterthought, developers added a warning message stating "Researchers: beware that the data available through this [web] page might bias your judgment." Design principles and guidelines helped the ADR team make design decisions and build the software. Clearly, *the principle of restricted data access* was not yet fully implemented. The lesson learned from this scenario was that design principles were not useful until they were applied during the development process. The data export design principles were refined, grouped, and formulated using the structure discussed in Section 4.3 to make design principles clearer and more precise (see Table 15).

Table 15. Design principles for data export in eHealth research software (version 4)

Design principle	**Specification**
The principle of simplicity	Provide easy-to-use data export functionality in order for [clinical] researchers to export data, preferably by a single click via a simple UI, given that such functionality should not require in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	Data export functionality should be divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	Data export functionality should be customisable in order for [clinical] researchers to tailor [their own] research data export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats such as CSV, XML, or spreadsheet.
The principle of accountability	a) Privacy: Data export functionality should anonymise[42] data in order to ensure research participants' privacy, given that

[42] All data must be de-identified so that a re-identification by use of different sets of published data or by linking exported data with other publicly available data sources is impossible. Note

	such anonymised data do not contain identifiable data or that ID fields are encrypted. b) Security: Data export functionality that enables the clinical researcher (i.e., study owner or principal investigator) to restrict data access in order to enforce governance policies, data extraction and ethical guidelines, given that such data access restrictions can be researcher-specific (based on access privileges). c) Auditability: Data export functionality should log all activities related to data export in order for study owner to fulfil audit and regulatory requirements, given that such logs store all data export events to facilitate follow-up by the study owner.

BIE Cycle V

Build

The one-click data export feature encountered performance-related issues during its use over time. Sometimes, when the clinical researchers were exporting data, the U-CARE software system took too much time to respond, and was perceived as *not responding*. This happened when the data export process running on the web server took too long, and the web application UI was waiting for the data export response. The clinical researchers felt frustrated and became impatient, making it difficult for them to wait for the U-CARE software system to generate the data export file. As a result, the clinical researchers often clicked on the browser's back button or clicked on the export link again, causing the data export web request to resend and be restarted. Sometimes, there was a session timeout from the web server, forcing clinical researchers to login and start the process again.

There were multiple reasons for slow responses from the data export feature. First, there were large amounts of data, and metadata[43] needed to be processed in order to generate data export results. Second, for every data export request, the process started from scratch for all research participants who responded to a particular questionnaire. With time, this performance issue became critical. In November 2014, a three-day workshop[44] (Figure 15, pt. ii-i) was organised with the primary objective of improving the software development process with a focus on quality assurance. The data export performance issue was also discussed. Various refactoring possibilities of the data export feature were suggested:

that terms *de-identification* and *anonymisation* are often used interchangeably in different contexts in the literature.

[43] Metadata is data [information] that provides information about other data. Here, this refers to data that are stored when the questionnaires are designed by clinical researchers, defining the questions, the types of questions, the flow of the questions, et cetera. Such metadata about questionnaires are required to present a questionnaire to research participants, to collect the responses and to export responses.

[44] This workshop is discussed in detail in relation to Case-ii (i.e., technology upgrade).

> […] one of the reasons this [data export] is complex, is that [the clinical researchers] have very specific requirements on how this exported data should be formatted. For instance, the current export function is very slow but solved this formatting requirement. What happens is that for instance if there are checkboxes [multiple choice questions for research participant, where you are supposed to] mark the alternatives that fit your mood, like [a)] I am alert, [b)] I am happy [c)] I am sad, and the [research participants] select a number of those [options, the clinical researchers] want the values of the selected [checkboxes]. However, if radio buttons are used, they [clinical researchers] might want something else. The way it is presented is very dynamic, the number of columns varies, and it can be even be different for different [research participants], and so forth. So, one of the things that was tricky was not just to get the right data and put in the right column, but also to format it in different ways depending on the [question type and] actual response […]. (ISR-1, 2014, Developers' workshop)

Storing the intermediate data export result in the database was suggested:

> the merits of the current solution are that they solve the requirements and reuse some of the business logic code to find things. The drawback of the current solution is that it takes time. […] Once we have extracted a lot of [questionnaires], most of them never have to recalculate, because they do not change once they are submitted. So, if we use the current logic and just cache or store old results in a table, it could be very easy. (ISR-1, 2014, Developers' workshop)
>
> that is what [ISR-4] was saying; that was his idea […] do data transformation at once, and once it is transformed then it is much more easily extracted and reported, but right now we transform every time we export data. (Dev-6, 2014, Developers' workshop)

> all we need to do is to say *When was the last time this thing was exported?* and then in the stored table we have all the already transformed things, and then when we do it again we just pick the new things that happen after that. (ISR-1, 2014, Developers' workshop)

Also, the suggestion was made to refactor parts of the code where (overly) complex queries were phrased in LINQ to SQL (.Net language-integrated query for relational data), to improve readability, maintainability, and possibly performance. One interesting thing was pointed out by Dev-4, during the data export feature's design discussion, regarding the relationship between study design and data export. He said:

> […] we can have a generated [static database table] when [clinical researchers] configure the studies […] when [they] decided which questionnaire goes where then we could design this [static table]. (Dev-4, 2014, Developers' workshop)

> When a study is frozen [the static table is frozen too]. When a study is unfrozen then [there may be a change in the static table]. (ISR-4, 2014, Developers' workshop)

> […] we do not really know what they want in the end. That is the problem, but there are two things here. There is an immediate problem which is we have an export function which does not perform well. That can be quite easily solved by not recalculating things every time. And then we have another issue here [related to our thoughts on] what they will want in the future […] if things change, we still need to know what it looked like before that change. So, we can match the actual results […] And we can do that, but it requires quite a lot of queries to do it, so this will help us to better understand, at a given point in time, what the configuration of the study [is] like. (ISR-1, 2014, Developers' workshop)

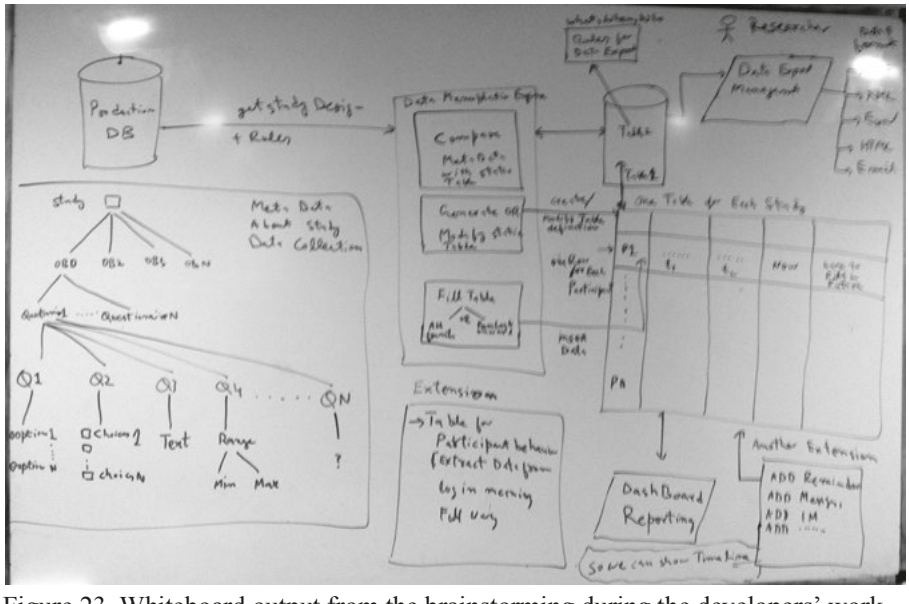


Figure 23. Whiteboard output from the brainstorming during the developers' workshop.

A conceptual architecture for the data export feature was proposed and drawn on a whiteboard (see Figure 23):

> […] when we have a [static linear] table in the system. And the [data manipulation engine] is filling this [table] with data daily. And then we can make a [user] interface for the [clinical researchers]. They can select what fields they wanted [for example] they want just the results of a questionnaire, not all the [responses], then it can be more [customised and user-friendly] […] the current [one-click] data export [feature is creating a temporary table] that [part] is a sort of [data extraction engine] and the second part is inserting the data [which is already functioning as well]. (ISR-4, 2014, Developers' workshop)

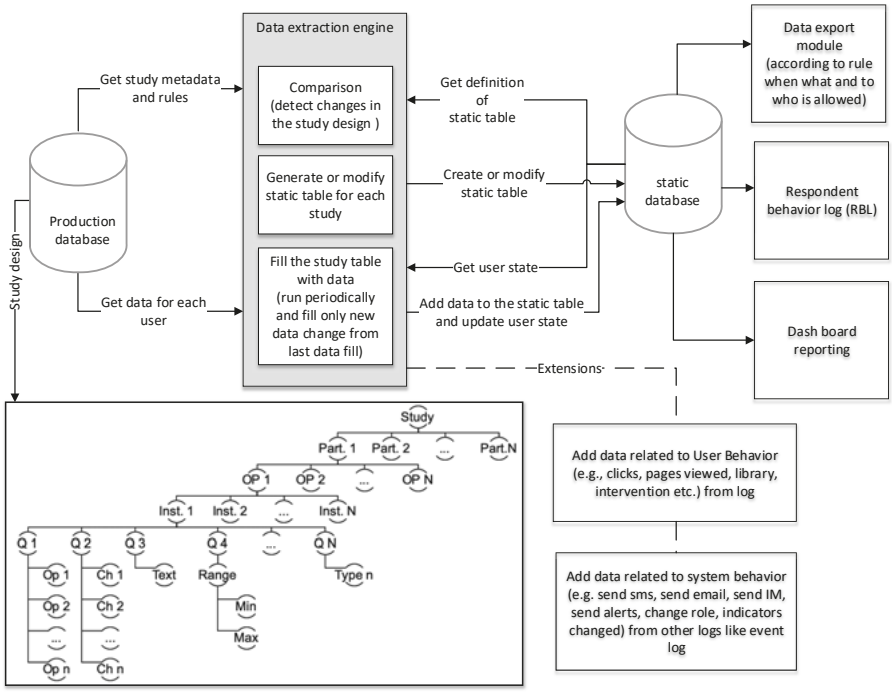


Figure 24. The initial conceptual architecture of the two-stage periodic data export.

The initial conceptual architecture led to further discussion within the ADR team and resulted in an improved conceptual architecture (see Figure 24). The conceptual architecture is referred to as the *two-stage periodic data export* in the U-CARE context (Figure 15, pt. i-j). The one-click feature improvements were suggested based on learnings from the data export feature, the data manipulation features (i.e., respondent behavioural log, event service, and cache service), and deliberations within the U-CARE development team. Based on the discussion, it was suggested that the data export feature needed to work periodically, like the existing feature event service, and export a small chunk of data from the hierarchical structure into a linear form, as a single row per research participant in another static database. In this way, data would not be processed at the time of the data export request; instead, data would be prepared periodically, over time. In this way, two separate databases would balance the load and the U-CARE software system would remain responsive all the time. Also, the data export feature was to be made more efficient through the use of a static database with pre-processed data.

The two-stage periodic architecture for data export assumed that *once the research participant submitted the responses, there was no possibility of making changes*. Data collection was designed with consideration of the ease of representation and storage in hierarchical form, whereas the data export feature design required a static linear form. Instead of exporting data from scratch

each time, proactively preparing the data would make subsequent data exports simple, consistent, robust, and efficient. In case of any bugs, the developers could focus on a specific period only and regenerate data from a particular point forward.

At the time of the three-day workshop, due to the limited resources for development and the presence of high priority items in the product backlog, the two-stage periodic data export architecture was postponed for future refactoring. To make data export efficient for the time being and the U-CARE software system responsive while executing data export requests, the ADR team stored all the responses of a research participant to a particular questionnaire together, in a specific observation point, in XML format in the database (2015, Source code repository log). It was a quick fix, based on the ideas discussed during the workshop. Whenever a questionnaire was exported, the data export feature looped through the research participants (who were supposed to respond); if the responses to the questionnaire were stored already they were used as-is, otherwise the responses were processed, stored as XML for future use, and used in the current request. In this way, every time a clinical researcher exported a particular questionnaire, stored data were used and only the latest response submissions were processed. This improved the data export feature's response time enormously (2015, IT meeting minutes).

Intervention and evaluation

The one-click data export feature was not only evaluated using test studies with test data, but also using some real pilot and full-scale studies with real data. Table 16 lists research studies whose data were exported using the one-click feature. Each data export request led to refinement and improvement in the one-click data export feature, for example with regards to performance and the handling of inconsistent and incomplete data.

Table 16. Data exported using the one-click data export feature

#	Research Study
1	U-CARE AdultCan Pilot
2	JUNO Pilot
3	U-CARE Heart Pilot
4	AIDA I {A, B, C, D}

The list of studies is based on the IT meeting minutes, the product backlog, the U-CARE software system's log, and the clinical researchers' own recollections of their use of the one-click data export feature.

Further development of the one-click data export feature was stopped, though the feature still had four known limitations. First, the clinical researchers had to export all questionnaires, one by one, for every observation point of a study and then put them together in one file for further analysis. The process of

merging data from multiple data files was error-prone, difficult and time-consuming. It was easier to maintain consistency across data sets with fewer, larger files. It was also more convenient for researchers to select a subset from a more substantial dataset, as compared with manipulating and filtering data in the files. Furthermore, they could export only data related to questionnaires[45]. Thirdly, clinical researchers could export data at any stage in the research study lifecycle, which was against the data export policies. Fourthly, the U-CARE software system logged the data export event, but it was not possible to trace[46] what data had been exported (based on a 2014–2016 log analysis).

Reflection and learning

During the one-click feature evaluation, one important realisation was made: that the design principles were mainly focused on the design of the data export feature and that the intended target audience was the software developers. However, during the appropriation in the empirical context, it was realised that the design principles needed to communicate key characteristics of the data export feature to a wide variety of stakeholders. In the U-CARE context, other stakeholders like clinical researchers and the management needed to be included in the intended audience. Also, there was a need for considering the research study design. Implementation of design principles required additional features in the study design section of the U-CARE software system, so it could be specified what data could be exported, when, and by whom. Much like the research study freeze function that existed in the study design section of the U-CARE software system, there was a need for study owners to enable or disable data export. Furthermore, any export of data needed to be saved for audit and to ensure adherence to data export guidelines. This was based on the existing practice of the clinical researchers in U-CARE, who kept track of changes in the study design. These study design change records were supposed to be used during data analysis to discuss any implications of such

[45] The clinical researchers could export reminders from another feature in the U-CARE software system that was partially based on the one-click data export feature. Reminders were short messages sent to the research participants to prompt them to engage or update with ongoing treatment activities. Reminders were automatically sent by the U-CARE software system event service. However, to make sure that they were sent, the clinical researchers needed to export the log related to the reminders.

[46] This refers to a configuration error in the U-CARE software system logging mechanism (the authorisation feature discussed in the data export feature's BIE cycle-i) with an unknown cause. There were two actions that needed to be logged, i.e., QuizResultService and QuizResultToExcel. QuizResultService was logged (the user ID, time, session ID, URL, et cetera) and provided some information on when the feature was used to export a specific study and by whom. But the action QuizResultToExcel was not logged, though it was required (for accountability compliance) to get additional context data that included the observation point and the questionnaire references that were exported. This realisation of the logical error occurred at a later stage during analysis of the design process by ISR-1, ISR-4, and Dev-6.

changes. Likewise, the study design changes had implications for the data export. The data extraction guidelines highlighted the role of the research study owner and it was advised that the clinical researchers "reconcile [their] research questions with the [study owner] who is responsible for the study, who has an overview of the planned research questions, articles and possible analyses in conjunction with the final evaluation" before data extraction (2013, U-CARE data extraction guidelines). Again, this could be ensured by adding additional configuration capabilities in the study design regarding data export.

Table 17. Design principles for data export in eHealth research software (version 5)

Design principle	**Specification**
The principle of simplicity	Provide easy-to-use data export functionality in order for [clinical] researchers to export data, preferably by a single click via a simple UI, given that such functionality should not require in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	Data export functionality should be divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	Data export functionality should be customisable in order for [clinical] researchers to tailor [their own] research data export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats such as CSV, XML, or spreadsheet.
The principle of accountability	a) Privacy: Data export functionality should anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, **and datetime field(s) are removed or offset.** b) Security: Data export functionality that enables the clinical researcher (i.e., study owner or principal investigator) to restrict data access in order to enforce governance policies, data extraction, and ethical guidelines, given that such data access restrictions can be researcher-specific (based on access privileges)**, time-specific (i.e., at multiple intervals with the same/refreshed/additional datasets, or one-off after the study completion or termination) and data-specific (i.e., partial, full, or selected datasets).** c) Auditability: Data export functionality should log all activities related to data export in order for study owner to fulfil audit and regulatory requirements, given that such logs store all data export events to facilitate follow-up by the study owner **and enable audit organisations to confirm compliance with legislation and ethics.**

Table 17 lists the revised design principles (any changes from the previous version and additions are highlighted in bold). The opportunities for learning

and refinements in the design principles continued during the prolonged and actual use of the data export feature in the U-CARE context. This reveals the need to consolidate design principles and evaluate their usefulness in the actual context over time.

Artefact Use Over Time and Learning

The one-click data export feature was limited to export of questionnaire data. The clinical researchers' data needs emerged over time, and they expressed additional requirements. They required being able to export of a variety of data. The U-CARE management streamlined the process for the clinical researchers, considering these complex and resource-hungry data export requirements, as follows:

> when you [clinical researchers] want to extract data, you put a description in the [product] backlog and then a developer will make contact, and you can discuss the details. (2014, IT meeting minutes)

The diversity of the clinical researchers' data export requirements can be recognised in the data export requests, such as:

> We at U-CARE Heart want to export [small sample of N] research participants as follows: evenly distributed within each group based on 1) group (treatment, control, reference) and 2) randomisation date between autumn 2013 and spring 2016.
>
> i. Answers to all questionnaires from all observation points (OP)
> ii. Hospital, Inclusion Date, Randomisation Date, CBT Intervention (Version), OP Date (All)
> iii. Number of logins (date/time, if possible, for all logins)
> iv. Number of visits to different items in the library (date/time, if possible)
> v. Number of visits to forums (possibly which threads) (date/time, if possible)
> vi. Number of visits to Questions & Answers (date/time, if possible)
> vii. Number of logins not related to OP (for processing only) (date/time, if possible)
> viii. First activity (item opened?) in treatment (date and time)
> ix. Last activity in treatment (date and time)
> x. Number of items done (marked cleared) (date/time, if possible)
> xi. Number of steps done (marked cleared) (date/time, if possible)
> xii. Submission of first homework (date and time)
> xiii. Submission of last homework (date and time)
> xiv. Number of homework [items] sent (date and time)
> xv. Number of modules started and finished (date and time)
> xvi. Number of messages sent to IM service (in addition to home assignments) (date/time)

> Also, we would like you to have a code key with personal number and ID number prepared for these [N research participants] to send to UCR [Uppsala Clinical Research Center] when they are ready. (CR-3, 2016, Product backlog)

Another data export request related to research participants, even if they were not randomised.

> We would like to combine all the questionnaires from all observation points in all groups, into a master extract for all answers to all quizzes, with one row per research participant. Each row will include the study group: control, treatment, reference; as well as the baseline research participants who were not randomised. (2017, Product backlog)

Here is a data export request is related to research participants' communication with additional requirements on data presentation:

> Create extracts from IM communication between therapists and patients during CBT treatment on the portal. For each IM, detail to or from therapist, sender or receiver, reason for IM (feedback, and communication), timestamp, subject, and body of the message.
>
> Create a simple matrix template with one column per day of CBT from 1 to 98, with one row per research participant, and a count of IMs to or from therapist per day.
>
> Produce three versions of the template, depending on the type of IM: 1) message from user [research participant] to therapist, 2) feedback from the therapist on homework and 3) message from therapist to the patient [research participant] (not feedback).
>
> The final report is a simple matrix of the 11 modules in columns, with one row per patient, showing the date the module was activated, if this happened. (2017, Product backlog)

The development team did not have any idea how the clinical researchers' data export requirements would develop over time. The above rich illustration of data export requirements was useful in the U-CARE context. Also, it might be helpful for researchers and practitioners in other contexts.

There were multiple studies in the U-CARE software system and these all had different study designs (protocols). The volume, variety and velocity of data entailed different challenges. For example, the U-CARE log accumulated rich information regarding research participant activities and events in the U-CARE software system. The log contained more than 7 million entries (in December 2017). The potential to gain valuable insights from log data attracted substantial interest from the clinical researchers. However, data cleansing, interpretation, understanding, and analysing has its challenges. The clinical researchers in U-CARE hinted at them, as illustrated by the following quote:

> The log needs to be pre-processed to get the information needed to create reports based on [research participant] activity. Need to analyse the current requests for extracts, and compare with what is stored in the log table parameters field for the relevant actions. (CR-2 and CR-3, 2017, Product backlog)

Data related to treatments (e.g., CBT) ware very complex and the tree structure was comparable to the previously discussed hierarchical structure of the questionnaires. Different research studies had different CBT treatment structures, for example, a) fixed, adaptive or self-tailored modules, b) modules contain zero or more steps, c) modules or steps contain different set of items, and d) modules available at once or sequentially, one by one. Above all, research participants' CBT items deviated from one another.

Practitioners (e.g., database administrators, developers, et cetera) tended to export data directly from databases using the DBMS features (Aspin, 2012) such as built-in data export and structured query language (SQL). Due to the diversity of data export requests, and the need to get things done quickly, the development team started to export the data directly from the DBMS using SQL. Later, multiple custom-made data export applications were developed and used to export data (see Appendix C.3). The use of the custom-made application(s) over time also uncovered many unanticipated challenges and problems concerning data export, such as missing data. Most of the real studies exported their data through the custom-made application(s). Table 18 lists names of full-scale studies, types of data exported, dates of data export requests (issue number if an issue was created in the product backlog), and dates when the clinical researchers received data. There was a waiting time between a data export request and data being received by the clinical researcher. The waiting time differed between data export requests, for various reasons. Waiting time as mentioned in the table below does not represent the development time or the development team's engagement in the data export task. The objective here is to show that data export was not instantaneous for the clinical researchers. This caused a problem, as the clinical researchers did not feel in control of their own research data; a deviation from *the principle of malleability*, which was articulated by the ADR team as a key principle for the design of data export feature in the U-CARE software system.

Table 18. Data exported using custom-made data export applications

#	Research study	Data	Requested (issue)	Exported
1	U-CARE AdultCan	Activities in the portal	2014-08-18 (#1298)	2014-08-28
2	U-CARE Heart	Library and forum visits	2014-08-18	2014-10-27
3	AIDA I	Communications between research participants and therapists	2014-11-06 (#1415)	2015-01-09
4	JUNO	Questionnaires	2015-03-25	*
5	ISAK	Questionnaires	2015-07-23	2015-07-23

6	AIDA II (A and B)	Library items visited, logged in time, and user behaviours	2015-10-12 (#1912)	2015-10-12
7	U-CARE AdultCan	Questionnaires	2015-10-13	*
8	U-CARE Pregnant	Questionnaires	2016-02-08 (#2094)	2016-03-16
9	U-CARE Heart	All data with limited research participants	2016-10-31 (UC-241)	2017-01-24
10	U-CARE AdultCan	Limited extracts of baseline data	2017-04-11 (UC-590)	2017-04-24
11	U-CARE Heart	Extracts from IM communications between therapists and patients during CBT treatment	2017-09-21 (UC-722)	2017-09-26
12	UPPS	Collected observation point data from the study, including questionnaires filled out by staff on behalf of the [research participants] patients	2017-09-21 (UC-723)	2017-11-20
13	U-CARE AdultCan	A reduced observation point (OP) extraction, up to and including OP 8	2018-01-15 (UC-910)	2018-02-05
14	U-CARE AdultCan	Extraction to merge control and treatment research participants' answers	2018-02-05 (UC-941)	2018-02-06
15	U-CARE AdultCan	A matrix extraction of number of accesses and total view times for library multimedia views [of research participants in treatment group (in step one)]	2018-04-04 (UC-1028)	2018-05-02
16	U-CARE: Online behaviour patterns in internet-based intervention studies	Extraction of log data to understand behaviour patterns [on the U-CARE software system]	2018-05-28 (UC-1086)	Ongoing
17	JUNO	All quizzes from the first two observation points, using the same OP format (non-merged) as in April	2018-06-21 (UC-1142)	2018-06-25
18	UPPS	An updated extraction for data collected since February 2018	2018-09-20 (UC-1278)	2018-09-20

The list of studies and related data is based on IT meeting minutes, product backlog, and the recollections of the development team, based on email correspondence with the clinical researchers [updated: March 06, 2019]. * data was exported but date is unknown.

These custom-made applications were developed, maintained, and executed by a single developer. It led to critical knowledge being confined to one individual (a.k.a., a knowledge silo). Ignoring proven best practices, for example, collective code ownership, code review, test-driven development, quality assurance, and several others, led to a technical debt in the U-CARE software system.

The custom-made data export applications dealt with three of the four limitations of the one-click feature. First, the clinical researchers received a single consolidated file in their desired format. Second, they could request any data that was available in the U-CARE software system. Third, the data export

requests were routed through the U-CARE management and the management handled the data export policies. The clinical researchers did not have direct access to these applications. It was appropriate that the U-CARE management, in a sense, controlled data access (by whom, what data, and when), but it was not empowering for the clinical researchers. The existing log mechanism of the U-CARE software system did not track these custom-made applications. Also, there was no manual audit log maintained regarding data export. Due to this, there was no record available that enabled audit trails and traceability. Not even the product backlog contained any record of all the data export requests. It was also evident in discussions with the clinical researcher that they wanted to export data themselves. The clinical researchers needed to be able to export data themselves, which was expressed as follows during an evaluation session:

> [regarding custom-made data export applications] it would be good if I could do [the export] it myself, maybe they [management] want to have control over who [does the] exports, but that can be managed with some functionality […] it could be timestamped, and id-stamped […] what export et cetera. […] [it is] very good if you can export yourself […] normally it would be enough to have the export at one time, but often something goes wrong or something comes to mind and you export and […] then it would be very inflexible to have it going through [specific developer] all the time, so it would be better or I think it would be preferable to do it yourself […]
>
> [regarding one-click data export feature] here you have separate files for every questionnaire […] so if we have a tree structure [… it] can be opened up to all the questionnaires […] and we could have some checkboxes [… that] could be to select questionnaires for export [… and] then have some grouping options […]
>
> But then things need to be exported from a behavioural log […] that could be in a separate file […] it would be ok if data export requests were processed overnight and [that you on the] next day receive a link to download the result. (CR-3, 2017, Interview)

Both the one-click data export feature and the custom-made data export applications were used at this time. There were full-scale research studies for which data already had been exported, analysed, and where the clinical researchers had published their results; which demonstrated the utility of these features. In addition, there were many ongoing research studies which had to export data. Nonetheless, further development on the feature remained necessary due to three limitations. First, the custom-made applications did not have an accountability log, which the previous data export feature versions did. For example, there had been log entries that showed how many times a clinical researcher had exported the data. Second, there was a dependence on a specific individual developer for the data export. Third, the clinical researchers were not empowered to export data themselves.

In February 2017, CR-2 and Dev-6 proposed a strategy for developing a comprehensive data extraction for the U-CARE software system. They described the data collection and execution of RCTs in the U-CARE software system with an analogy to a banking system. A banking system allows customers to manage their accounts, given that they observe the complex rules of the bank. Such a system is optimised to save customer transactions, but not to facilitate any future analysis, for example, customer behaviour. Similarly, the U-CARE software system could be viewed as a transaction-based system and "its purpose is to correctly move each study research participant through the various paths of the system, while collecting data along the way, and verifying that the business rules of the study have been followed to the letter." In this strategy, it was argued that most of the data collected would not be subject to any retrospective changes, except in a few very exceptional cases. CR-2 and Dev-6 proposed "to build an archive model, which is *add only* – as new information becomes available in real time, it can be appended to the archive, but there is no need to edit the archive." The purpose of the archive was "to consolidate participant-based behavioural data, for easy and meaningful extraction by authorised users." They also cautioned that such an archive posed a security risk for unauthorised access. So, they proposed data security rules to be followed for any data extraction from the archive such as:

> i) The data officer, often the [principal investigator], defines and publishes exactly what data is extractable, by whom, and under which circumstances; ii) All successful production extractions from the database archive need to be logged by timestamp and authorised user; iii) Rogue attempts to access extractions must be reported to the security officer. (CR-2 and Dev-6, 2017, Data extraction strategy)

The strategy also emphasised that the data export presentation would be in the form of rows and columns, including both a linear and a matrix format. Based on previous data export experience, they argued that data export is a (sort of) sequential narrative, as most of data extraction requests include timestamps and durations of research participant activities. The activities are performed either by or for the research participants. Activities also included automated events by the system acting on behalf of the stakeholders (e.g., a study owner, or psychologist). The data extraction strategy was based on learnings from previous use of the data export feature(s) and influenced by discussions around the *two-stage periodic data export* architecture (Figure 15, pt. i-j). In the ADR context, this is well aligned with the BIE stage principles, i.e., reciprocal shaping (the artefact and U-CARE organisation shaping one another) and mutually influential roles (software developers, Information Systems researchers and the clinical researchers influencing one another). Based on the discussion within the development team, an improved conceptual architecture was proposed (Figure 15, pt. i-k).

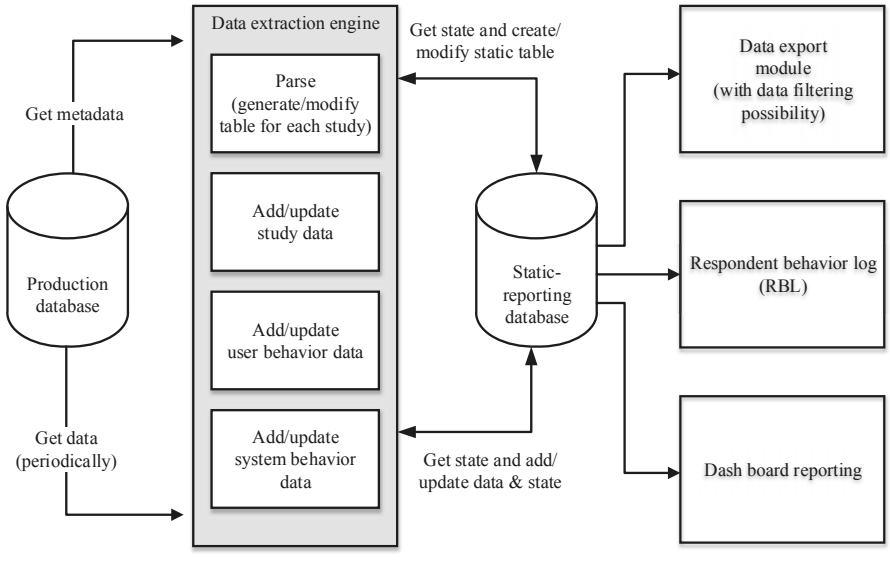


Figure 25. The improved conceptual architecture of the two-stage periodic data export.

Figure 25 shows the conceptual architecture for the data export feature, consisting of two stages. The ADR team concluded that the data export feature construction would be easier with a two-stage decomposability (*the principle of modularity*). In the first stage, a data extraction engine was to use the metadata (related to the research study, questionnaires, and treatment) and parse them to create/modify static tables (ideally one table per research study). This step would be linked and activated based on the study design freeze-unfreeze mechanism. It was assumed that once a research study was frozen, its structure would not change until it was unfrozen. The engine would also be responsible for periodically extracting data from the production database and inserting them into the static reporting database. In the second stage, the proposed data export module would be used by the clinical researchers to export data. The proposed data export module was to keep track of when, what, and how much data was to be exported by whom, and also to anonymise it. In other words, it would enforce *the principle of accountability*. In accordance with the one-click data export feature and *the principle of simplicity*, the UI of the proposed data export module would be simple. Similarly, the proposed data export module would be simple with regards to functionality, thanks to reduction and hiding of complexity, decomposition, and organisation into multiple layers and modules. Also, the data export module would allow data filtering and saving the data export as a template for subsequent exports, similar to the generic data export feature, as it was requested by the clinical researchers (CR-3, 2017, Interview). Data filtering and the data export template would enable user-malleability, as the clinical researchers could customise

their data export requests. This led the ADR team to revise *the principle of malleability* (see Table 19).

This two-stage architecture would also facilitate features[47] like RBL, dashboard reporting, et cetera. Research data collected automatically via logs (system log, event log, exception log, et cetera) and extracted to the static reporting database could be quickly analysed and reported by other features. For example, RBL could process user behaviour data (i.e., clicks, pages viewed, library items visited, interventions followed, questionnaires answered) and dashboard reporting can process system behaviour data (i.e., send text messages, send emails, send IMs, send reminders, send alerts, and make role changes) from the static reporting database. The data extraction engine could also be enhanced to support the U-CARE software system event and cache services data processing. It is important to note that the two-stage architecture is partially based on the event and cache services' architecture. In essence, the two-stage periodic data export feature would provide a better and simpler data export functionality, allowing the clinical researchers to export consolidated data efficiently.

Although the conceptual architecture of the two-stage periodic data export had been reviewed by the development team for an extended period, it remained under consideration. The ADR team convinced the U-CARE stakeholders to use the existing knowledge and experience to develop a data export using the two-stage periodic data export architecture. Stakeholders like Dev-6, CR-2, CR-3, ISR-4 and others, were inclined towards data export feature refinement, but the decision was made not to prioritise and allocate resources for such large-scale refactoring at that point. Nonetheless, the conceptual architecture served as a tool for discussion. The development of a proof-of-concept prototype based on the proposed architecture was considered as a way forward in subsequent BIE cycle(s).

[47] It is not possible to provide complete details of these artefacts as they fall outside of the scope of the dissertation; details will be published by fellow Information Systems researchers separately.

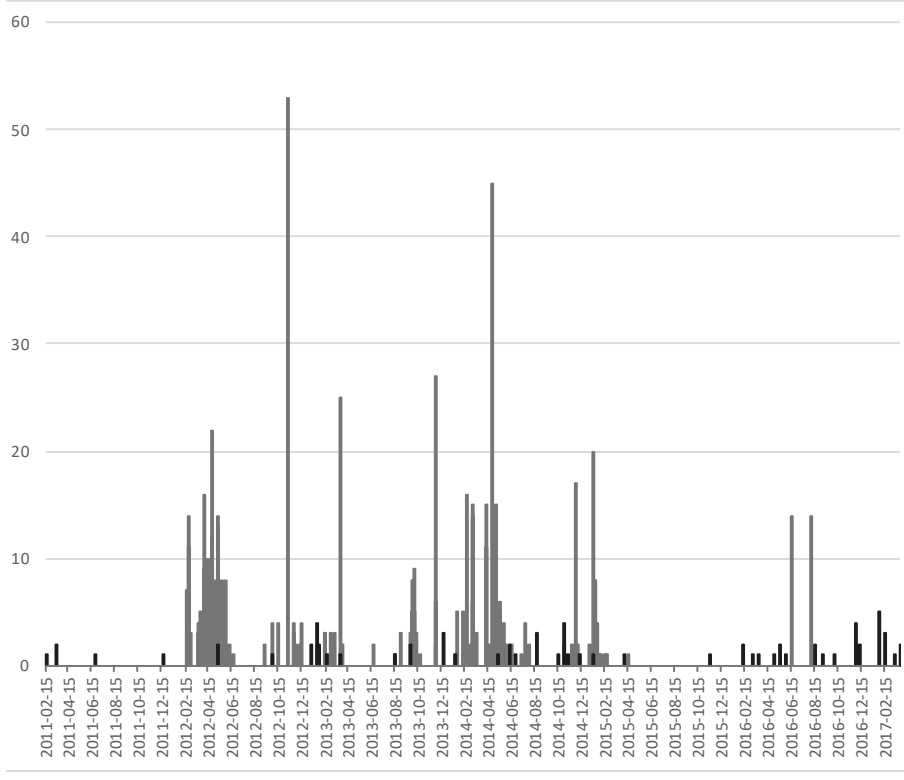


Figure 26. Data export feature source code changes vs. the discussions in IT meetings regarding the data export feature.

Retrospective analysis of the design process, using CoDisclose, revealed that the data export feature remained active throughout the U-CARE software system development. Figure 26 represents IT meeting instances[48] in which the discussion of stakeholders concerned the data export (black bars) and changes[49] in the U-CARE software system source code (light grey bars) based on data from February 2011 to February 2017. The code changes are only related to the *generic* and the *one-click* data export features because – as explained above – the custom-made applications were not part of the code repository.

The data export design principles were revised and augmented based on the lessons learned from the prolonged and actual use of the data export feature in the U-CARE context (Figure 15, pt. i-l). One such lesson was that data export was as important as data collection for eHealth research software in an academic research context. There was a need for continuous reflection on the data export functionality, in each system development phase, for example, during every sprint (following the agile methodology), to identify any

[48] Number of data export related keywords found in IT meeting minutes.
[49] Number of data export related source code files.

adjustments needed in the data export feature due to changes in data collection, business rules/domain, and parts of the software system. Also, such reflections require sample data exported from real studies to identify data discrepancies, investigate the reasons for these, and resolve them.

Table 19. Design principles for data export in eHealth research software (version 6)

Design principle	**Specification**
The principle of simplicity	Provide easy-to-use data export functionality in order for [clinical] researchers to export data, preferably by a single click via a simple UI, given that such functionality should not require in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	Data export functionality should be divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	a) Customise: Data export functionality should be customisable in order for [clinical] researchers to tailor [their own] research data **and descriptive metadata** export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats, such as CSV or XML, or **tailored for** spreadsheets **or common statistical packages, in a way that is useful for downstream applications.** **b) Filter: Data export functionality should allow data filtering in order for [experienced clinical] researchers to customise data export according to their preferences and needs, given that such functionality should guide the researcher to filter exportable data and allow the researcher to save and reuse their data exports as templates.** **c) Schedule: Data export functionality should allow scheduling data export requests in order to get data after specified intervals [based on study design] or when data is available [in cases where the volume of data would increase data export processing time].**
The principle of accountability	a) Privacy: Data export functionality should anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: Data export functionality that enables the clinical researcher (i.e., study owner or principal investigator) to restrict data access in order to enforce governance policies, data extraction, and ethical guidelines, given that such data access restrictions can be researcher-specific (based on access privileges), time-specific (i.e., at multiple intervals with the same/refreshed/additional datasets, or one-off after the study completion or termination) and data-specific (i.e., partial, full, or selected datasets).

	c) Auditability: Data export functionality should log all activities related to data export in order for study owner to fulfil audit and regulatory requirements, given that such logs store all data export events **[when (timestamp), who (user identity – role), how (encrypted/plain text), why (purpose specification and use) and what (data specification)]** to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics.

Table 19 lists the resulting design principles (any changes from the previous version and additions are highlighted in bold).

5.3 Formalisation of Learning

Design artefacts are evaluated mainly based on the functions and properties they possess, and how these matches the end-users' requirements. However, requirements change as the end-users' environment is always changing (Truex *et al.*, 1999). Data export is a key functionality for eHealth research software in an academic research context. Given the emergent character of eHealth research software, data export, in order to be useful, needs to be based on: a) an understanding of the needs of the stakeholders (mainly the clinical researchers) who want to export [or reuse] data; b) a data export design that complies with design principles which enact best practices in eHealth software design, while designing data export in eHealth research software; c) appropriate key quality characteristics in order for eHealth software to be sustainable; and d) the data export functionality should be reflected upon continuously by both researchers and software developers in order to ensure data integrity. The case shows that malleability, decomposability, simplicity, and accountability are the key quality characteristics for sustaining the usefulness of eHealth research software in the academic research context. From the software development perspective, the most significant hurdle would be the continuous refactoring and quality assurance of a data export feature in a changing operational environment where the clinical researchers' data export requirements evolve. Hence, sustaining such quality characteristics is a challenge due to the normally very limited resources for software development in an academic research context. However, I agree with Störrle *et al.* (2016) recommendation that "as software complexity (and cost) grows exponentially with size, you simply have to invest in quality. Don't hesitate to scrap and redo a project or component." The learnings from this case are articulated and formalised as design principles in the following table.

Table 20. Design principles for data export in eHealth research software

Design principle	Specification
The principle of simplicity	Provide easy-to-use data export functionality in order for [clinical] researchers to export data, preferably by a single click via a simple UI, given that such functionality should not require in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	Data export functionality should be divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	a) Customise: Data export functionality should be customisable in order for [clinical] researchers to tailor [their own] research data and descriptive metadata export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats, such as CSV or XML or tailored for spreadsheets or common statistical packages, in a way that is useful for downstream applications. b) Filter: Data export functionality should allow data filtering in order for [experienced clinical] researchers to customise data export according to their preferences and needs, given that such functionality should guide the researcher to filter exportable data and allow the researcher to save and reuse their data exports as templates. c) Schedule: Data export functionality should allow scheduling data export requests in order to get data after specified intervals [based on study design] or when data is available [in cases where the volume of data would increase data export processing time].
The principle of accountability	a) Privacy: Data export functionality should anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: Data export functionality that enables the clinical researcher (i.e., study owner or principal investigator) to restrict data access in order to enforce governance policies, data extraction and ethical guidelines, given that such data access restrictions can be researcher-specific (based on access privileges), time-specific (i.e., at multiple intervals with the same/refreshed/additional datasets, or one-off after the study completion or termination) and data-specific (i.e., partial, full, or selected datasets). c) Auditability: Data export functionality should log all activities related to data export in order for study owner to fulfil audit and regulatory requirements, given that such logs store all data export events [when (timestamp), who (user identity – role), how (encrypted/plain text), why (purpose specification and use) and what (data specification)] to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics.

6 Case II: The Technology Adaptation Process – the U-CARE Maturing Phase

This chapter describes the design and evaluation of the U-CARE software system development process during a technology upgrade. The case is a rich illustration of how technological innovations in the surrounding environment affected the design process in an academic research context. The case aims to explore the impact of changes in the environment on sustaining the usefulness of eHealth research software. Section 6.1 explains the design context and problem relevance. The iterative building, intervention and evaluation cycles are presented in Section 6.2. Formalisation of learning is presented in Section 6.3 as a (final) set of design principles which emerged during iterative BIE cycles.

6.1 Problem Formulation

Rapid advancements in technology put strains on software developers. The developers have to face constant changes in user requirements, the organisation, and the environment. User requirements are often hard to define or visualise and can rarely be anticipated to be stable throughout a project. Agile software development is claimed to be more responsive to changes than other traditional methods. Its lifecycle is flexible enough to enable for organisations to respond to constant changes (Beck *et al.*, 2001). In agile development, the software design continuously evolves, which requires regular refactoring. If this is not done, the code will *rot*, to paraphrase Martin (2000). Every time developers change code without refactoring it, rot worsens and spreads. Code rot frustrates them, costs them time, and ultimately shortens the lifespan of useful software.

In recent years, rapid growth has occurred in software development-related technologies (specifically open source, with freedom of use in software development at no cost), for example, availability of reusable components[50]. In-

[50] In computing, a reusable software component adds a specific feature to an existing computer program.

creasingly, web development has embraced the use of open source components (e.g., frameworks[51], libraries[52]) often implementing core functionality, extending system capabilities, and contributing significant resources to such projects (Sojer & Henkel, 2010). These technologies are growing, evolving, and being upgraded at a rapid pace, especially in the web development area. The technology landscape is changing quickly, and as a result, software development needs to be flexible enough to incorporate the latest technologies to ensure sustainable eHealth research software.

Likewise, in the U-CARE research context, the environment is continuously changing due to changes in the existing studies and additions of new research groups and their studies (i.e., associated studies). As a result, the U-CARE software system continuously evolves, expands, and is extended, primarily due to new feature requirements from existing and associated studies. The development team needs to support the usefulness of the U-CARE software system in this evolving design landscape. Furthermore, they must balance productivity gains (i.e., facilitating the clinical researchers' needs) with quality improvements (that are needed due to the complexity and evolution of the system and its environment). Furthermore, the software developers in this eHealth academic research context must develop a system with scarce development resources, junior developers (Master's students), a small budget and tight schedule, high uncertainty (unknown unknowns), strict regulatory compliance, and life-critical consequences (at least for some end-users).

The increasing complexity of the U-CARE software system and the system's steady ageing in the face of rapid technological change made it less responsive over time. The system needed refactoring and technology upgrades to meet the changing needs of the clinical researchers, the U-CARE context, and the external technological environment. At the time of this case, an increasing number of individuals were using their mobile devices to access the internet in Sweden. Mobile internet use provides additional opportunities to offer psychosocial care. Research has supported the use of mobile apps to provide psychological treatment for behavioural health care (Luxton *et al.*, 2011; Cohn *et al.*, 2011; Smedberg & Sandmark, 2012; Rini *et al.*, 2012). Furthermore, research has also revealed that participants are willing to use mobile apps (Weaver *et al.*, 2007; Harrison *et al.*, 2011). Advantages of mobile-adapted self-help programs are not solely restricted to their broader reach, but also encompass increased convenience for the research participants and the opportunity to provide information in an interactive and timely manner. Considering this technological change in the end-users' environment, the ADR team initiated the U-CARE software system adaptation to mobile devices (Figure 15, pt. iii-b), which is explained in detail in the third ADR case

[51] A software framework provides a generic functionality that can be adapted to or changed with additional user-written code, thus used to resolve many things.
[52] A library is a reusable code to resolve just one specific thing.

in Chapter 7. Due to steady ageing and increased complexity in the existing system, the team initiated the technology adaptation process (Figure 15, pt. ii-b).

The problems mentioned above related to the U-CARE software system can be taken as an instance of a class of problems (i.e., *coping with a continuously changing design landscape*) faced when designing any eHealth research software in an academic research context with limited resources. Following the ADR method, Information Systems researchers wanted to consider design principles for the design process that would apply to a class of similar problems. Therefore, the case-specific research question was: *What principles should guide the design process to sustaining the usefulness of the eHealth research software in the continuously changing design landscape in the academic research context?* The next section presents the BIE cycles in this ADR case.

6.2 Building, Intervention and Evaluation Cycles

During the technology adaptation process case, the U-CARE software system went through three BIE cycles: i) *proactive refactoring*, ii) *reactive refactoring*, and iii) *quality assurance*. Proactive refactoring and reactive refactoring led to improvements in the U-CARE software system through refactoring, while quality assurance led to improvements in the design process of the U-CARE software system.

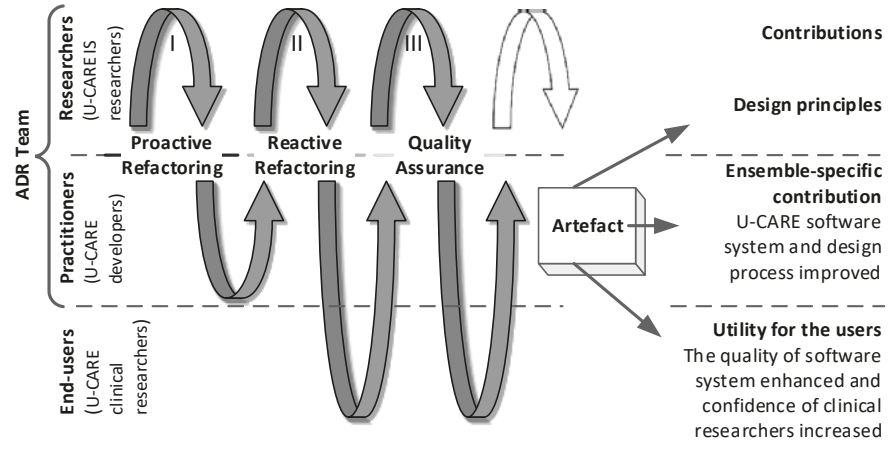


Figure 27. The BIE cycles during the technology adaptation process including contributions and stakeholders involved in the design.

Figure 27 shows the BIE cycles, in which the U-CARE software system and design process were improved, put into the organisational context, and

formatively evaluated to meet the stakeholders' needs. The ADR team consisted of Information Systems researchers and Information Systems practitioners (i.e., software developers) in the U-CARE context. The following sections provide a detailed description of each BIE cycle.

BIE Cycle I

The technology adaptation process started when there was already back-end refactoring underway. Actually, two instances of significant refactoring had been performed in the U-CARE software system before the technology adaptation process case. The first refactoring[53], performed in September 2012 (Figure 15, pt. i-d), was related to the system back-end; the system was restructured from feature-oriented to activity-oriented, to make it a better long-term solution. This refactoring was discussed within the development team in a group programming[54] session in order to disseminate knowledge within the team. The second refactoring, from August 2013 to October 2013 (Figure 15, pt. i-e), was also related to the back-end and aimed for quality improvement. The development team focused on optimising the source code and improving maintainability. The development team also focused on the development process by learning about how they could work as a team, what their knowledge gaps were, their knowledge of the source code, et cetera. Additionally, the development team participated in two workshops about design patterns and testing, respectively. As a consequence, the development team set up a test server to test new features before publishing on a production server[55]. This deployment on a test server allowed quicker identification of software bugs. During this time, additional developers joined the development team. A document about code conventions was created to improve code readability and maintainability. The development process was also improved by adding new functionalities into the product backlog feature of the U-CARE software system (for details see Section 3.5, particularly Figure 11 and Figure 12). Now, a stakeholder who submitted an item (i.e., feature, bug, et cetera) to the product backlog could be notified when the item was completed.

Build

In September 2013, two design workshops (Figure 15, pt. iii-c and iii-d) were held to discuss adaptation of the U-CARE software system for mobile devices.

[53] Also mentioned previously in the data export feature case.

[54] Group programming is very similar to pair programming, except one person is coding, while the others observe, comment, and make suggestions.

[55] A similar practice exists in the industry, where a portion of users are given access to a new version of the software. In U-CARE, only clinical researchers were given access to the latest software version before it was published on a live server. The test application on the test server had the same database as the production application. This has a benefit for developers, as they do not have to populate a test database, but has a downside as changes made during test application usage affect the production database and consequently the live application as well.

The development team assessed that the U-CARE software system required substantive refactoring on the front-end side to be compatible with suitable third-party technologies for mobile adaptation. In this BIE cycle, the third (proactive) refactoring was performed. The team split into two small development teams. One development team kept working on the back-end refactoring in a one code branch. Another team started front-end refactoring in another code branch. For simplicity, the code branches are referred to as b branch and f branch, respectively, in this text. The front-end refactoring required a revision of more than 400 view pages. The HTML markup of the view pages was adapted to Razor (a view engine that needs a specific markup) and HTML5. The .Net framework was upgraded from 4.0 to 4.5 and ASP.Net MVC from 2 to 4. The integrated development environment (IDE) was upgraded from Microsoft Visual Studio 2010 to 2012 with an additional extension in the form of Web Essentials 3.2.

The use of the Razor view engine enhanced the U-CARE software system in terms of malleability, decomposability, and simplicity, through its multiple layouts and its source code cleanness, readability, and maintainability[56]. Also, it allowed unit testable and pre-compiled views, which enabled detection of errors at compile time instead of at run time. Furthermore, it supported partial views, which enabled code reusability and extensibility, and reduced code duplication. Web Essentials facilitated the minimisation of JavaScript (JS), and Cascading Style Sheets (CSS) files to decrease the size of the views. Additionally, multiple JS and CSS files were grouped into bundles. The bundles of files minimised the number of requests to the web server, resulting in increased efficiency and improved performance.

The technology upgrade had a significant impact on the software development process of the development team, for instance through the NuGet[57] package manager for the Microsoft Visual Studio. The package manager enabled the development team to have one central reference list[58] of all useful packages with their version numbers (including both top-level and down-level dependencies). The package manager facilitated for the development team to install or upgrade packages seamlessly in their projects. Packages could be effortlessly moved between developer computers, source control repositories, build servers, and so forth, with the help of just one reference list file. Thus, the need to commit these packages was eliminated, meaning less space in the source control repositories was wasted. The use of the package manager by

[56] Developers have described several advantages of the Razor view engine and technology upgrade in 2015 interviews and later, during the during the mobile adaptation in 2016.
[57] https://www.nuget.org/ [accessed: August 24, 2017] (The package manager had 89,585 unique packages when it was last checked).
[58] The package manager enables for production and consumption of packages from the NuGet central package repository. The repository contains third-party libraries in the form of packages (CSS, JS, and other languages), with their versions and references. The package manager provides the means to restore all referenced packages upon request.

the development team minimised discrepancies across their development environment. Hence, it enhanced the quality of the software development process and indirectly the quality of the U-CARE software system, thanks to consistency in the development, test, and production environments.

Intervention and evaluation

In the case of back-end refactoring, there was decent test coverage[59] through unit tests. However, in the case of front-end refactoring, the UI – consisting of 400+ views in MVC context – had little-to-no test coverage in the front-end. Thus, automatic testing (which would have been ideal in this case) could not be carried out. During the front-end refactoring, there were too many changes in the system, mainly in the HTML syntax of UI. However, the changes were not supposed to affect the UI itself. The development team devised a test approach based on the criterion that *the changed views should interact with users, present information and receive data for further processing in the same manner as existing views did* (Design workshop, 2013). The development team identified that *data collection through questionnaires* was the most critical part of the system, due to the risk of a negative impact on the ongoing research studies as a result of bugs.

The development team selected a pilot (small-scale) research study which used nearly all the U-CARE software system's features and functionalities. The pilot study was designed and run by the clinical researchers as a feasibility study ahead of a future full-scale RCT study. They planned to test the U-CARE software system by going through the selected pilot study steps using test user accounts on both the existing and the refactored version in parallel. The assumption was that if both U-CARE software system versions behaved in the same manner, using the same set of conditions (e.g., participants, study design, data, browser, et cetera), then the front-end refactored code could also be considered reliable for production release. As the existing software system was equipped with a logging function, the development team was able to list the most frequently accessed URLs (i.e., browser requests mapped to controller/actions in MVC) concerning user roles and frequencies. In this way, the development team had a systematic test plan with a list of tasks to be carried out, reported errors for specific tasks, and could later verify corrections for any particular task. The manual process of comparing the two versions was very tedious, time-consuming and error-prone. The development team members also expressed that it was difficult to compare them in two different tabs of a browser. A comparison tool was developed to ease and simplify comparative analysis of the two system versions (see tool-related details in Appendix D.1) (Figure 15, pt. ii-c).

[59] In software testing, *test coverage* refers to a measure used to describe the degree to which the source code of a software is executed when a particular test suite runs.

During the testing workshops, the development team evaluated the system in three ways. First, they performed a comparison of UIs while logged in to both systems using test user accounts. Second, database records were analysed and compared. Third, the database log table entries were analysed. The process was repeated using different user accounts and following different paths in the intervention flow. The screenshots taken in the comparison tool were analysed after every workshop to find errors and remaining URLs that were not yet dealt with. The testing workshops led to the detection and handling of some errors in the source code. Also, during the process, errors were found and fixed.

The comparison tool made the evaluation very simple for the development team, as they could go through all web pages of the existing and refactored software system, take screenshots, and compare differences. It helped in finding tiny errors in the layout or content, even if only a single pixel had changed. For example, the translation of a few web pages was not working; though the system had been tested many times, the development team had not been able to spot the difference. The comparison tool spotted this and many other bugs efficiently, even when things look similar at first glance. The translation bug found in the existing system related to the unique key generation of a translation phrase when a partial view was loaded in the parent view. This error emanated from the back-end logic layer of the translation feature, so the development team fixed the translation feature. Another related bug in the existing system was also fixed by returning partial views from the controller, as it was incorrectly returning partial views as main views. This was not a problem for the old web form view engine, but for the Razor view engine, a correct specification as either a partial or a full view was required.

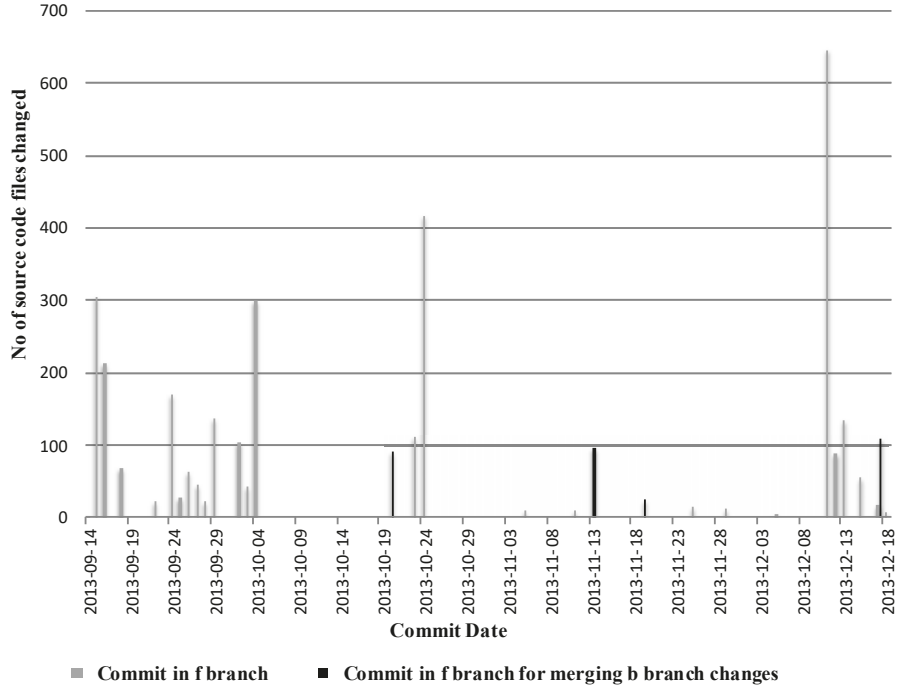


Figure 28. Changes in source code files committed during front-end refactoring.

In Figure 28, it can be seen that testing and evaluation resulted in significant code changes in December 2013, almost double those in the original development from September to November 2013. The back-end and front-end refactoring code branches are denoted *b* and *f*, respectively. After testing workshops, the evaluation results of critical parts of the system were considered to be satisfactory. The development team agreed to release the latest version of the software system to the clinical researchers for beta testing, and that the beta version would go into production when it showed sufficient stability.

The development team kept working on the b code branch, although back-end refactoring ended in October 2013. After testing some releases on the test server, the b branch with the refactored code was released on a production server for the first time in January 2014. This was the first production release since the refactoring started in August 2013. Later and fewer releases (i.e., 06-2013, 08-2013, 01-2014 and 02-2014) were one of the significant consequences of proactive refactoring. The back-end refactoring also impacted on the front-end refactoring. More developers were working on the b code branch; as a result, there were more changes in the b branch compared with the front-end refactoring code branch f. Often, the version control system (e.g., Subversion and Git) facilitated the merging of the two code branches, but since there were significant changes in the structure and code of the source files, merging could not be done automatically. So, each change in the b code

branch was manually rewritten in the f branch. Figure 28 shows the code committed in the f branch and code committed due to merging the b branch changes with the f branch. This led to more work for the developers who worked with the f branch when they had to merge the changes with their branch.

On one hand, this extra work put pressure on the development team to merge both teams into one and to start working in a single branch. On the other hand, there was a pressure to release the latest version of the production server. However, this could not be done until the system had been thoroughly tested. So, the f branch refactoring release was delayed multiple times and needed to keep the two code branches continuously synchronised. After nearly three major and one minor production release of the b branch, the developers finally switched to the f branch completely. In March 2014, the f branch was released in a production environment for the first time. The technology upgrade and adaptation to the Razor view engine made the U-CARE software system more compatible with other devices than just personal computers (PCs), such as mobile phones and tablets. The Razor view engine also supported unit testing of UI, which was better for any future upgrades. It was also much better in terms of the maintainability of the U-CARE software system, as it had improved readability and organised the source code syntax, in a much more simple, cleaner and better way.

Reflection and learning

During the proactive refactoring, the development team recognised the importance of considering the ecology of artefacts in order to sustain the existing U-CARE software system over time. The development team considered boundary resources, such as development tools, frameworks, et cetera in the ecology of the U-CARE software system. They also realised that they needed to continuously keep the U-CARE software system in sync with such boundary resources to keep it aligned with the changing design landscape. For example, the U-CARE software system's compliance with new versions of the framework made it easier to utilise new capabilities that supported a better user experience for the users of the U-CARE software system. As a result, the development team was able to comply with new requirements from the U-CARE stakeholders and sustain the usefulness of the U-CARE software system in the emerging technological landscape. The development team had to consider trade-offs such as resource efficiency and alignment with existing technology, as well as the utility of the software system. These trade-offs influenced the design process and the quality of the software system over time.

The U-CARE stakeholders were accustomed to an iterative process of providing feedback on the features. During the technology upgrade refactoring, a change of the existing release process was introduced in the form of alpha-beta releases in a test environment before production release. The introduction of a test server facilitated this process further by enabling early alpha-

beta releases on the test server and subsequently facilitated the receipt of early feedback from the stakeholders. The development team also realised the importance of test coverage and started writing more test code, which was reflected on by the team leader as follows:

> The culture of the team is changed, now more developers have started writing test code. (ISR-1, 2014, IT meeting minutes)

During refactoring, the product backlog was frozen. The development team support was only allocated for urgent situations in ongoing studies in the production system. Although the product owner kept adding new items to the product backlog, there were no more features implemented during refactoring and only limited development was visible to U-CARE stakeholders. Thus, it was not only hard to convince the U-CARE management to allocate resources for refactoring, but also to show the long-term value of the time and resources invested in refactoring. The development team was under pressure, as they were not able to show any progress during refactoring. The refactoring was challenging, given the limited resources and that maintenance of existing system functionality and addition of more features were ongoing as well. The first draft of the design principles that considers the technological-ecological adaptation was phrased based on the learnings in this cycle (see Table 21).

Table 21. Design principles for sustaining the usefulness of eHealth research software (version 1)

Design principle	Specification
The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by the software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging technological landscape, in order to promote the research software's fitness to the changing design landscape.

The next BIE cycle (Figure 15, pt. ii-f) describes the reactive refactoring in the U-CARE software system.

BIE Cycle II

Build

After proactive refactoring, there were some errors encountered once the software system was released on the production server. The system failures were problematic, as they appeared seemingly at random. Here is an example report of the system failure:

> Today, the U-CARE production server was once again very slow. I did some checks on the server while the problem occurred, and it turns out that the

> w3wp.exe process (i.e., the host process for IIS) is consuming nearly all CPU time (~98%). Restarting the web server from the management view (temporarily) solved the problem.
>
> Some conclusions: (1) The problem does not reside on the database side. (2) We need to understand why our application builds up to consume so much server time. I have got a feeling that the problem may be related to all the Ajax requests made by clients (checking for chat messages, new instant messages, et cetera). If many people leave their browsers on while logged in, we will have a fair number of requests piling up.
>
> I think we need to make a collaborative effort to identify and remove this problem. Perhaps some change in the server settings (quicker timeout/clean-up of old requests?) could fix it, in concert with changes to the code to minimise the pressure on the server. (ISR-#, 2014, Email)

One reason for the problems was that there was less test coverage of the code, leaving errors unchecked in the system. Also, no code review was performed. There were some changes in the business logic just before moving to the Razor view engine. Most times when the system went live, the development team focused on the Razor view engine as a source of performance errors. For example, the system shut down at night or was not responsive. Using a trial and error method, the development team changed some code. For example, action rendering was changed to partial rendering, which led to other problems such as that the translation and logs entries being changed, as they were specific to the URL context. This was also due to differences in the Razor view engine's rendering of different parts of the system UI. It was later discovered that there was a logic error in the event service that made the database too busy to allow the core system to access the database for login et cetera.

The proactive refactoring (Figure 15, pt. ii-b) also led to problems in the system log mechanism. This happened in part due to the technology upgrade when intermediate calls (a kind of user controls, technically ASP.Net MVC RenderAction) to a parent web page (which was composed from many user controls – technically a view inside another view) bypassed the controller; hence, they were not subjected to the log mechanism of the U-CARE software system. This log was the basis for accountability and traceability of the system events, in addition to its other areas of use. The reminders were also not triggering correctly, due to both incorrect configuration and changes in the design of the studies during a six-month period (January to June 2014) (2014, IT meeting minutes).

Because of these system failures, the development team initiated reactive refactoring (Figure 15, pt. ii-f) of the source code. The development team was continuously engaged in *fire-fighting*[60]. This tied up resources that were needed for designing and adding more functionality. Detecting the reasons for

[60] In software development, *fire-fighting* is an emergency allocation of resources, required to deal with an unforeseen problem, for example, assigning extra programmers to fix coding bugs that are discovered close to a product's release date.

errors in the system drained resources and led to many unanticipated refactoring measures to improve code clarity and page rendering performance. During this fire-fighting, the development team got new members. Still, it was hard for new team members to understand the U-CARE software system and quickly resolve problems. The new team members were not familiar with the existing code base and domain knowledge. Additionally, to strengthen the development team, one of the Information Systems researchers started working as a full-time developer for a three-month period. The Information Systems researcher's main task was to fix the bugs, as he was familiar with the existing code base and domain knowledge (business rules and process).

Intervention and evaluation

The development team started to monitor the U-CARE software system to confirm that it was running and doing what it was supposed to. For example, 24/7 monitoring and diagnostics of the web server were set up through a third-party application, LeanSentry[61]. This application helped in understanding how the U-CARE software system behaved on the web server. The U-CARE software system logged all errors in an exception log. The development team started monitoring this log for error analysis. Performance testing was also performed indirectly by analysing and comparing average code execution times.

The technology adaptation process resulted in a change in software development tools and practices. There were changes in the steps during the development and deployment of the system, such as the addition of new tools like the NuGet package manager, for addition and upgrade of library packages, and the Microsoft ASP.NET Web Optimisation Framework, for bundling/configuration and minification for CSS and JS. On many occasions, the development team forgot that the above changes in the development process had been introduced. Sometimes they forgot to install the frameworks and sometimes they forgot to deploy the minified files. These inconsistencies led to errors in the production environment, as the system in the development environment behaved differently due to non-minimisation JS vs. minimisation JS in the production environment. This indicates that it took some time for the development team to understand the new tools.

There were also improvements to the system that came with fixing these bugs, for example, optimisation of the U-CARE software system, improved performance, elimination of redundant views, and placement of business logic in the business layer separate from the presentation layer (i.e., views). The Razor view engine allowed the rendering of the same view as a full and partial view with the same contents in it. During the technology adaptation process, one obvious result was that the development team did not have a suitable mechanism for UI testing. The comparison tool was good, but required that

[61] https://www.leansentry.com/ [accessed: March 10, 2014].

the development team manually went through the URLs one by one, took screenshots, and reviewed them. To improve this process, a design workshop to introduce Selenium[62] was organised (Figure 15, pt. ii-e). During the UI testing workshop, the functionality of Selenium was demonstrated, and the development team members discussed how they could use it to improve testing in their development process. The open discussion highlighted the following fundamental design implications, design decisions, and design actions regarding

a) the functionality and use of Selenium:

> It simulates keyboard events […] here we can make assertions and check that the page is rendered properly […] we can access the entire DOM [Document Object Model[63]] in the rendered page and see for instance if a specific div has the class hidden or any test of the actual result on the user side […] currently, we can only see what is returned by the controller […] we can export the clicks and actions […] Selenium looks simple to me […].

b) assessing Selenium relative to current testing tools and routines:

> The main difference here is that the Selenium framework requires us to have an actual web server running and it makes the request properly using an engine for a web browser. Then we make assertions based on what is a return to that web browser. What we are doing right now in the unit test is that we simulated the entire web context. There is no real web server involved, and there is no real web client involved, it is just a simulation of these things […].

c) the reflection on Selenium's usefulness in the U-CARE context:

> What we gain from this [Selenium] is that we have [Java]Scripts in many pages and this will also indicate if the scripts work or not. We can make assertions on how the page rendered rather than just [what is returned by the controller] […] since we can combine things here, we can use the Selenium engine to make some calls. But if we can do this in test mode, then we could also make assertions about the state of the web server and messages sent and so forth […] that combination would be very powerful. That would be real integration testing, showing a lot of things and also for different web browsers […] so it seems very powerful.

d) proposed changes/improvements in testing:

---

[62] Selenium is a suite of tools to automate web browsers across many platforms. It can be controlled by many programming languages and testing frameworks. Selenium WebDriver and Selenium IDE, parts of Selenium, were necessary packages for developers. Selenium WebDriver helps to create robust, browser-based regression automation tests and distribute scripts across many environments. Selenium IDE is a Firefox add-on that can be used for record-and-export interactions with the browser. http://www.seleniumhq.org/ [accessed: February 13, 2014].

[63] https://en.wikipedia.org/wiki/Document_Object_Model [accessed: August 08, 2017].

The test project is quite messy […] we should have at least two projects. One in which we have the real unit tests […] [where we call] these isolated functions and do real unit testing, and then we have integration testing maybe in there both the current way we do it, and this could be [combined] […]

What we would need to do would be to run these UI tests against a database that is clean and configured before each test and then the test should run. So basically, when we create a test we need to set up environments and then run the tests […]

[We should] set up the database wisely […] we should create at least one study, and we should add some basic data that allows us to record, walking through each function basically or based on each role we should do things […] so log in as a research participant, fill in the baseline, [get] randomised, [and] then the module should be activated, we submit a homework [assignment]. Then we log in as a therapist and provide feedback. Then you know to test this entire chain it would not take that much time and if we can run that every time before we publish than we have come far […].

e) the usefulness of the improved testing process:

It could be a simulation of the entire system […] this could also help the new developer to see the whole system, how it is working […] we can simulate an error and put a breakpoint to understand what is going on at a particular point in time or in a sequence of events […].

f) identification of possible challenges in the implementation:

The problem is if you are recording things then [a presumption would be] that the database base has a certain content when you start using it, so the next time you run that script you might get a different result for instance or the script might even do things that change the database. So I think we need – if we want to use it in a systematic way – we need to make sure that we set the database in a certain state before recording the tests and before each test it should be reset to that state […].

g) improvement in running and writing tests:

I agree that the sooner we get this thing going, the better it would be, because it would really help us to quickly create new tests every time we have a new feature or a bug or something. Then we can do this straight away and then just get the code out of it to make it possible to repeat the test basically. The optimal way if we find a bug would be to reproduce the bug and record this as a test using Selenium and then we have the test. We know this should pass, and then we change the code and then when the test passes we know we have solved it. Then the next [test] could be to use some of these combinations to simulate through different browsers and different input values on the same test […]

When we are about to test something, we have to consider which type of test is the most suitable for this particular thing and sometimes it might be a mock unit test and sometimes [it might be to] simulate a typical user behaviour, flow or how people do things in the [system] […].

The above workshop, relating to the testing process, led to the addition of Selenium tests in the existing testing framework. The addition of Selenium tests made the U-CARE software system more testable and resulted in an increased test coverage.

Reflection and learning

During this trial and error process, the development team realised that they needed an improved design process to solve problems in the U-CARE software system. Continuous upskilling of the development team was needed. The system refactoring impacted negatively on the team members' experiences of the existing system and technologies, in particular for team members new to the system. The results indicated that the development team was frustrated during this BIE cycle:

> We wait until a problem occurs. We drive our development based on problems. Dev-#, 2014, Observation)
>
> We do not follow the sprint plan. (Dev-#, 2014, Observation)
>
> Every time we publish it is worse than before. (Dev-#, 2014, Observation)
>
> Whenever the development team publishes on the production server, there are always some bugs, and that becomes a priority to solve, [rather] than following the sprint plan. [There] seems [to be] a problem with testing, as testing is not performed [before] release of the system to production and also there is no beta testing for the last few releases. (ISR-#, 2014, Research log)
>
> There is no one responsible for anything. There is a need for a leader to lead the project and be full time and make decisions as and when required. (Dev-#, 2014, Research log)

The clinical researchers were not as engaged in the beta testing during technology upgrade as they were during initial system design. There was no incentive and motivation for the clinical researchers to test the system again. They were busy with their research and not very interested in testing the system. Also, it was difficult for them to identify errors. Consequently, the system failure after the refactoring made the clinical researchers and the U-CARE management dissatisfied, as they became less confident in the system.

The development team had mainly been occupied with bugs. The bugs had been reported by stakeholders on a continuous basis, with the implication that the development resources were allocated to correcting problems rather than preventing them in the first place. However, this did have one positive result: the development team became more conscious of taking too big steps and began testing the changes in the system thoroughly before releasing them into production. Though tests had been conducted, the fact that they were not run

on a daily basis meant that as the code changed, new bugs could occur and, as the tests were not automated, all changes were not always tested. With this in mind, a decision was made to create a batch script to automate the execution of tests (a.k.a., continuous testing[64]). However, the continuous testing was the result of the learnings and implementations of multiple tools over a long period of time.

The development team also decided to increase the test coverage. This was achieved by adding the Selenium recorded test[65]. However, the path to introducing Selenium was not straightforward. Many workshops were held by the development team to discuss how to utilise the existing testing framework better and how to integrate Selenium. Selenium IDE was used to create quick bug reproduction scripts, which were used in the tests. In a way, these scripts simulated research participant and system behaviours. Scripts replicating known bugs were used to prevent new occurrences of these bugs. The development team decided to focus on the most critical components first, for example, the core need of the researchers that research participants would be able to login:

> To get the research participants to start working on the [system] it is important that the first impression be good. That is, when logging in and answering questionnaires for the first time and when starting an intervention. This must be user-friendly and working. (ISR-1, 2014, IT meeting minutes)

The development team also decided to use ReSharper[66], a Visual Studio Extension, for static code inspection. ReSharper helped in cleaning and fixing the source code. ReSharper provided an excellent facility for code refactoring. Every new version of Microsoft Visual Studio IDE also improved its features related to refactoring. However, utilising such refactoring features required continuous upgrades of IDE.

Based on the lessons learned during this cycle, the design principles were revised. Table 22 lists the revised design principles. Changes (if any) from the previous version are highlighted in bold.

[64] Continuous testing is the process of executing automated tests to obtain immediate feedback on the defects associated with a software release candidate. This reduces the time and effort that must be spent finding and fixing defects. As a result, it increases the velocity of the development team and the frequency at which quality software is delivered. Continuous testing is part of continuous integration and continuous delivery.
[65] Selenium IDE is Record/playback tool. The tool allows the recording of test scripts that can be exported to a high-level language (e.g., C#).
[66] https://www.jetbrains.com/resharper/ [accessed: March 24, 2013].

Table 22. Design principles for sustaining the usefulness of eHealth research software (version 2)

Design principle	Specification
The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging technological landscape, in order to promote fitness to the changing design landscape, **given that the development process is supported by adequate test coverage, automated and continuous/frequent test-deliver-feedback practices, a set of appropriate supporting tools, and continuous upskilling of the development team.**

BIE Cycle III

Build

In this BIE cycle, the development team reflected on their development process and organisation together with the U-CARE management and the product owner. The result of the reflections was a realisation that the Scrum-based development process needed to be followed and improved (Figure 15, pt. ii-h). Also, there was a need to go through the software system more thoroughly as a team. The development team organised a three-day workshop (Figure 15, pt. ii-i) to assess the limitations of the current software development process and to devise a new plan with improved quality control. They focused on: 1) where we are; 2) how we got here; and 3) where we want to be.

During this workshop, the development team discussed the existing design process, the consequences of refactoring, and technical debt. The development team believed that despite the quality assurance efforts in the design process (during the previous BIE cycles), there had been quality problems in the U-CARE software system. The testing activities had not been given a continuous high priority. The development team had been working under high pressure due to a continuous influx of new requirements from the U-CARE stakeholders, leading to a reduced focus on quality. The bugs reached a peak in early Autumn 2014, causing problems with impact on the U-CARE context. Bugs, however, had been reported continuously by the clinical researchers, with the implication that development resources were allocated to correcting problems rather than preventing them in the first place. This phenomenon is referred to as *technical debt*. An analysis was undertaken to see how much technical debt the team had been dealing with over the preceding year. The analysis revealed the problem of the *technical debt*, i.e., at any given time there were high priority bugs to fix that forced the development team to work on correcting errors made in the past (technical debt) rather than on building new features, making improvements to existing features, et cetera. It turned out that technical debt had not been decreasing; rather, it had increased slightly.

The workshop focused on three themes: i) core business activities in U-CARE; ii) database issues; and iii) a new testing strategy. The core business activities in U-CARE served as the starting point for the new testing strategy. The product owner helped the development team to identify and focus on core business activities in U-CARE. The DBMS was an important design aspect of the U-CARE software system. The development team came up with a set of decisions to use features of the DBMS to simplify the design of the U-CARE software system and to improve the overall software performance. The development team identified and assessed four approaches that were in use at the time: System tests from the user perspective, based on Selenium, were (1) recorded tests (RecTest) and (2) programmed tests (ProTest); (3) Generic tests (GenTest) were unit and integration tests based on web context simulation; (4) Data, database and schema consistency tests (DbTest). RecTests were further categorised as core business processes, issue verification, and study-specific tests. ProTest made use of an evolving tests to produce combinatorial tests automatically. GenTest programmed tests that tried out functionality up to the controller level. DbTests were tests based on SQL queries and stored procedures with the purpose of checking the integrity of the database. The main points in the new testing strategy, which was developed in the workshop, were to:

1. Automate build-and-test [code] every night before publication on beta and production servers. Test results must be reviewed before publication, and all tests must be passed.
2. [Draw up] an improved and explicit *definition of done*[67] that included: **a)** The developer having interacted with stakeholders to understand the requirements; **b)** Backlog task description being updated to include an understandable description of the requirements; **c)** The code being implemented; **d)** The feature being *sufficiently tested*, with a note in the sprint backlog stating 'what was done in the testing while solving the task'; **e)** After an issue being marked as done, it would be annotated; and **f)** After publication, a revision report would be generated (based on the annotations of completed issues) and sent to the product owner for further distribution.
3. [Define] *sufficiently tested* at the task level. **i)** A basic task (complexity point < 5) is sufficiently tested when existing test code is considered comprehensive enough to cover the changes made to the code, or existing test code has been complemented with new tests to achieve coverage of the code written to solve the task, **ii)** A

[67] The *definition of done* is a simple list of activities (writing code, coding comments, unit testing, integration testing, design documents, et cetera) that add demonstrable value to the product. Highlighting value-adding steps allows the team to focus on what must be completed in order to build software.

complex task (5 ≤ complexity points ≤ 8) is sufficiently tested when the testing requirements for a basic task are fulfilled, and another developer has reviewed the code changes.

4. [Ensure] human validation of data – in interaction with the stakeholders, the team should make core business indicators visible to the [clinical researchers] so that they can assess if activity in the [software system] is reasonable.

The workshop also resulted in design decisions (see the full list in Appendix D.2, which includes details about rationale, requirements, and follow-up). The most significant decision was that *the software developers in U-CARE would work half-time with proactive practices, i.e., testing, refactoring and documentation*. This decision was reflected in the *definition of done* as well. No backlog item, whether it was a bug fix or new feature, was considered complete until: a) Tests have been written as part of the resolving backlog item; b) In-code documentation was adequately produced upon resolving the backlog item. The new *definition of done* required the development team to take on half of the usual complexity points per sprint for testing. This allowed the development team to work half-time with testing and documentation.

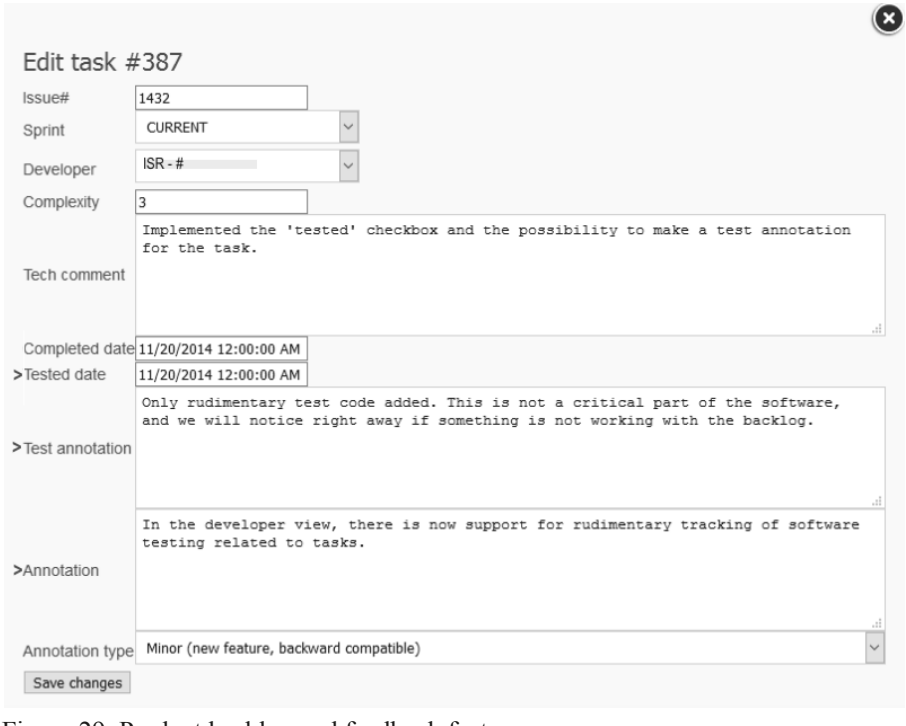


Figure 29. Product backlog and feedback feature.

The new *definition of done* was implemented in the development process by enhancing the existing product backlog and feedback feature of the U-CARE software system (explained in detail in Section 3.5). New fields were added, for example, test date, test annotation, and annotation (see Figure 29).

Intervention and evaluation

The development team organised multiple workshops (Figure 15, pt. ii-j and pt. ii-k) to evaluate and further improve the software development process with a focus on test coverage and quality assurance (see details about decisions in the workshop in Appendices D.3 and D.4). The development team complied with the new testing strategy by including the date of the test with a brief description and adding annotations regarding testing for all completed tasks. The development team used the backlog feature to follow up on and report the testing strategy. The development team also improved the continuous integration by using TeamCity[68] for running automated builds and tests. The TeamCity was better than batch script running, as it supported the web-based administration of the continuous integration process. All tests were executed at least once a day and the development team received a daily build report (i.e., the number of passed and failed tests) to validate the quality of existing and new code. This enabled a rigorous quality control ensuring that any changes to the current system did not introduce unintended errors. The development team also configured automated deployment of the system on the web server. Automated deployment not only eliminated the possibility of errors being introduced through manual deployment on the web server, but also made the process simple and efficient.

The testing framework was further improved to facilitate the design and construction of advanced tests of the U-CARE software system. The development team added recorded tests for core processes related explicitly to research participants. The development team added a recorded test for every bug identified. They also added many recorded tests for every research study running on the U-CARE system. The study-specific recorded tests simulated a research participant completing observation points one by one while going through the CBT and interacting with the system. Selenium-recorded tests enabled system integration testing of the U-CARE system across the full technology stack. The development team also improved and coded additional programmed tests (ProTest), generic tests (GenTest), and database consistency tests (DbTest). The development team started generating a bi-weekly system status report for all stakeholders, particularly the clinical researchers. The report contained core indicators to enable the clinical researchers to assess the activity of the system and make sure the system was functioning as it should. These reports resulted in further strengthened testing with stakeholder-centric human data validation.

[68] https://www.jetbrains.com/teamcity/ [accessed: February 7, 2015].

The development team also started working half-time with proactive quality assurance tasks, i.e., testing, refactoring, and documentation. This was achieved by considering testing, refactoring, and documentation in the story points while estimating the complexity involved in the user stories. The focus on quality assurance resulted in stabilisation of the system, while new features were developed at a slower pace than before. The increased focus and priority on refactoring needs led to the identification of a need for adaptation of the system for mobile devices (see case III). Also, the real efforts made by the development team in increasing the test coverage of the system were made visible in the subsequent successful extension of the artefact, as presented in case III.

Reflection and learning

The main challenge emphasised by the development team during the third BIE cycle was that the software system needed to evolve in response to the changes in the technical landscape, as well as in the U-CARE context. The development team communicated the difficulties they had experienced during the proactive refactoring and the U-CARE management allocated development time for such refactoring. The development team learned that reducing the level of testing led to an increase in unplanned development which was time-consuming. Also, as seen in this case, fixing the problem could cost more than it would have cost to prevent it. It was also difficult for new developers to understand and use the existing code without documentation. The documented and refactored code let the development team build new features faster and led to a potential velocity increase.

Development of new features was more interesting to the stakeholders, especially as the internal software quality was not visible to them. This made it hard to allocate resources or prioritise quality-related issues such as refactoring, increasing test coverage, and technology upgrades. The stakeholders learned that they had to wait with development of new features, to resolve the technical debt. They also agreed to provide resources for proactive software development practices, such as refactoring, documentation, et cetera. The *definition of done* forced a structure onto the development team ensuring that the development tasks were indeed completed, not only regarding functionality, but concerning quality as well. Based on the learnings from this BIE cycle, the design principles were revised (see Table 23) (Figure 15, pt. ii-l).

Table 23. Design principles for sustaining the usefulness of eHealth research software (version 3)

Design principle	Specification
The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging technological landscape, in order to promote fitness to the changing design landscape, given that the development process is supported by adequate test coverage, automated and continuous/frequent test-deliver-feedback **development** practices, a set of **appropriate tools**, and continuous upskilling of the development team.
The principle of embracing proactive practices	**The software developers of eHealth research software should embrace proactive practices in order to improve code readability, extensibility, testability, simplicity, and potential velocity increase, given that resources (time, money, and attention) are allocated for such practices.**

Artefact Use Over Time and Learning

The U-CARE software system evolved from the *design and development* of a system to offer psychosocial care in a specific research setting to a *research software system as a service* for associated studies and ultimately to a *health care provider* in psychosocial care. The system was initially designed for three RCTs, and up to the start of 2019, it handled at least 16 studies with approximately 4,000 research participants. The source code volume increased as well, and became divided into multiple layers. The evolution of the system resulted in a move of the existing servers, which had been placed at different locations, to a centralised and dedicated infrastructure at Uppsala University.

The technological landscape also changed during the U-CARE software system evolution, for example, Microsoft Visual Studio 2010 evolved to 2019, HTML 4 to HTML5.1, CSS 2.1 to CSS 3, ASP.Net 4.0 to 4.7 and .NET Core 2.0, and ASP.Net MVC 2 to 6. JS frameworks evolved for a more interactive web experience and implementation of advanced design framework at the client side (i.e., Angular, Backbone.js, et cetera), jQuery 1.7 evolved to 3.2, et cetera. Other technologies that changed included hardware, operating systems, platforms, and client software, like browsers.

The above changes have severe implications for Information Systems practitioners and researchers in designing the U-CARE software system, as substantial functionality of the software system was dependent on reusable components (e.g., libraries) and adaptation of third-party software assets (e.g., frameworks). The technologies that were used, whether commercial or free, were changing continuously. As these were multiple interdependent technologies in a hierarchy (Xu *et al.*, 2010), they could not be treated as singular and independent of each other. These technologies co-evolved in an endless reciprocal cycle – in which changes in one technology set the stage for changes in

others and vice versa (Magnusson & Bygstad, 2014). There is a growing trend to make the latest versions of technologies compatible with each other, so as to make it is easy to use the latest versions of technologies, particularly in web-based software development (de Bayser *et al.*, 2015). Such a trend can be seen in the NuGet ecosystem[69] on the Microsoft development platform. However, as the U-CARE software system ages, its source code and the libraries it depends on are becoming increasingly hard to maintain. This case showed that there was a need to constantly keep track of the updates, update frequency, major improvements, obsolete functionalities, other alternatives, et cetera, in order to understand and maintain the involved technologies. One way this can be achieved by relying on tools and package managers "to check that the right versions of required packages are installed and install or upgrade them if they are not" (Taschuk & Wilson, 2017). Lightweight virtualisation containers like Docker[70] make technology adaptation process easier as multiple combinations of technologies can be tested on a virtual machine or container system and development team can see what breaks and discover dependencies (Taschuk & Wilson, 2017). Also, virtual machine or container system, which contains a copy of software system, data and the environment – everything needed to run the software, can be used to have consistent development (even production) envirnoment among the entire team, testing, deployment, and replication a system stated at a later date.

6.3 Formalisation of Learning

Technology has and will continue to change the landscape of the software industry. Software developers need to frequently adapt, adopt, and shape the technology available to them, based on their existing development practices, to ease the design and development of systems and satisfy end-users' needs. The case showed that malleability, decomposability, simplicity, and accountability are the key quality characteristics for sustaining the usefulness of eHealth research software in the academic research context. However, sustaining such quality characteristics is a challenge in the software development process due to limited resources in an academic research context. Advanced technical knowledge and expertise are significant for effective use of agile practices (Senapathi & Srinivasan, 2012). The case resulted in improving the design process by augmenting the *definition of done* with testing and allocation of fifty per cent of the development team's time in each sprint to proactive practices such as testing, refactoring, and documentation. Evaluation of the design process was found to be equally important as evaluation of the design

[69] https://docs.microsoft.com/en-us/nuget/policies/ecosystem [accessed: January 18, 2018].
[70] https://www.docker.com/ [accessed: January 18, 2018].

artefact (Alturki *et al.*, 2013). Adequate support from management was necessary to enable the development team to gain the skills required. The learnings from this case have been articulated and formalised as design principles in the following table.

Table 24. Design principles for sustaining the usefulness of eHealth research software

Design principle	**Specification**
The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging technological landscape, in order to promote fitness to the changing design landscape, given that the development process is supported by adequate test coverage, automated and continuous/frequent test-deliver-feedback development practices, a set of appropriate tools, and continuous upskilling of the development team.
The principle of embracing proactive practices	The software developers of eHealth research software should embrace proactive practices in order to improve code readability, extensibility, testability, simplicity, and potential velocity increase, given that resources (time, money, and attention) are allocated for such practices.

7 Case III: Extending the Artefact – the U-CARE Mature Phase

This chapter describes the design and evaluation of the U-CARE software system adaptation to mobile devices (referred to as the mobile adaptation). In this ADR case, the main focus was on the research participants. Some research participants accessed the U-CARE software system through mobile devices, and the system was not designed for this (i.e., not compatible with mobile devices). Section 7.1 explains the design context and problem relevance. The iterative building, intervention and evaluation cycles are presented in Section 7.2. Formalisation of learning is presented in Section 7.3 as a (final) set of design principles which emerged during iterative BIE cycles.

7.1 Problem Formulation

The internet is one of the vital components in providing psychosocial care and support (Lochan, 2012) and mobile internet adoption is increasing across the world. A growing number of individuals use their mobile devices (which afford advanced computing and internet connectivity) to access the internet. In Sweden, there are 14.2 million mobile broadband subscriptions, and there has been a sharp increase in mobile surfing[71], more than doubling since 2014. This remarkable growth in mobile internet use provides additional opportunities to offer psychosocial care. The usage of mobile applications (or *apps*) has grown dramatically (Barak & Grohol, 2011). Research has supported the use of mobile apps to provide psychological treatment for behavioural health care (Luxton *et al.*, 2011; Cohn *et al.*, 2011; Smedberg & Sandmark, 2012; Rini *et al.*, 2012). Furthermore, research has also revealed that participants are willing to use mobile apps (Weaver *et al.*, 2007; Harrison *et al.*, 2011). Advantages of mobile self-help programs can be seen not only in their broader reach, but also in the increasing convenience for the research participants, who can get/access information in an interactive and timely manner.

Systematic reviews of mobile health suggest the following: there is a striking difference in the number of commercial apps compared with the small

[71] http://statistik.pts.se/, Swedish telecommunications market first six months 2018, report published on 2018-11-19 [accessed: March, 13, 2019].

number of reliable, evidence-based apps (Wac, 2012; Donker *et al.*, 2013; Bastawrous & Armstrong, 2013); there is a limited number of apps for health behaviour change interventions (Clough & Casey, 2011; Free, Phillips, Galli, *et al.*, 2013; Bastawrous & Armstrong, 2013); there is limited use of mobile technologies for mental health (e.g., use of SMS service only) (Clough & Casey, 2011; Gurman *et al.*, 2012; Free, Phillips, Galli, *et al.*, 2013); industry regulation and scientific rigour (evidence base) should be considered (Wac, 2012; Donker *et al.*, 2013); and further research and development are called for (Clough & Casey, 2011; Gurman *et al.*, 2012; Donker *et al.*, 2013; Free, Phillips, Watson, *et al.*, 2013; Free, Phillips, Galli, *et al.*, 2013; Bastawrous & Armstrong, 2013). Little or no evidence has been found regarding therapist-guided mobile applications, even though guided approaches (that combine structured self-help material with the vital role of a therapist who provides support) have been found to be superior to unguided online treatment (Andersson, 2009). Although the future of internet-supported psychotherapy appears bright, there is an essential need for additional research to examine whether therapeutic goals are being met and that interventions are optimised for delivered through mobile devices.

According to the futuristic view of Alcañiz *et al.* (2009), an e-therapy system should be based on *ubiquitous computing*[72] for using the system in any location, at any time, and on any of several technological platforms. Miller (2012), also a proponent of smartphone use in psychology research, stated that a smartphone could collect vast amounts of ecologically valid data, easily and quickly, from large global samples, which could transform behavioural research even more profoundly than PCs and brain imaging have done. Internet interventions will likely follow the move towards mobile internet access, which ultimately will allow for greater patient access (Mewton *et al.*, 2014). Innovative use of information technology for interventions holds tremendous potential for health care, but developing a comprehensive software system that adapts to technological advancements and innovative ideas, such as mobile devices, carries with it legal, ethical, privacy, security, and practical hazards and problems.

Dadgar *et al.* (2013) in their literature review and analysis, concluded that although the health care industry has begun using mobile health, and patient care could be significantly transformed from this, research in this area is sparse and at its initial stages. They call on Information Systems researchers to design and conduct research in the mobile health area (Dadgar *et al.*, 2013). Sørensen & Landau (2014) stated that the Information Systems field had not established a significant response to the challenge of mobile information and communication technologies (mobile ICT). What they suggested was that the

[72] Ubiquitous computing is a concept in software engineering and computer science which means that computing can be performed anytime and anywhere, (https://en.wikipedia.org/wiki/Ubiquitous_computing) [accessed: September 4, 2014].

Information Systems field should respond in an agile manner to these emerging socio-technical phenomena by applying a qualitative, explorative approach and incorporating mobile ICT into the mainstream academic discussions.

Mobile devices are an essentially personal possessions, which most people carry at all times, and psychosocial care self-help applications can tap into the common habit of playing with a device during moments of free time (Nylander *et al.*, 2009). Effective psychosocial care depends on how patients collaborate with their caregivers to manage their care. The patients need not only comply with their psychological treatment; they also need to adopt a lifestyle behaviour that optimises their care. Online psychosocial care requires patients to access and use the platform for a certain amount of time to produce the desired outcome or effect. The convenience of access and therapist interaction influences online treatment satisfaction. Online psychosocial care can be provided using several technological platforms, including desktop, web, and mobile applications. The advantages of mobile applications include their ease of access, increasingly widespread use, and better user engagement. Mobile applications enable patient involvement and the provision of ubiquitous and instant feedback to realise new behaviours and sustain desired performance. The mobile applications can help health care professionals to interact with patients in a timely manner, which can lead to treatment satisfaction.

In 2014, the U-CARE software system provided research participants access through desktop only. There were many compatibility issues when the U-CARE software system was accessed on mobile devices. Among other things, videos were not compatible with iOS devices. Facilitating access through mobile devices, like smartphones and tablets, could engage research participants, specifically young people, who already used such devices for a significant part of their activities. Psychosocial care could easily be delivered through the internet. As more people began accessing the internet through mobile devices, adapting the U-CARE system to mobile devices was seen as an effective way to increase psychosocial care access and use. The ADR team initiated the U-CARE software system adaptation to mobile devices because of this change in the research participants' technology environment.

The problems mentioned above relating to the U-CARE software system can also be taken as an instance of a class of problems (i.e., *coping with continuously changing design landscape*) faced while designing any eHealth research software in an academic research context with limited resources. Following the ADR method, Information Systems researchers wanted to consider design principles for the design process that would apply to a class of similar problems. Therefore, the case-specific research question was: *What principles should guide the design process to sustaining the usefulness of the eHealth research software in the continuously changing design landscape in the academic research context?* However, in this case, the design process involved multiple stakeholders, and resulted in changes in many parts of the U-CARE

software system. Hence, the problem class and research question are the same as in Chapter 6, but this case is richer as it considers multiple stakeholders and deals with the whole system. The next section presents the BIE cycles in this ADR case.

7.2 Building, Intervention and Evaluation Cycles

The U-CARE software system adaptation to mobile devices went through two BIE cycles (i.e., proof-of-concept and the U-CARE software system adaptation to the mobile devices).

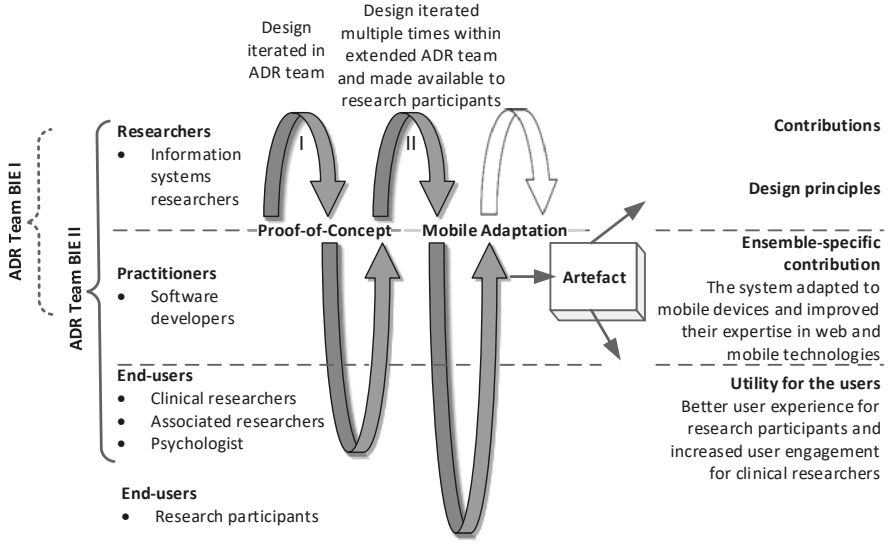


Figure 30. The BIE cycle during adaptation to mobile devices including contributions and stakeholders involved in the design.

Figure 30 shows the BIE cycles in which the U-CARE software system adaptation to mobile devices was put into the organisational situation, and formatively evaluated to meet the stakeholders' needs. Initially, in the first BIE cycle, the ADR team consisted of Information Systems researchers and Information Systems practitioners (i.e., software developers) in the U-CARE context. In the second BIE cycle, the ADR team was extended by including representatives from other stakeholders' groups, such as clinical researchers, associated researchers, psychologists, research assistants, et cetera.

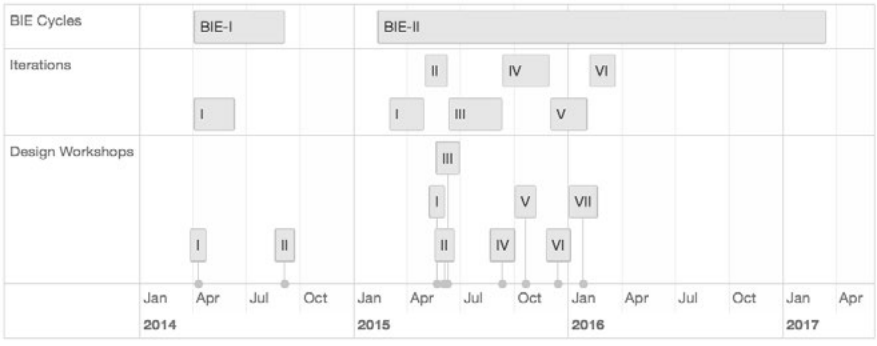


Figure 31. The timeline for the BIE cycles of adaptation to mobile devices.

Figure 31 gives an overview of development iterations and design workshops during the BIE cycles. Each BIE cycle is detailed in the following subsections, by discussing the build, intervention, and evaluation activities. At the end of each cycle description, an account of the lessons learned is provided. I built the *proof-of-concept* prototype during the first BIE cycles (Figure 15, pt. iii-e). The ADR team built the full adaptation of the U-CARE system in the second BIE cycle (Figure 15, pt. iii-h). The first three design workshops in the second BIE cycle were an evaluation of the existing system, and they were independent of build iteration sequences. The rest of the design workshops were linked to and based on the sequence of build iterations in the system adaptation.

BIE Cycle I

Build

The adaptation to mobile devices started as an explorative process with the purpose of better understanding the context of the design as regards (i) adaptation needs and the installed base, and (ii) the *technology landscape*, i.e., technologies and development strategies that might be relevant in the context.

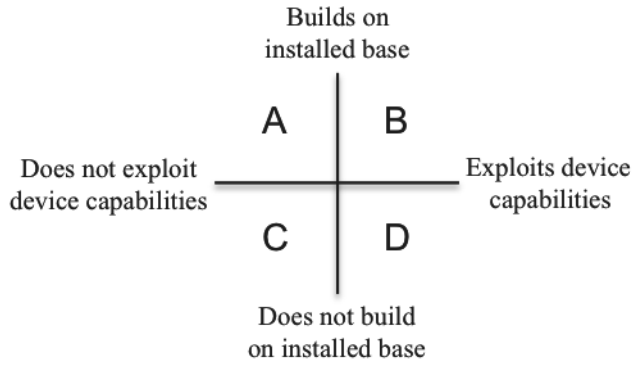


Figure 32. A rudimentary model for comparison of development approaches (Mustafa *et al.* 2014).

Literature and online resources (e.g., blogs and forums) were studied to learn more about mobile adaptation, including devices, operating systems, frameworks, standards, regulatory organisations, and developer resources. There are different types of applications when it comes to internet connectivity: some work offline, some need occasional internet connectivity, and some need the internet all the time. Appendix E.1 describes various approaches, which could be used to achieve mobile adaptation. Based on this, a rudimentary model (Figure 32) was created, to characterise various development approaches used in mobile adaptation. [A] refers to responsive design frameworks for making a website adapt to different screen sizes; [B] relates to various hybrid app solutions that mix web and native code using scripting frameworks, such as jQuery mobile, and thus enable websites to get the look and feel of the UIs of native mobile apps; [C] denotes dedicated mobile websites or separate mobile themes; and [D] represents completely native apps. Given the requirements R1–R6 (listed in Table 25), [A] was the ideal quadrant in the U-CARE setting for the following reasons: 1) The range of devices was increasing so fast that creating an app for each platform was not feasible for the U-CARE development team with its limited development resources; 2) Maximising the use of existing resources built on the installed base allowed re-use of infrastructure (e.g., privacy, security, et cetera); and 3) There would be only one system to design and maintain.

Following knowledge exploration, the ADR team presented the idea of adaptation to mobile devices as an innovation challenge during the U-CARE scientific advisory board (SAB) meeting in 2013, to engage with all stakeholders (Figure 15, pt. iii-c). Open feedback on the issue was gathered from the U-CARE stakeholders, and particularly the SAB members. There were two key results from the discussions: *technology could be used more effectively to promote behaviour change* and [this could] *enhance recruitment to intervention studies, for example, via the design of the [U-CARE software system]'s layout*. The SAB meeting resulted in an increased awareness of issues

related to mobile devices and triggered a further discussion on the adaptation to mobile devices within U-CARE.

The ADR team organised a second design workshop to discuss design choices regarding mobile adaptation (Figure 15, pt. iii-d). The workshop included a presentation of mobile adaptation choices and arguments for or against them, followed by a Q&A session and an open discussion. Here are a few highlights of the discussion:

Adaptation with minimal changes

> The review of the options we have is valuable […] I want to see if you run the (existing) system on mobile devices, how can we find the minimal situations to adapt the system for better user experience. I agree that it is elegant to adapt the system using Bootstrap [CSS framework] […] one way forward is experimental refactoring and modernising the development environment. (ISR-1, 2013, Design workshop)

Architectural challenges, limited resources and risks of change

> […] architectural issues prevent a mobile [native] app approach implementation […] if we redesign the entire software as a platform […] then we can have apps interact with the platform […] right now an app cannot be considered an option […] while solving one problem we may be creating more problems […] the change in the system is a risk too.
>
> […] we have to balance long-term goals and the vision for the platform with the immediate requirements […] there are issues related to resources, and our work is governed by the organisational context we are working in.
>
> […] this is most important at times when the system is running in a production environment and the studies (RCTs) are going on, with real research participants. Any change in views [UI] without thoroughly testing could result in discrepancies in the RCTs' research data, disruptions in information flow, and scrambled study material related to interventions. Ultimately, this could cause undesired (negative) effects on [research] participants' health. (ISR-1, 2013, Design workshop)

The discussion highlighted the characteristics of mobile adaptation as elegance from a user's perspective and robustness from a developer's perspective. The development team agreed to follow the *mobile first* philosophy and to implement responsive web design on an experimental basis using a CSS framework. At the time, the development team was working on the back-end refactoring of the existing software. The development team agreed that they should work on front-end refactoring as well, to implement the mobile adaptation needs. The development team assessed that the software required substantive refactoring on the front-end to be compatible with relevant third-party technologies for mobile adaptation. The details of this refactoring have already been discussed in Chapter 6.

Table 25. Mobile adaptation requirements

No	Requirement description	Stakeholders
R 1	The system should be accessible through the research participant's choice of devices	Clinical researchers
R 2	The system should provide security and protect the privacy of research participant	Clinical researchers
R 3	The system should require minimal development/maintenance effort	Development team
R 4	The system should use existing infrastructure in so far as possible	Development team
R 5	The system data collection should be consistent with the existing web-based system and should not corrupt the existing study data that have been collected over time	Clinical researchers
R 6	The system should remain in a stable state (we should be sure that it is working with 100% accuracy)	Clinical researchers

Basic requirements for the U-CARE software system adaptation to mobile devices were drawn up through a series of informal discussions with U-CARE stakeholders. Mobile adaptation had been recognised as necessary since the inception of U-CARE, but it had never been at the fore in the design process. However, early discussions during the design workshops for the technology adaptation process made it possible to identify a set of requirements for adaptation of the system for mobile use (see R1–R6 in Table 25). These requirements illustrate important conditions that governed the process of adapting the system for mobile devices.

After the technology upgrade (presented in Chapter 6), a proof-of-concept prototype was developed. The significant changes in the existing system were that a few JS and CSS libraries had to be removed, the code should be standardised using only one framework (i.e., Bootstrap), and the core UI should be changed for all users (i.e., the clinical researchers, research participants, psychologists, et cetera). Some architectural design decisions were made to accommodate the responsive design while utilising and maintaining the existing system architecture in so far as possible.

Intervention and evaluation

The proof-of-concept prototype was presented in a design workshop with only a few UIs converted to responsive design using the Bootstrap framework, such as home, log in and the developers' dashboard screen. The workshop participants (CR-3, CR-5, Dev-4, ISR-4, CR-2) were representatives of different stakeholder groups: a clinical researcher, a therapist, a developer, an Information Systems researcher and a person from the U-CARE management (i.e., product owner). The objective of the workshop was to show the stakeholders the responsive concept and how different UI screens responded when the screen size was changed. The workshop participants suggested three key recommendations as:

1. We should focus first on the research participant views.
2. We should get feedback on the prototype at bi-weekly sprint meetings.
3. We should get in touch with the research participants (i.e., end-users) for their feedback early on.

Once again, the development team was in the same situation as it was in during the technology adaptation process case (Figure 15, pt. ii-b). On one hand, there was a need for systematic testing and refactoring to achieve stability in the system; on the other hand, to continue the development of the prototype would have required extra effort in code synchronisation (due to parallel development in two teams). Therefore, prototype development was stopped until the development team had a stable system, and the management could reprioritise mobile adaptation.

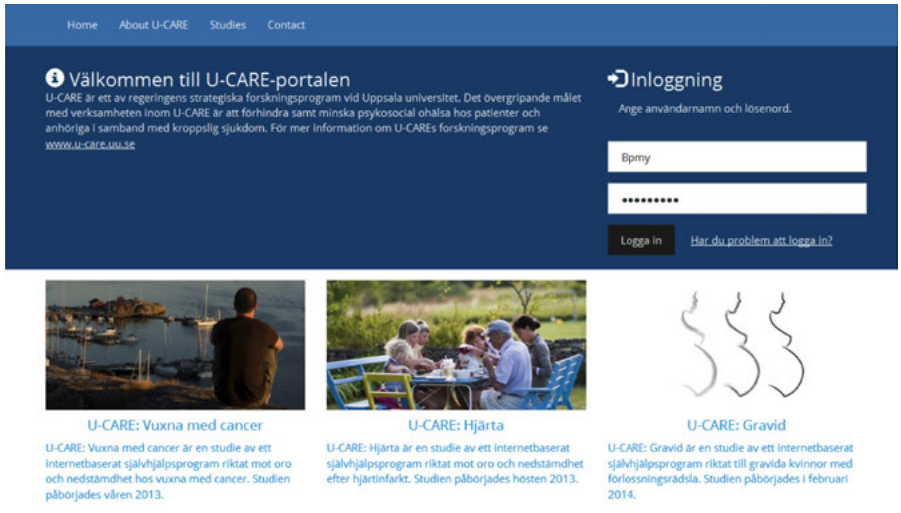


Figure 33. Proof-of-concept prototype – home page on desktop.

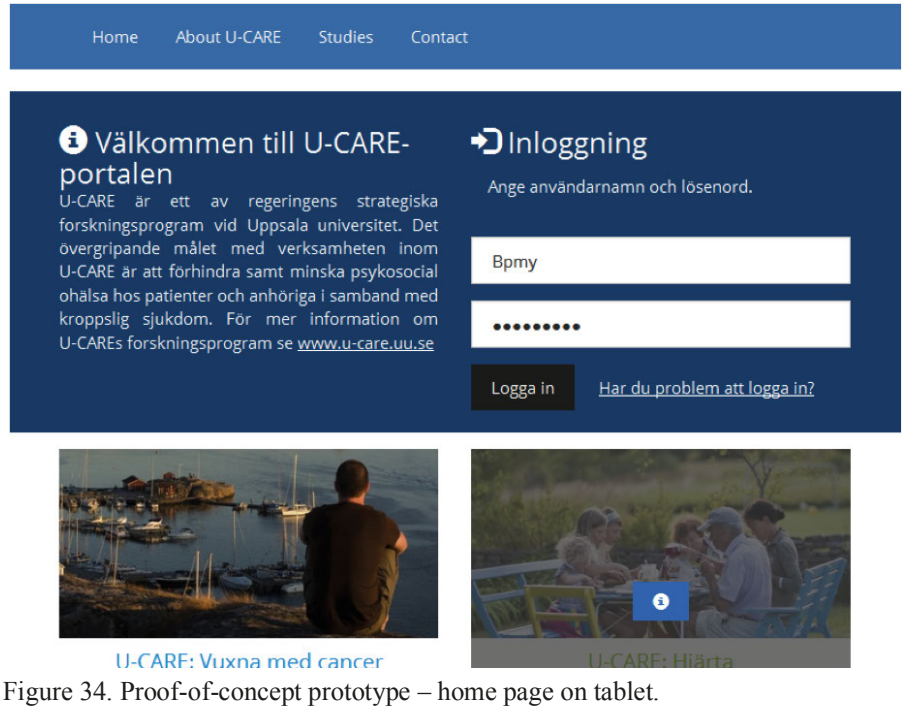


Figure 34. Proof-of-concept prototype – home page on tablet.

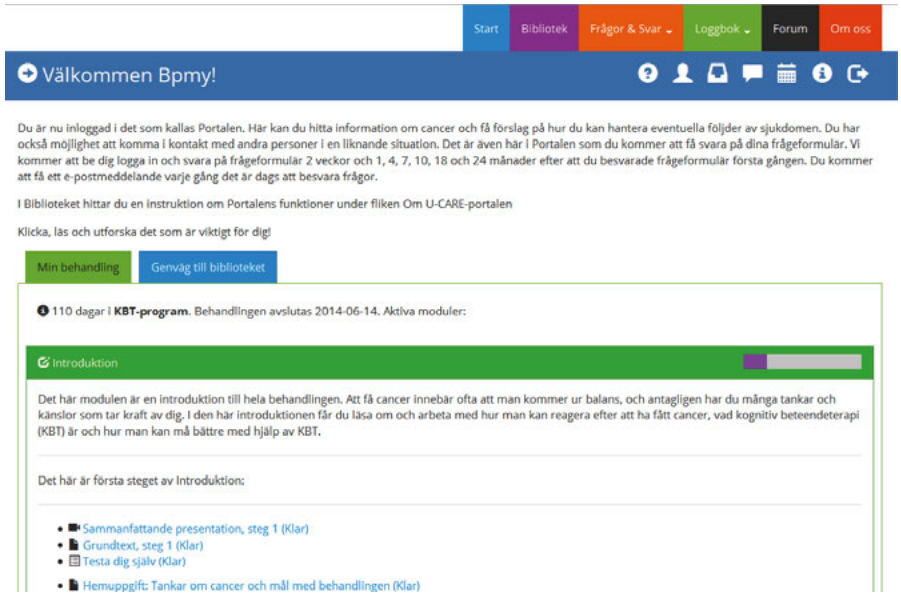


Figure 35. Proof-of-concept prototype – research participant dashboard on tablet.

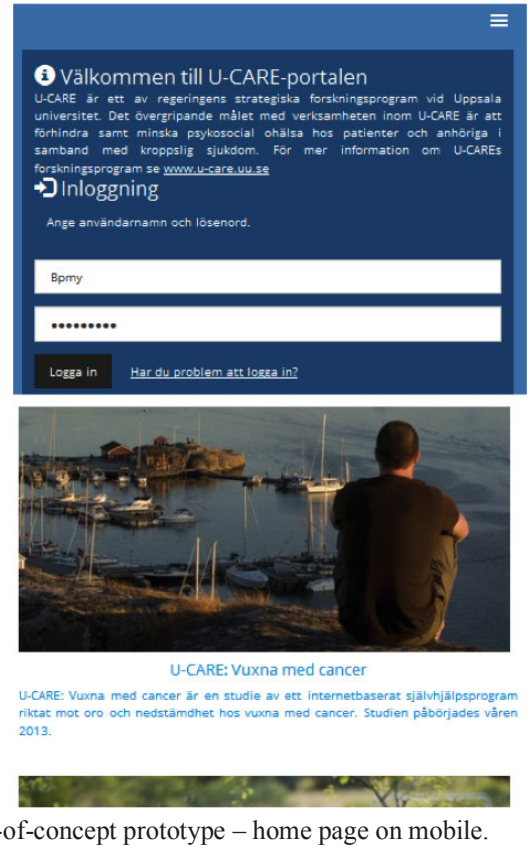


Figure 36. Proof-of-concept prototype – home page on mobile.

As mentioned above, at a previous SAB meeting (2013), a mobile adaptation idea had been presented. During the 2014 SAB meeting (Figure 15, pt. iii-f), Information Systems researchers described the importance and difficulties of designing flexible and interactive eHealth solutions. The proof-of-concept prototype was presented to demonstrate the interactive user experience on different mobile devices. Figures 33–36 are examples of the UI design of the prototype (more examples can be seen in Appendix E.2).

Open feedback on the matter was gathered from representatives of the various stakeholder groups, particularly the SAB members and associated clinical researchers. The demonstration led to an interesting discussion. Here are a few highlights:

Colour and readability

> How do you decide on colours? […] Well, it has to do with the readability. A very light green colour is not easy to read […] You might also have different profiles so that the user can choose. Either an old-fashioned black-and-white thing or something with colours or different colours […]. (Board member, 2014, SAB meeting)

Sustainability and adaptation to new releases of mobile platforms

> I was wondering about sustainability, if you program all of this and Apple decide [to release] iOS 10 or something, do you have to re-program everything, or it is simple to achieve it? Do you have to change all of it? (Board member, 2014, SAB meeting)

> It is platform-independent as well as hardware-independent. That is why we are using responsive design […]. (ISR-4, 2014, SAB meeting)

UI, user experience, attractiveness and appealing design

> Just a quick comment on the presentation here is that if the intervention is accessible from a broader range of devices such as mobile phones, we find it likely that more people [will] go there and use it. And if it is not accessible, we find it likely that they [will] consider this software to be old-fashioned and not appealing and that […] might scare people away basically. If the technology is old and non-accessible, so that is my interpretation of [what ISR-4 is saying] the sense that if it feels old and not as good as the other software, they are using […]. (ISR-1, 2014, SAB meeting)

> But this is also a complicated aspect. We have been trying to, we talk a lot about the design, the layout, or the [U-CARE software system] which we use at the moment, and we always end up with different opinions about what is nice and what is not nice and how should be presented […]. (CR-1, 2014, SAB meeting)

> For most patients, I think the simple design with basic texts is enough for them to get started with the treatment and the behavioural changes what we think

> are good for them. And then I think there is a group that may be thinks *What? I can't log in with my mobile phone! Then I will not do it*. And maybe they are not text people, so they need pictures and multimedia […]. (SAB participant, 2014, SAB meeting)

User engagement and usability

> I think it is hard to [imagine] that you will be able to redesign the whole [system] and see as a global increase in treatment response. I think you have to be more like guided by usability design and what is good practice, rather than test everything and randomise control trials. (Board member, 2014, SAB meeting)

There were two key results from the discussions: *it is necessary to remove technical obstacles to make the [U-CARE software system adapt for] all devices* and to *keep the design simple but attractive*. The workshop resulted in an increased awareness of design issues regarding the U-CARE software system adaptation to mobile devices and triggered a further discussion on the UI design within U-CARE.

During the subsequent IT meeting, it was suggested to make a new, more modern theme for the U-CARE software system; in other words, to adapt the UI for mobile devices. Furthermore, it was also suggested to outsource this task as a Master's thesis project due to a limit of resources, skill sets and knowledge required for such adaptation within the development team. This resulted in publishing an advertisement for a degree project for a Master's student to develop a responsive web application for smartphones and tablets (see Appendix E.3). It is important to note that during this period, the existing development resources were prioritised for stabilising the system, the quality of the system, and the design process improvement. One such example is the developers' three-day workshop (Figure 15, pt. ii-i). Core processes for research participants on the U-CARE software system were documented. The objective of this documentation was to enable for itemisation of requirements for developing the mobile web application by the Master's thesis student.

Reflection and learning

Due to the complexity of the existing system, the comprehensive redesign of the system for mobile adaptation was resource-heavy and required the investment of a significant amount of time and engagement from the stakeholders. The prototype design was not the final design solution, but it enabled the ADR team to instil design curiosity among the stakeholders and enabled the stakeholders to explain what they wanted. The interactive prototype showed the idea of mobile adaptation and what the development team was trying to accomplish with their concept. The prototype generated actionable feedback from stakeholders. The first draft of design principles was phrased based on learnings during this cycle (see Table 26).

Table 26. Design principles for sustaining the usefulness of eHealth research software (version 1)

Design principle	Specification
The principle of engagement with stakeholders	The software developers of eHealth research software should continuously engage with stakeholders, in order to adapt the software in a direction which will satisfy stakeholders, given that the stakeholders are willing and committed to such engagement in the long term.

The next cycle elaborates on the design activities that enabled the U-CARE software system adaptation and its actual use by the research participants.

BIE Cycle II

Build

The proof-of-concept prototype demonstration and discussion around the U-CARE software system's UI resulted in the system's adaptation to mobile devices. The feature request was added to the product backlog and given high priority. The effort to find a Master's thesis student to adapt the U-CARE software system for mobile devices failed, due to a lack of applicants with the right competence. Ultimately, the task was assigned to the development team. Previously, the development team's main focus had been on system quality, developing the test framework, and design process improvements. Once the development team was back on track after the bumpy road of the technology upgrade, the U-CARE management decided to give top priority to the mobile adaptation.

At this stage, the Information Systems researcher who was team leader and technical lead left the development team. The product owner took over the role as team leader, and one of the developers was given the technical lead role. This change affected the design process, which is discussed in the intervention and evaluation section below. In this BIE cycle, the ADR team was reorganised and extended by including representatives from various stakeholder groups. Meanwhile, the development team became smaller in size due to two full-time development team members leaving, and also some of the Information Systems researchers, who contributed 10% in the development, becoming less involved, as they were busy with their research work (Figure 15, pt. iii-g).

The product owner and team leader took the leading role in the ADR team. The team leader also held many design workshops. The technical lead was responsible for ongoing system maintenance (bug fixes) and support-related issues. The rest of the development team, consisting of two full-time developers, had mobile adaptation responsibilities. The development team investigated the adaptation of the U-CARE system to mobile devices with minimal

development efforts. The technical lead prepared an initial plan on how to proceed in this regard, while the ADR team determined the goals of the mobile adaptation. They were as follows:

1. Adapt the research participant views (UI) for mobile devices.
2. Changes should not compromise research (i.e., research participants should have the same functionality and content on mobile and desktop devices).
3. Ensure better user experience on mobile devices.
    a. Resolve mobile-specific issues (e.g., PDF, video, et cetera).
    b. Adapt in accordance with user expectations based on experiences from using other mobile apps, for example as regards look and feel.

The project went through multiple iterations to achieve the aforementioned goals. The length of the sprints was different in each iteration, while maintenance of the existing system remained based on two-week sprints. The development team started from scratch and learned the Bootstrap framework by the trial and error method. Based on their experience and knowledge, they considered different design decisions and design practices to achieve the goals. One such decision was to keep the existing jQuery UI framework and jQuery plugins. Due to this decision, the existing proof-of-concept prototype could not be used as-is. Instead, the development team used bits and pieces from the prototype to implement the Bootstrap framework. Later in the adaptation process, the design decisions were revised. For example, the development team started to test the system using real mobile devices and tablets instead of emulators, for a more accurate and actual user experience. The following section describes in detail how the ADR team went through several iterations and adapted the system based on the feedback received during various design workshops. During design workshops and between iterations, exploratory tests were performed by focus groups, the development team, and a few selected individuals. Notes were taken and used for next iteration of testing.

Iteration I

The development team investigated different technical solutions to make the system usable on mobile devices. Microsoft Word documents were used to communicate the tasks, schedule, feedback, design decisions, and problems and their resolutions, instead of using the existing product backlog and feedback feature. The documents were stored centrally and shared among all stakeholders to make the process transparent. All stakeholders, particularly the clinical researchers, were heavily involved in this effort and provided lots of feedback about the design. During this iteration, a learning session was organised with an expert on usability, to understand the challenges of mobile application development. It focused on usability in UI design.

Design decisions and implications

The development team created a separate branch of the code for this project (i.e., mobile-legacy). They decided to design a new theme/template for the system, which could be adapted to different mobile device screens. The decision was made that a strong base of UI should be created during the initial adaptation, which would be used to convert the whole existing U-CARE system to a more modern, responsive design. In other words, the effort was focused on learning the technologies and identifying the challenges. The decision was made to keep the existing functionality in so far as possible, to rely solely on CSS, which should be purely cosmetic, and to avoid changes to the HTML at this point. The branch name mobile-legacy indicated the intention that the goal was to keep as much of the legacy code as possible, until a complete redesign of the entire visual layer could be carried out. The decision was made to perform the beta testing on all supported [real] mobile devices. The development team also decided to implement the responsive design using Bootstrap framework, but they kept the existing library jQuery UI framework. Initially, they decided to only design a theme, but over time they learned that they had to change the HTML in the views as well. Then they decided that they would only change views that were related to research participants' actions/activities and keep the overall existing architecture unchanged. Later, this resulted in challenges.

Design actions undertaken

The developers trained themselves in HTML, CSS, and JS, skills needed for responsive design development. They assessed the viability of the refactoring with fewer developers, who were ready to learn but had limited experience with the latest technologies. They prepared a UI policy for the process. Each iteration was deployed on the test server for alpha testing. In the beginning, the mobile adaptation was tested using mobile device emulators. In some development cycles, the responsive design was implemented. The developers built a theme specifically for mobile devices, which loaded if a user accessed the system on a mobile device and a clinical researcher had enabled the mobile theme for the particular study. In the theme logic, the Bootstrap library was overridden by the jQuery UI to keep the existing CSS classes working like they had before. The developers created theme-specific CSS to deal with CSS class conflicts, consisting of additional CSS classes to implement the responsive design.

Iteration II

The development team demonstrated their new mobile adaptation prototype to the U-CARE stakeholders. The most notable visual changes at this stage were that the existing left icon menu had been removed and added as a small

icon menu at the bottom of the screen. Based on the feedback from stakeholders, the development team improved the menu design, and started working on the second iteration.

Design decisions and implications

The decision was made to invite a clinical researcher to act as a test user (a surrogate for research participants) and go through the list of views in the Google Chrome web browser, which needed to be configured to emulate different mobile screen sizes. The suggestion was also made to invite all U-CARE researchers[73] (i.e., IT meetings participants) by email whenever testing of the mobile adaptation prototype was needed. Their task would be to provide feedback on the visual and functional aspects of the system. Examination was done of the technical limitations of mobile technologies, to uncover potential *show-stoppers* early on. The clinical researchers were informed that there was still a risk that adding new functionality for mobile devices might not be feasible within the short timeframe given (four months). The decision was made to go through three major parts of the U-CARE software system and get feedback from stakeholders to identify any problems and then fix the problems in the next iteration. The development team organised three workshops to get all stakeholders on board (see below, Design workshops I–III).

The development team decided to keep the documentation of any improvements or changes to the code of the existing system for use later in merging both branches. The decision was made not to duplicate existing UIs for mobile adaptation. Later, the development team was required to add tasks to the product backlog and to keep the task status updated. The research participants' views were divided between two software developers, who worked on them separately. The decision was also made to continue the exploration and training on CSS and JS during the next iteration (which was going to deal with strategies for managing multi-media audio/video content and PDF files on mobile devices). The Klonk pain survey[74] was excluded from the process, as it did not seem likely to work on mobile devices [being an external service, the development team had no control over its design]. The library and the CBT section of the system were also excluded.

Design actions undertaken

In the first iteration, the development team' main focus had been on the UI design and configuration of the JS and CSS assets. In the second iteration, the team mostly focused on adapting different features of the U-CARE software system to responsive design, such as IM, chat, forum, questionnaires, home-

[73] All researchers who are working in the U-CARE context, including clinical researchers, associated researchers, health economists and Information Systems researchers.
[74] http://drawsurvey.com/ [accessed: October 9, 2014, an external questionnaire service].

work, and user navigation. They also adjusted the font size of the text depending to the device screen size. The business logic regarding the handling of actions and loading assets was refactored to accommodate the responsive design implementation. Three design workshops were organised.

Design workshop I

This workshop mainly focused on the features of navigation, forum, chat, IM, and 'ask an expert.' The workshop participants (CR-2, CR-3, CR-8, ISR-4, Dev-4, Dev-5, Dev-6, RA-1, and RA-2) were given a list of tasks to perform before the workshop, using test user accounts on the alpha version of the U-CARE software system. It was deployed on the test server for this purpose with a test database. The workshop participants reflected on their first impressions of the system and how the navigation worked. They also described and provided feedback on the problems they encountered while using the navigation, forum, chat, IM, and 'ask an expert,' for instance regarding 'navigating on the homepage,' it was stated that the "top buttons are too small" (see Appendix E.4, for additional details on the task list and feedback given).

Design workshop II

At this workshop, the mobile UI was discussed in regards to the questionnaires and homework. The clinical researchers investigated the existing studies to see what could be improved to enable use of the existing material for future studies on mobile devices. The workshop participants (CR-2, CR-3, CR-4, CR-8, ISR-4, Dev-4, Dev-5, Dev-6, RA-1, and RA-2) were asked to log in to the system and to try to answer questionnaires using test user accounts. After filling out the questionnaires, they received a mock intervention. At this stage, the PDFs in the intervention were not working, so the workshop participants were asked to try to answer the homework only. They described their experiences of filling out the questionnaires and homework. They also provided feedback on the problems they had encountered in the process, for instance regarding 'filling out questionnaires,' it was stated that "table labels are too wide" (see Appendix E.5, for additional details on the task list and feedback given).

Design workshop III

At the third design workshop, the library and PDF files were discussed. Before the workshop, the invited workshop participants (CR-2, CR-3, CR-4, ISR-4, Dev-4, Dev-5, RA-1, and RA-2) logged in using test accounts and navigated the library section. They were asked to explore the library slide menu and provide suggestions on optimal design, as there were already several menus on the page. At the workshop, the workshop participants (CR-2, CR-3, Dev-4, Dev-5, RA-1, and RA-2) provided feedback on the problems they had encountered while using the library and footer menu, for instance regarding 'try to navigate in the library slide menu,' it was suggested to "replace the current

library slider with a slider plugin that works well on mobile" (see Appendix E.6, for additional details on the task list and feedback given).

The design workshops (I–III) not only engaged the stakeholders, but also involved them as co-designers in the mobile adaptation design process. This led to initiation of a design activity to redesign and adapt the CBT content in the U-CARE software system.

Iteration III

During this iteration, the main focus was to improve the UI for different components of the system based on the feedback received in three design workshops during Iteration II. The clinical researchers also considered the CBT part of the system, which had been left out in previous design workshops. How different media formats (PDF, video, and audio) were rendered on different mobile devices with different operating systems and browsers was also investigated in this iteration.

Design decisions and implications

The main decision in this iteration was to change the HTML of views, as well as to implement responsive behaviour using the Bootstrap framework. It was suggested to use text instead of PDF format. The existing video files were in the Flash video (Flash Live Video – FLV) format. This video format did not work on Apple devices. The decision was made to convert all video files to a supported format. There was no decision made regarding the image files (e.g., resizing and rendering different resolutions on different devices). It was suggested to the clinical researchers to change the CBT, homework tasks, and questionnaires to support mobile devices. Audio files were already in MP3 audio format, which was compatible with mobile devices. The decision was made to keep them as-is without any changes.

Design actions undertaken

The homepage was made responsive, and the pictures from the original homepage were hidden. Only the login feature was displayed at the top. Forum, chat, and IM were adapted to mobile devices. The development team integrated notifications in the footer based on the activities in the chat and IM. Menu options were reorganised and moved. Various issues were fixed regarding pop-up screens and alignment on the chat page. This made it more obvious when a user was visible to others in the chat feature. The library section was adapted using wiki-style[75] for titles and subtitles, and aligned images based on the size of device browsing the library pages. The forum and questionnaire

[75] In the U-CARE context, wiki-style refers to the Wikipedia heading style seen when the wikipedia.org website is accessed on mobile devices.

layouts were improved; for example, some questions were changed from columns to rows to look better on mobile devices.

In the first stage, the development team performed a preliminary test of CBT modules. They activated modules, viewed items, submitted homework tasks, and read feedback. Based on this, they suggested to the clinical researchers to avoid having more than four tabs in any CBT homework/questionnaire. They also provided a list of homework that needed to be rewritten by the psychologists to fit on mobile devices in accordance with the suggestions discussed in the second design workshop. In the second stage, the development team created test user accounts for the clinical researchers to go through the CBT. As a result, they also received additional feedback from the clinical researchers. The development team converted all FLV video format files to MP4 format. They also replaced the old JS video player plugin with an HTML5 video element.

Design workshop IV

This workshop was organised to test all components of the system (i.e., navigation, communication, questionnaire, homework, library, multimedia content, and CBT module). The workshop participants (CR-2, CR-3, CR-4, ISR-4, Dev-4, Dev-5, Dev-7, RA-1, and RA-2) were given a list of tasks to complete and to provide feedback on during the workshop. The workshop participants provided feedback on the problems they encountered while using various components of the U-CARE software system (see Appendix E.7, for additional details on the task list and feedback given).

Iteration IV

In this iteration, the development team went over the feedback from workshop IV and implemented the requested changes.

Design decisions and implications

The decision was made to test the functionality of the portal thoroughly, after changes were made for mobile adaptation and before going live. The reason was that the previous iterations had been more focused on if the UI of the U-CARE software system worked well on mobile devices. The development team also decided to implement a new version of responsive EQ5D. The clinical researchers had created a new study with mobile-adapted questionnaires[76]. They also created a mobile-adapted CBT and homework tasks for this new study. The new study was tested on the alpha server.

[76] Research on how to adapt the standard questionnaire for mobile devices is still in its infancy. There were no guidelines for making a standard questionnaire adapted to the mobile device while minimising the effect that may have on questionnaire responses due to different characteristics of mobile devices compared with PCs (e.g., smaller screen sizes).

Design actions undertaken

The library wiki-style headers (not just the text) were made clickable. They integrated Glyphicons (font format icons). The carousel slide menu was improved, and it was no longer greyed out when not selected. Instead, a highlighted box appeared. Also, a swipe functionality was added to the carousel menu. The images in the slide menu were aligned based on the size of the device that was browsing the library pages. General issues with clicking on carousel items, viewing the first library section, loading times, as well as paging that was not working, were all fixed. The FAQ, forum, IM, and navigation sections were improved. The new EQ5D link that supported mobile devices was incorporated.

During the mobile adaptation, the HTML markup of the UI was also changed. The questionnaire now worked correctly with the newly created mobile theme. However, to verify that the UI was still functional with the old desktop themes (backward compatibility), the questionnaires were tested using the old desktop theme, but with the changed code in the views. The testing was performed using one study and running a few questionnaires as a sample to identify possible bugs or visual inconsistencies.

Design workshop V

This workshop focused mainly on planning the testing. During the mobile adaptation, there were many changes in the system views (i.e., UI). During this workshop, the development team planned to test the system using the existing themes and rendering functionality. The main objective was to test backward compatibility, to make sure the mobile adaptation did not affect the desktop version. The system had to work in the desktop environment as well. During the workshop, the decision was made to use the existing recorded test using Selenium. These was also a discussion regarding use of an automatic tool like Eyes.Selenium[77] for visual testing. This tool would add automated visual validation for Selenium tests to verify that the UI appeared correctly across all devices and browsers. Another in-house developed tool was also discussed which would use the perceptual image difference to test two parallel versions during the technology upgrade.

Iteration V

In this iteration, the development team implemented the test plan prepared in workshop V by increasing the test coverage of recorded tests, executing tests, and correcting the code to pass failed tests.

[77] https://www.nuget.org/packages/Eyes.Selenium and https://applitools.com/ [accessed: October 14, 2015].

Design decisions and implications

The development team decided to proceed with recorded tests for the time being and to consider integrating automated visual testing tools into the testing framework at a later stage. As the recorded tests did not cover all parts of the system, the decision was made that the development team should work on test coverage first and then execute the test on both branches of the code (the old version and the mobile adaptation). It was also decided to verify any information stored in the database and that was a consequence of user activity. It should be identical when running either branch of the code, which would validate the data storage accuracy.

Design actions undertaken

The development team ran tests on the old code branch. They also ran the tests on the new mobile adaptation branch and on the merged branch. The team documented all errors during the revisions (see Appendix E.8) and corrected the bugs. This was done as part of the alpha testing. After this, a beta version was released on the test server. Once again, the design workshops were held for beta testing (see below for details).

Design workshop VI

This workshop was organised for beta testing. Instructions were sent to the workshop participants (CR-2, CR-3, CR-4, CR-5, CR-7, ISR-4, Dev-4, Dev-5, Dev-7, RA-1, and RA-2) with a list of tasks to be completed, so they could provide feedback during the workshop. The workshop participants provided feedback on any problems they had encountered while using various components of the U-CARE software system (see Appendix E.9, for additional details on the task list, scenarios and feedback given).

Design workshop VII

This workshop was held to perform a second round of beta testing after addressing most of the feedback from the previous workshop. These very intense, continuous, and interactive feedback sessions were organised with a limited number of participants (ISR-4, Dev-4, Dev-5, RA-1, and RA-2). Testing steps from previous sections were already matured and most of the problems were identified, so a limited number of workshop participants went through all the steps and spent more time on evaluation. Small numbers allowed them to perform evaluation continuously for an extended period and in a more interactive and shorter feedback cycle with the development team. The workshop participants once again provided feedback about the U-CARE software system (see Appendix E.10, for additional details on the task list and feedback given). There was also feedback which was not directly related to the mobile adaptation, but which was valuable all the same. The additional feedback resulted in improvements to the existing system at a later stage.

Iteration VI

This was the final iteration. The code was published in a production environment and tested again before the clinical researchers could enable access through mobile devices. Here are some screenshots of the UI design that was published in the production environment at the end of this iteration (see Figures 37–41).

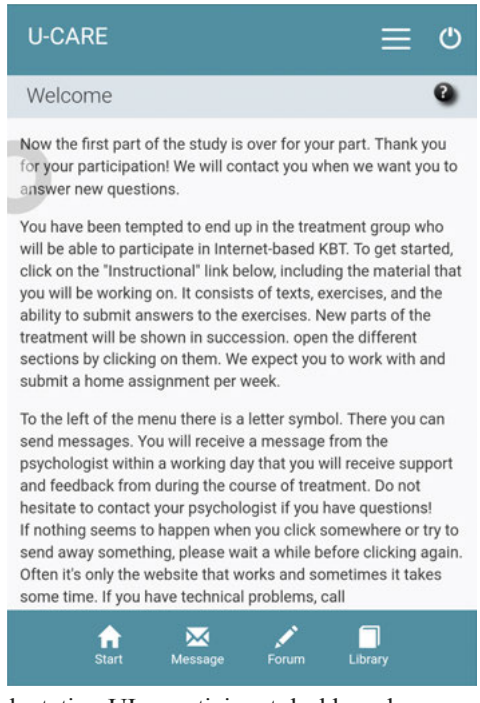


Figure 37. Mobile adaptation UI – participant dashboard.

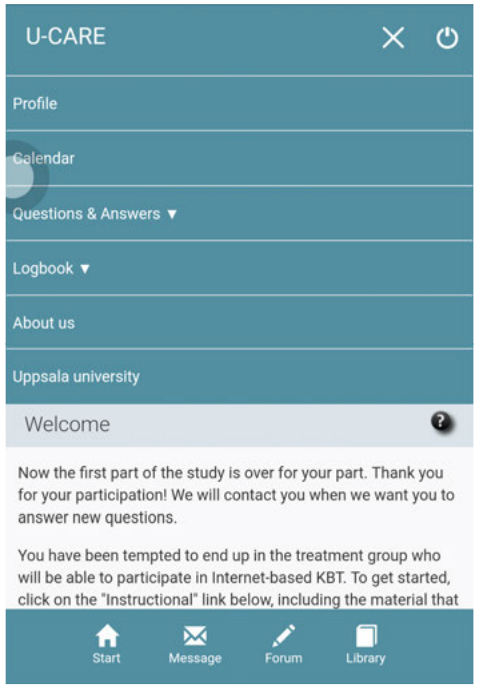


Figure 38. Mobile adaptation UI – menu options.

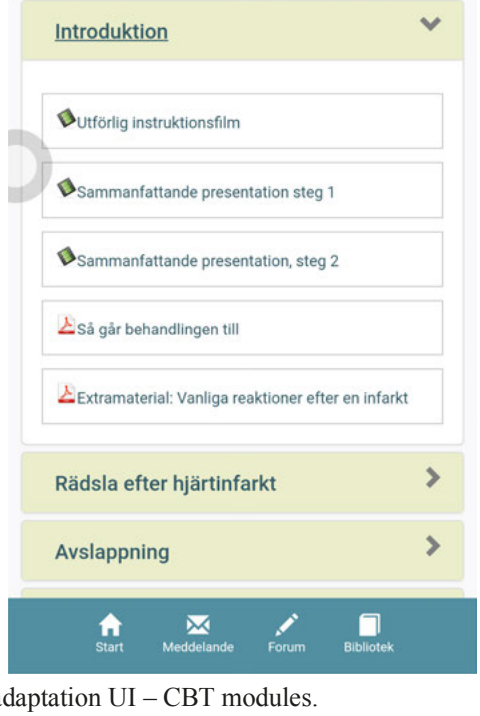


Figure 39. Mobile adaptation UI – CBT modules.

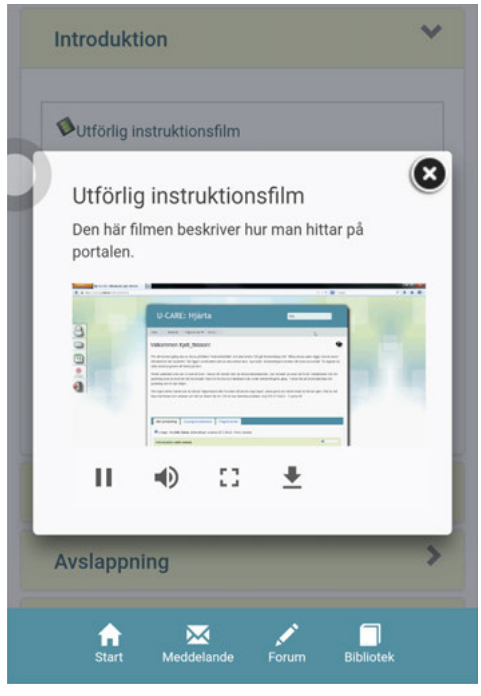


Figure 40. Mobile adaptation UI – video player.

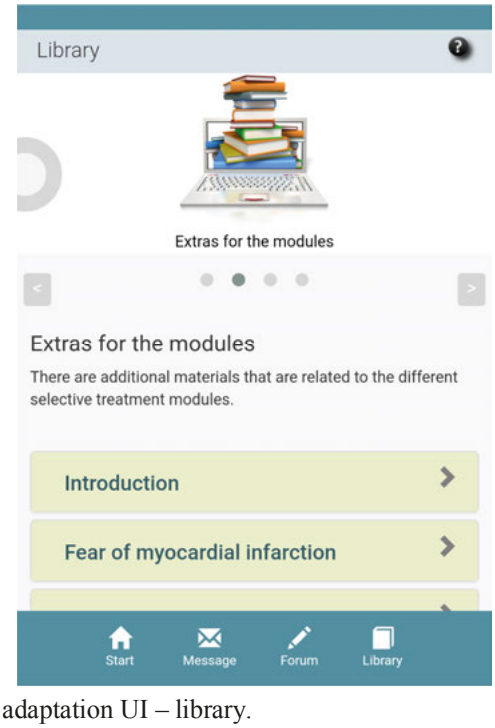


Figure 41. Mobile adaptation UI – library.

Design decisions and implications

The development team decided to release the new version of the U-CARE system once the test results were satisfactory and feedback was positive. The development team communicated to all stakeholders that the system was ready for use on mobile devices. They also communicated that although the system supported several types of devices and operating systems, it did not support all types, as some features required the latest versions of the operating systems and browsers. The development team also recommended tablet mobile devices, as some mobile phone screens were too small. Use of such devices would make the user experience sub-optimal, especially if research participants accessed the intervention or answered questionnaires. The development team enabled system access through mobile devices one study at a time. The possibility to configure research studies enabled switching to the mobile-adapted UI through *theme* settings, which allowed some studies to enable the mobile-adapted UI, while others could keep the desktop-adapted UI.

Design actions undertaken

The development team kept track of the research participants' devices and only allowed access to the mobile-adapted study and content if the device was supported. If a research participant had a device that was not supported by the system, the system informed the user with a message and recommended use of a computer.

Intervention and evaluation

In early March 2016, the latest version of the system was deployed in production after mobile adaptation (Figure 15, pt. iii-i). U-CARE Heart was the first study which allowed their research participants to use mobile devices. This was a smooth transition with next to no errors or system failures.

The success of the U-CARE software system adaptation to mobile devices was evident in the usage statistics, based on an analysis of environment logs performed after one year of system use (see Table 27). In fact, over a quarter (25%) of the research participants began to use the adapted system once it was deployed. Interestingly (as elaborated on below), in the U-CARE Pregnant study, many research participants were accessing the system through mobile devices before the adaptation, and after mobile adaptation, this number of research participants increased even further. The analysis of environment logs also revealed other relevant findings, for example in relation to the operating systems of mobile devices (only those known and detected). In the case of the Pregnant study's research participants during 2013–2015, 69.5% used iOS and 30.5% Android. However, describing and presenting the detailed analysis falls outside the scope of this dissertation.

Table 27. Research participants' mobile devices usage statistics

No	Duration	Entries	All	Heart*	AdultCan*	Pregnant
1	2013-06-28 to 2014-03-15	1,802	4.1%	5.4%	5.0%	
2	2014-03-15 to 2015-03-15	7,379	16.7%	8.9%	6.8%	29.6%
3	2015-03-15 to 2016-03-15	8,244	14.3%	8.0%	12.0%	41.2%
4*	2016-03-15 to 2017-03-15	4,165	24.1%	23.6%	24.4%	56.7%

* Only the indicated research studies enabled the mobile-adapted theme and only during 2016–2017. In other cases, research participants had accessed to the U-CARE software system using the previous theme, which was not adapted to mobile devices. Data are based on analysis of the research participants' environment logs from 2013–2017.

The success was also evident in the product backlog and feedback log, where there were no system failures or major bugs reported regarding the mobile adaptation. The support issues log also did not show any mobile adaptation-related issues reported by the research participants. The results indicated that the clinical researchers were satisfied with both the content and the system adaptation for mobile devices, and that the system was stable over the year.

The U-CARE software system was adapted to mobile devices based on the assumption that this would lead to increased research participants' engagement (or simply increased activity and use of the system), inclusion, and retention. While we could see that access to the system through mobile devices had increased through the environment log, comparing research participants' activity on mobile vs. desktop would require further investigation. The clinical researchers had not only planned such analysis, they also had ethical approval for it. Such investigations are to be conducted once the intervention studies of the clinical researchers are concluded. It is beyond the scope of this dissertation to go into the details of the evaluation of engagement and effectiveness.

Reflection and learning

The research participants' age and tech-savviness are significant factors to consider when adapting the eHealth interventions for mobile devices. The research study Pregnant did not require a lot of support or reveal many technical issues, possibly because the age of the research participants was lower than that of the research participants in the research studies U-CARE Heart and U-CARE AdultCan. The analysis of the environment log also revealed that a large percentage of the research participants in the Pregnant study accessed the U-CARE software system through mobile devices even before the system and the CBT contents officially supported the mobile devices. In January 2016, when the U-CARE software system was adapted to mobile devices, the Pregnant study was nearing its end. The clinical researchers also realised that older research participants would not be able to interact with the contents on mobile phones (particularly with a small screen), as contents were not tailored

for such an audience. Therefore, older research participants were recommended to use mobile devices with large screens, such as tablets or desktop computers.

During the second BIE cycles, the ADR team instantiated design principles, that were formulated during the technology adaptation process case (Figure 15, pt. ii-l), into the U-CARE software system and the design process. The design process was supported by a) adequate test coverage, b) frequent and automatic running of tests (daily builds), c) continuous and iterative development, frequent design workshops with detailed feedback, d) tools like TeamCity, Selenium IDE, and multiple real mobile devices for proper testing, and e) a development team with continuous upskilling regarding the tools and technologies for mobile development (e.g., HTML5, CSS3, Bootstrap, jQuery, FFmpeg for audio/video format conversion, et cetera). The improved design process enabled the development team to adapt the U-CARE software system in accordance with the changes in the technological landscape, as well as to fulfil the requirements of stakeholders. This meant that the development team followed *the principle of technological-ecological adaptation*. As a result, the design principles guided the design process in sustaining the usefulness of the U-CARE software system.

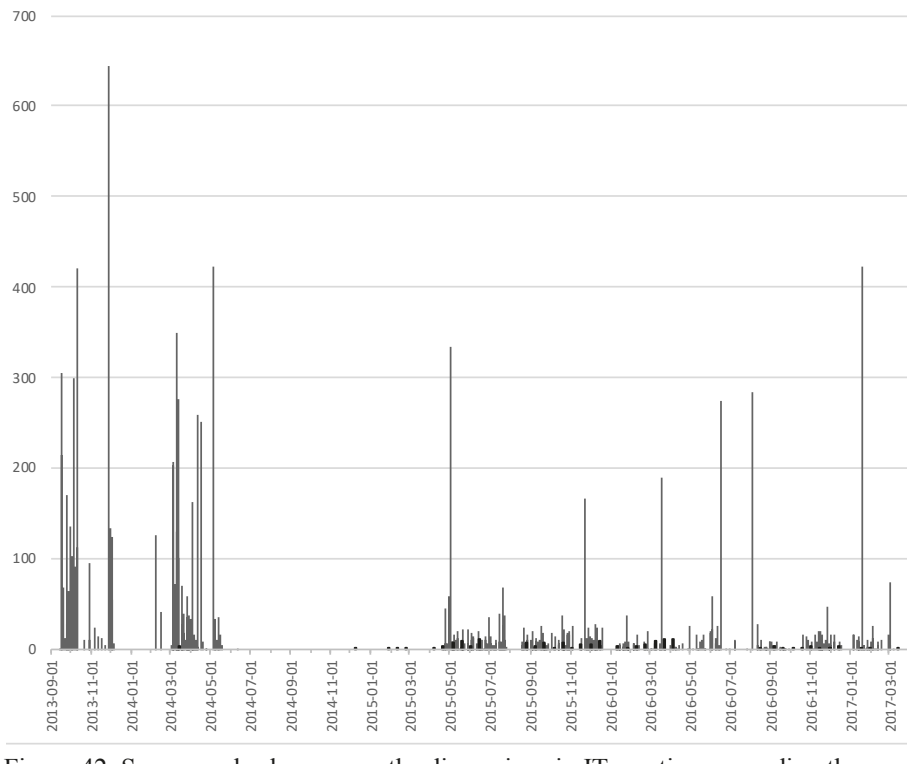


Figure 42. Source code changes vs. the discussions in IT meetings regarding the mobile adaptation.

Retrospective analysis of the design process, using CoDisclose, revealed that the mobile adaptation, like the data export feature, remained in active development for an extended period. Figure 42 shows the changes in the U-CARE software system source code (light grey bar), based on data from September 2013 to March 2017. It is important to note that there were many source code changes during the technology upgrade within a short period (September 2013 to December 2013). Similarly, many source code changes were made during the mobile prototype development in the first BIE cycle, which took place during a short period (March 2014 to April 2014). However, the source code changes during the second BIE cycle were fewer and took place over a long period (May 2015 to March 2017). The small changes during the long period indicate that the development team was embracing the proactive quality assurance practices in the design process. Figure 42 also represents instances of IT meetings at which the stakeholders' discussions concerned the mobile adaptation (black bars). It is important to note that the instances of stakeholders' discussions are almost invisible in Figure 42, because most of the process was not discussed at the IT meetings, but rather at specialised and documented design meetings (see, e.g., Appendices E.4 to E.10).

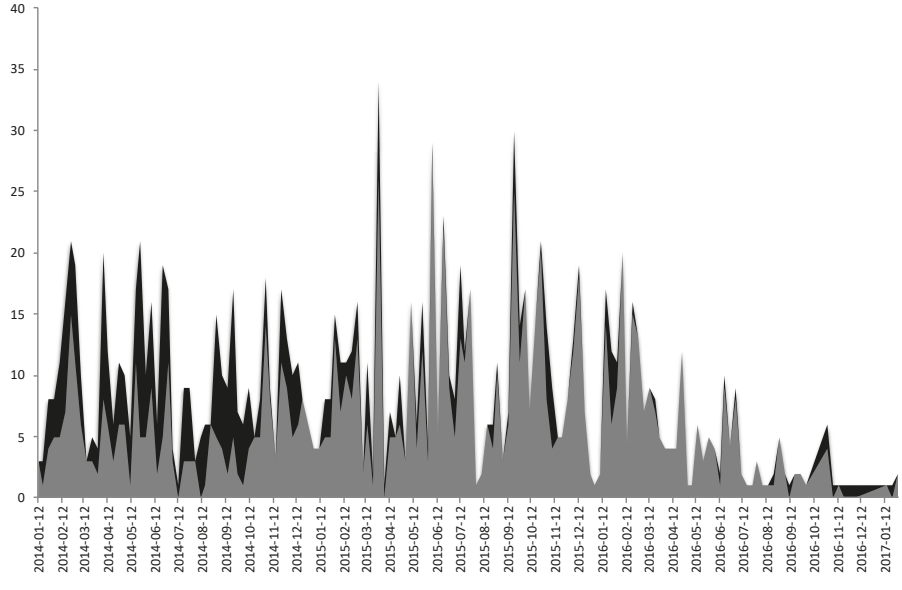


Figure 43. Technical debt during mobile adaptation.

An analysis of technical debt revealed that it had decreased over time. Figure 43 shows the number of bugs to fix (top area, in black) and the number of non-bugs to fix (bottom area, in grey). The figure includes only high priority issues that were reported in the product backlog. The diagram illustrates that the technical debt decreased over time as more features were developed, while

bug-fixing decreased. Also, this shows that the development team was successful over time in sustaining system functionality, while minimising bugs and errors. Following quality assurance, the development team gave a continuous high priority to testing activities. The testing safety net enabled the development team to adapt the system to changing requirements. The U-CARE management, especially the product owner, allocated the resources for refactoring, documentation, writing tests, and design workshops. The development team used a *learning by doing* approach, which also required time being dedicated to gaining skills regarding various technologies. Thus, the development team was given extra time for learning in this case. This resulted in the U-CARE system increasing in testability and extensibility. In other words, the development team observed *the principle of embracing proactive practices*.

During BIE cycle II, the development team involved all stakeholders early in the mobile adaptation, observing following *the principle engagement with stakeholders*. The stakeholders also showed their willingness and commitment to engage in mobile adaptation. They were engaged throughout the second BIE cycle, which can be seen in the participation in design workshops and their detailed feedback (see, e.g., Appendices E.4 to E.10). The development team not only engaged with stakeholders, but also made the mobile adaptation in a co-design with them. The co-design led to an increase in engagement and collaboration between the stakeholders. The stakeholders provided continuous feedback. The development team was also able to get requirements and knowledge about the CBT contents iteratively. The clinical researchers and psychologists were motivated in adapting the system for mobile devices and had the incentive to participate in the co-design not only to present their requirements and feedback, but also to get a possibility to reflect on their research studies. Furthermore, they could consider adaptation of CBT contents, such as videos, PDF and questionnaires, which was required due to the mobile adaptation. During the co-design, the development team also gained the trust of stakeholders as regards the design process and the adapted system. Thanks to the stakeholders' multi-disciplinarity, the development team received very creative ideas for solving various problems. The U-CARE management allocated time and other resources to this co-design process. The clinical researchers performed continuous beta testing, supported by automated recorded UI tests using Selenium. The design principles were revised based on learnings from this cycle (see Table 28) (Figure 15, pt. iii-j).

Table 28. Design principles for sustaining the usefulness of eHealth research software (version 2)

Design principle	Specification
The principle of engagement with stakeholders	The software developers of eHealth research software should continuously engage with stakeholders, in order to adapt the software in a direction which will satisfy stakeholders, given that the stakeholders are willing and committed to such engagement in the long term.
The principle of co-design with stakeholders	**The software developers of eHealth research software should co-design with stakeholders, in order to obtain continuous and early feedback, knowledge and requirements elicitation, gain trust, increase relevance and usefulness, and enhance creativity, engagement, and collaboration, given that there are incentives or motivation for stakeholders to participate in co-design and an availability of resources to make co-design possible.**

Artefact Use Over Time and Learning

The artefact improved over time, and the design process went through various changes, such as a) manual to automated deployment of the latest system release in the test and production environments; b) manual visual testing to automated testing using Selenium; c) manual builds and test execution to automated builds and test execution using TFS; d) backlog and sprint management using MS Excel to U-CARE backlog feature and later through JIRA; and e) source code version control from Subversion[78] to Git[79]. Various technologies were used and later discontinued, for example, the development team used LeanSentry to monitor the portal performance, Re-Sharper for static code inspection, and TeamCity for automated builds and test execution. Also, routines emerged regarding for instance the data extraction guidelines, professional secrecy, and a General Data Protection Regulation (GDPR) compliance policy.

[78] Apache Subversion (abbreviated SVN) is a software versioning and revision control system.
[79] Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development.

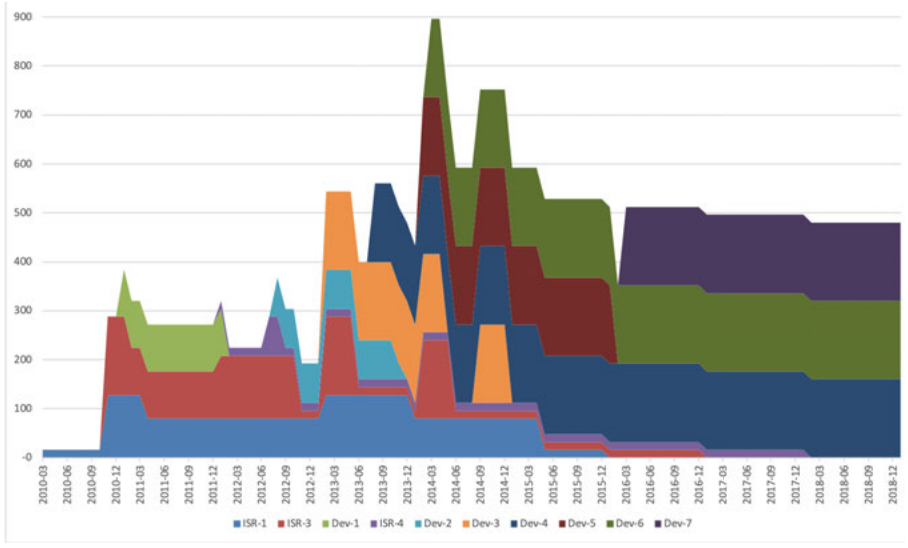


Figure 44. Approximate developer hours available monthly.

The development team increased from one to four members and then shrank back to three full-time developers. The Information Systems researchers also worked as developers. In addition, this stakeholder group included PhD students and Master's thesis students, either full-time or part-time, adding further variety to the development team. As the team grew, different roles emerged, for example, team leader, scrum master, support, and technical lead. The changing development team composition over time can be seen in Figure 44 (approximate developer hours available monthly are stacked). The inconsistent development hours, lack of continuity, and developer turnover were challenges in the U-CARE context. At several occasions, the development team fluctuated either in the number of developers or the team members' availability for development activities (e.g., 10%, 50%, 100%).

The most crucial problem was preserving the valuable technical knowledge of developers who were leaving, and ensuring a transfer of knowledge to new or existing developers in the team. The developers had invested a great deal of expertise and time in the U-CARE software system. The departure of a developer resulted in a loss of project-related skills and familiarity, leaving the development team unable to do their work. For example, when the technical lead left the development team, a lot of critical software development knowledge was lost, particularly related to systems architecture and design decisions. The Information Systems researchers' motto was *continuous improvement* in the artefact and the design process. The departing technical lead, being a design science researcher and team leader, had a significant role in the U-CARE context as regards innovative ideas, refactoring and, in particular, the malleable design of the U-CARE software system. Another developer left the team immediately after the mobile adaptation went live, resulting in a loss

of knowledge regarding mobile-related technologies and familiarity with the testing framework in the U-CARE software system. As a result, bug fixes, maintenance, and development of new features took significantly more time. Another example was the impending risk of knowledge loss related to the data export feature if the development team were to lose another developer.

The ease of designing new research studies led to a steadily growing system, and it took an ever-increasing amount of time to keep everything operational. The development team could not keep up with their daily workload. The development team did not have the time to fix longstanding problems with the U-CARE software system, much less redesign them to make the best use of new technologies. Due to limited development resources, the technical lead adopted a *break-fix* approach in line with the popular saying *if it ain't broke, don't fix it*.

The focus of the development team also changed over time. For example, Information Systems researchers led the development process initially, and the main focus was then on designing an innovative artefact and exploring the problem domain. When the artefact went into production, this changed the focus of the development team to development and operation (i.e., DevOps). They also worked with maintenance and providing support. Meanwhile, the focus of the Information Systems researchers shifted to follow-up and evaluation of the artefact.

Eventually, the development team's focus shifted back to innovation, for example, online consent, integration of BankID[80] (for two-factor authentication), and the addition of video chat and enhancement to the U-CARE software system for the ENGAGE (1000g) study. One key factor was that the U-CARE context had matured and the development team had become confident in making big changes to the codebase. The design process had also stabilised thanks to the appropriation of learnings from designing the U-CARE software system over time; having the same team members for an extended period; gaining a balance in the innovation, DevOps, support, and maintenance activities; and maturation of the requirements elicitation process.

The development team used the proposed design principles in practice. The effectiveness of the design principles was evident in that the design principles were actionable by the development team and instantiated into the U-CARE software system, which afforded the action described by the design principle. For example, *the principle of technological-ecological adaptation* was instantiated as presented in the following section.

The instantiation of the principle of technological-ecological adaptation

The test coverage of the U-CARE software system had increased. The development team automated the execution of repetitive tasks, like testing, integration, and deployment, which enabled them to solve the problems quickly by

[80] https://www.bankid.com/en/ [accessed: October 09, 2017].

detecting errors in the U-CARE software system more readily and locating them more easily. Selenium recorded tests were very rigorous and simulated the actions of various user roles in the system. Running the Selenium recorded tests minimised or even eliminated the need of clinical researchers' beta testing the U-CARE software system multiple times. Moreover, the automated process enabled running tests more often and more quickly than before, when tests were manually conducted by the clinical researchers, a process which was time-consuming and error-prone. The development team adopted JIRA, Git, TFS, and SourceTree[81] (Graphical UI Tool for Git), recognising the need for the tools in software development. They also acquired hardware, such as different mobile phones and computers, to test the U-CARE software system on real devices (instead of an emulator), with various operating platforms and web browsers. The development team also focused on upskilling, so that they learned about mobile app development-related technologies. For example, Dev-4 and Dev-5 spent their time learning mobile app development during the mobile adaptation, while Dev-6 learned through designing an app for the U-CARE ParentsCan study (i.e., the PUSSEL app, see Appendix E.12). Similarly, Dev-7 had to learn during enhancement of the responsive design for the desktop environment in the U-CARE software system (see Appendix E.11). It is important to note that the development team learned native mobile app development only recently.

One significant change was related to the development team moving from the existing product backlog/feedback feature[82] of the U-CARE software system to JIRA, and from SVN to Git. The development team was pleased with JIRA as it meant that they had one less feature in the system to support, while giving them a lot more functionality (Dev-4, 2017, Discussion). Similarly, the development team felt more productive with the implementation of Git:

> Moving to Git was successful. We are much more productive. We can work on different features seamlessly by shifting between different branches. Also, before merging code, we can see the differences between branches and possible conflicts, which saves the team a lot of time in removing conflicts on their branches before merging to the master branch. (Dev-4, 2017, Discussion)

Git enabled the development team to shift between feature[83], bug or hotfix branches quickly while working on new features as well as on maintenance tasks. Git also enabled them to act immediately by merging the hotfix to a

[81] https://www.sourcetreeapp.com/ [accessed: January 16, 2017].
[82] The backlog feature was essential to Information Systems researchers (to document the design process), and was made to cater their research needs, among others things. However, for the software developers, it was less useful than JIRA.
[83] Technically Git, being a distributed version control, supports the use of multiple (feature. or design epic.specific) branches on the local computer of a developer and switching to a different branch is easier than in other version control systems.

stable branch (which was the same as the live production version) and deploying code to live production without waiting for the hotfix to be pushed during a scheduled deployment. Git enabled knowledge sharing across team members during development.

The development of the U-CARE software system was a long project, encompassing diverse functionalities, multiple ongoing research studies, and enormous amount of research data. In the struggle to meet the continuous need of building new features, proactive software development practices (e.g., TDD) were sacrificed and became a non-priority. However, the development team eventually caught up on writing tests and were able to increase their test coverage. *The principle of embracing proactive practices* was instantiated as presented in the following section.

The instantiation of the principle of embracing proactive practices

The development team realised that they had to write code that could be intrinsically maintainable. The development team observed good coding practices such as code review and documentation (both in code and separately) for better code readability; refactoring, and modular coding for simplicity and extensibility; and estimating efforts for writing tests and dealing with technical debt while estimating the user story points for testability. JIRA enabled the development team to visualise the product backlog. This visual backlog board provided the development team with a streamlined development process, including what needed to be done and what needed attention. Additional tools were also used in the improved design process, such as Fisheye[84] (to view code changes side-by-side in the integrated development environment and JIRA) and Confluence[85] (a wiki for sharing information). The development team insisted on a well-defined scope during IT meetings. They also separated the code committing and code integration with the production branch, so these were no longer performed by the same developer. The code committing of a feature created by one developer was integrated by another. Hence, quality assurance was achieved through separating contribution from integration[86]. The U-CARE software system's bi-weekly monitoring report was another proactive practice embraced by the development team. This report not only provided insights into the existing system state, but also gave stakeholders trust in the system and the ability to take prompt actions regarding ongoing RCTs. The monitoring report consisted of information such as a) total research participant logins; b) time from SMS code receipt to successful login (to identify issues regarding login); c) number of research participant logins using mobile devices; d) issues related to reminders, if any; e) CBTs offered;

[84] Visualises and reports on activity and searches for code commits, files, revisions, or teammates across it. https://www.atlassian.com/software/fisheye [accessed: January 16, 2017].
[85] https://www.atlassian.com/software/confluence [accessed: January 16, 2017].
[86] In open source projects, it is a proven practice to have dedicated code review and to separate contributors and committers.

f) suicide risks; g) welcome e-mails; and h) observation point completions. The team leader regularly monitored the product backlog to balance between user requirements with the technical debt. The team leader (and product owner) also realised the limitations of the small development team and provided flexible deadlines on feature development, allocated fifty percent of the team's time to proactive practices, and provided more development resources, such as tools, hardware, servers, and infrastructure services. In this way, the development team embraced agility without compromising quality. As a result, the aforementioned software development practices contributed to mitigating issues related to sustaining the usefulness of the U-CARE software system.

The lessons from the instantiations of the two design principles above were that the focus on state-of-the-art tools and best (proactive) practices alleviated many problems that software developers faced when sustaining the usefulness of eHealth research software. There were instances of instantiations of the two design principles above during the early stage of the U-CARE software development, such as:

> The endeavour to have a well-reflected architecture and rigorous testing is related to a high initial cost (in terms of time), while at the same time it aims at (i) promoting the maintainability and quality of our software product and (ii) increasing the development speed in future sprints. (ISR-1, IT meeting minutes, 2010)

The U-CARE software system was refactored from a .NET web application into an MVC-based one and changes were made in the architecture to allow for automated unit testing. At that time, the short-term productivity was hampered due to the allocation of resources to educate the development team (as two out of three members were inadequately skilled in MVC) and to implement automated unit tests (which required design changes and a team member focused on testing, as well as educating the team in TDD). However, in the long term, adoption of new technology and proactive practices (an upfront investment) had a positive impact on sustaining the usefulness of the U-CARE software system.

7.3 Formalisation of Learning

Health care implements only evidence-based interventions and the evidence takes a long time to accumulate. This creates a particular difficulty in eHealth interventions, where technological development is fast. At the time when a sufficient level of evidence is reached, the technology might be outdated or there may be a better one available (Glasgow *et al.*, 2014). During an ongoing

RCT, the intervention must be locked down for evaluation. Such locking down of interventions reduces the opportunities for adaptation to the changing technological environment (Mohr *et al.*, 2015). This was a challenge in the U-CARE context, considering its goal of implementing the successful interventions in regular care (Grönqvist *et al.*, 2017). Mohr *et al.* (2015) conclude that there is a need for clinical evaluations to keep pace with the level of innovation in eHealth. This case shows the successful implementation of a technological change in the U-CARE software system in parallel with ongoing RCTs, while minimising the risks associated with in-trial changes. This success was due to the participation of all U-CARE stakeholders in the co-design activities, the efforts of the development team in streamlining the design process, and the adoption of the proactive practices. The case shows that the quality characteristics *simplicity* and *embedded in design system* are the key to sustaining the usefulness of eHealth research software in the academic research context, as the eHealth research software was used by many stakeholder categories and large numbers of individuals.

Groen *et al.* investigated the relation between development practices and the size of a development team, observing that: "new practices are typically adopted when a development team has recently increased in size [… and were] slightly reduced when the respective development teams became smaller" (2015, p. 16). In U-CARE, the development team size fluctuated over time (see Figure 44) and it was observed that this had a similar effect on the adoption of best practices. Groen *et al.* (2015, p. 12), while discussing the difficulties of achieving a consistent application of best practices in an academic research context, stated that "the extent to which best practices are applied depends strongly on personal commitment of the individual developers." This was observed also in the U-CARE context.

Lessons learned from this case were that proactive practices, such as continuous refactoring, strict TDD[87], unit testing, better test coverage[88], code review, pair programming, continuous integration and documentation helped the development team to deliver faster and to sustain the usefulness of the eHealth research software. However, it is important to allocate resources for such proactive practices. For example, training, learning, and trying new technologies impacted the software development team's velocity, but costs associated with upskilling the team paid off in the long run. The development team

[87] It was evident in Chapter 6 (Case II) that when the development team's quality assurance practices did not match to their development pace, their agile became fragile. This was due to the pressure to develop the U-CARE software system faster. They learned that without strict adherence to TDD and creating and maintaining a test safety net, their incremental design might never be realised. The development team needed to make sure that they had a solid suite of tests (a.k.a., a safety net) before refactoring (Fowler et al., 1999; Fowler, 2018).

[88] For example, Code Refinery (an organisation that provides training and e-infrastructure for research software development – https://coderefinery.org) suggests that one should "not trust a research software if: a) its tests do not cover its claimed capabilities (test coverage); b) its tests do not pass; c) there are no tests at all; and d) the tests are never run."

needed to fully understand the libraries and frameworks they used in software development. Hence, they needed to practice in a safe-to-fail or proof-of-concept environment before making changes in the production environment. Similarly, they required support tools to track their progress and receive fast feedback from end-users.

The development of eHealth research software in an academic research context was a long-term endeavour and required additional resources to support a small development team to follow best practices, as was evident in the U-CARE context. As a consequence, the clinical researchers (particularly the principal investigator) needed to be aware of recurring and ongoing costs associated with eHealth research software development and sustaining its usefulness over time. Additionally, they also needed to consider hiring a development team with an adequate number of developers (preferably from the start), to keep the team consistent for an extended period, and plan for the development team turnover in advance. The clinical researchers (particularly the principal investigator) also needed to be aware of that continuous training of the development team was essential for an acceptable development process when agile methods were used. Such continuous learning and training had a positive and significant impact on team productivity. The learnings from this case are articulated and formalised as design principles in the following table.

Table 29. Design principles for sustaining the usefulness of eHealth research software

Design principle	**Specification**
The principle of engagement with stakeholders	The software developers of eHealth research software should continuously engage with stakeholders, in order to adapt the software in a direction which will satisfy stakeholders, given that the stakeholders are willing and committed to such engagement in the long term.
The principle of co-design with stakeholders	The software developers of eHealth research software should co-design with stakeholders, in order to obtain continuous and early feedback, knowledge and requirements elicitation, gain trust, increase relevance and usefulness, and enhance creativity, engagement, and collaboration, given that there are incentives or motivation for stakeholders to participate in co-design and an availability of resources to make co-design possible.

In conclusion, as in the context of wicked problems, an essential aspect of the design process is that the problem domain is better understood over time through the design activities taking place (Gregor & Hevner, 2013). Following ADR, the contributions to theory (including empirical) and practice are formulated as they emerged in the longitudinal design and use narratives in Chapters 5, 6, and 7. In summary, design principles and quality characteristics for sustaining the usefulness of eHealth research software in an academic research context were identified through the empirical instantiation of ADR in

the U-CARE context which contributed to practice by addressing real-life organisational problems, concerning the design and use of the data export feature, the process of technology adaptation due to continuous technological-ecological changes in the design landscape, and the adaptation of the U-CARE software system to mobile devices.

Similarly, research interest or a class of problems may also be identified over time through reflections on design experiences (Sjöström, 2010). In the following chapter, the retrospective analysis is presented.

Part IV: Analysis and Reflection

8 Retrospective Reflection and Learning

In this chapter, the analysis and reflection are presented in detail. First, in Section 8.1, the details of how the retrospective analysis was conducted are presented. This section also discusses design principles, quality characteristics, and typology for sustaining the usefulness of eHealth research software in an academic research context. A reflection on the ADR research method is presented in Section 8.2. Lastly, in Section 8.3, ADR across multiple cases is presented, including augmented action design research (AADR), augmented reflection and learning (ARL), and appropriation of ARL in this dissertation.

8.1 Retrospective Analysis

The problems perceived during the multiple ADR cases, presented in the previous chapters, were retrospectively reinterpreted and mapped to the class of problems. The class of problems, for which research in this dissertation aims to generate knowledge, is *sustaining the usefulness of eHealth research software in the academic research context*. Following the advice of Gill & Hevner (2013), I looked back over time and analysed the past in order to trace the evolution of the U-CARE software system (from its inception via design and construction, implementation in U-CARE context, to the stage when it was almost ready for transition into real-world operational environments).

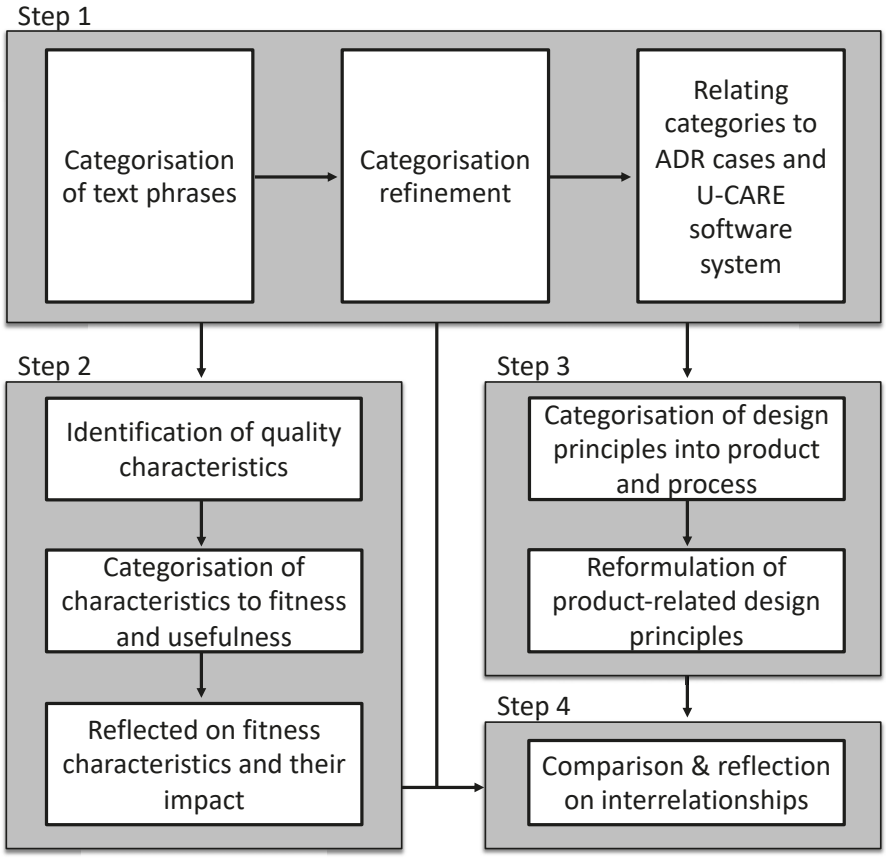


Figure 45. Retrospective analysis process.

The retrospective analysis was conducted in four steps as shown in Figure 45. In **step one**, based on the broad class of problems and research questions, the text phrases were sorted into categories, for example, activities, design decisions, events, and features, which had been inductively identified and revised during the analysis. This process was followed by axial coding to relate categories to sub-categories, for example, activity – development, activity – documentation, design decisions – technology-centred, design decisions – client-centred, feature – authentication, feature – backlog, et cetera. The categories and sub-categories were grouped based on ADR cases (e.g., data export feature and mobile adaptation) or with a basis in the entire U-CARE software system. These categories helped in understanding the design processes.

During the prolonged engagement in the U-CARE context, I was not only involved in my own ADR cases, but also in the development of multiple features and participatory observations (2012–2017). This prolonged engagement allowed me to reflect on the overall U-CARE software system and the design process. The field notes (a.k.a. the research log) contained continuous

on-the-spot rich records of reflections and learnings. There were occasions when I was not part of the empirical context (2010–2011) or was not directly observing (2018–2019). In such cases, additional data were studied, for example, IT meeting minutes, product backlog history, code repository, et cetera. Beyond the data collected during my active time in the project (2012–2017), additional data sources were used in the analysis (see Table 7 for details). Step one resulted in reconstructing the whole design process, establishing a chain of evidence and understanding the steps through which the design of the U-CARE software system had evolved.

In **step two**, the quality characteristics were initially identified inductively and labelled based on characteristics mentioned in ISO/IEC Standard 25010:2011, for example, characteristic – testability, characteristic – reusability, characteristic – modularity, et cetera (see Appendices A.1 and A.2 for a full list). Later, based on the fitness-utility model (Gill & Hevner, 2013), the quality characteristics that contributed to sustaining usefulness were grouped as fitness characteristics while others were grouped as usefulness characteristics, for example, fitness – malleability, fitness – openness, usefulness – testability, usefulness – usability, et cetera. In this way, quality characteristics that had impacted on sustaining the usefulness of the U-CARE software system were identified systematically, and theoretically validated.

In **step three**, learning across multiple ADR cases was synthesised to generate more abstract design knowledge. The objective was to have a broad enough level of abstraction to enable researchers and practitioners in other contexts to enact the design principles while designing their research software. The abstraction process started with combining the designing principles formulated across three cases.

In **step four**, in order to establish the relationship between the design principles and quality characteristics, a typology of sustaining usefulness and (re-)construction thereof concluded the retrospective analysis.

Overall, in the retrospective analysis, I closely followed the principles for interpretive research (Klein & Myers, 1999, see also Section 4.3). Table 30 shows how the principles were used in the present work.

Table 30. Appropriation of Klein and Myers' principles

Principle	Appropriation in this dissertation
1. The fundamental principle of the hermeneutic circle	The focus shifted between details (e.g., actions, events, design decisions, quality characteristics, design principles, design process, design artefacts and features) and the understanding of the phenomenon as a whole that sustained the usefulness of eHealth research software. The retrospective analysis was based on the iterative process of moving around data, concepts, and categories.
2. The principle of contextualisation	The understanding of the phenomenon was further increased by investigating the historical background of the empirical context. Data were retrieved and analysed from the very inception of the U-CARE research programme (e.g., IT meeting minutes, source code, documentation, et cetera). The original motivation of U-

	CARE stakeholders enabled me to better understand how the phenomenon of interest emerged.
3. The principle of interaction between the researchers and the subjects	During the ADR cases, data were collected and interpreted together with the ADR team. For example, IT meeting minutes and design workshop reports represented an overall and shared view of stakeholders. U-CARE being an academic research context, was very transparent and open. Hence the data and interactions between me and the stakeholders were also very transparent and open. However, interviews, discussions, observations and field notes were of sensitive nature, making ethical and privacy considerations necessary. As I was an ADR researcher and an insider, additional steps were taken in data interpretation and data presentation, as discussed in Chapter 4. For example, due to my familiarity with the context and data, another Information System researcher was involved in interpretation to remove bias (prejudices) in the analysis. Additionally, the research log was useful for building a chain of evidence to reduce my bias.
4. The principle of abstraction and generalisation	The design principles were abstracted based on both empirical findings and theoretical understanding through a fitness-utility model and quality characteristics. The emergence of the principles and the typology, and their interrelationships, shows the progressively more generalised and abstract concepts to support designing sustainable research software.
5. The principle of dialogical reasoning	The identification of quality characteristics in the empirical context was based on a continuous cycle between the guiding fitness-utility model (theoretical basis) and the actual empirical findings. This is apparent in differences between the proposed quality characteristics and an existing list of candidate fitness characteristics.
6. The principle of multiple interpretations	The data was triangulated through multiple sources of evidence for integrating multiple interpretations into a coherent understanding of the phenomena. Examples include interviews, participatory observations, field notes, IT meeting minutes, source code comments, code commit history, design workshop reports, system logs, product backlog, developers' notes, developers' diagrams and informal discussions. The secondary data enabled triangulation of data during the analysis. Discussions with stakeholders, particularly with Information System researchers and practitioners, were conducted throughout the research project to ensure the validity of the interpretations from various stakeholders' points of view.
7. The principle of suspicion	The data were carefully interpreted recognising stakeholders' different, sometimes conflicting, views based on their interpretations of the evolving context. For example, the development team members revealed differing views of the U-CARE software system at different interviews. Thus, *why* different views were expressed was also considered during data analysis. Prolonged engagement and persistent observations in the empirical context facilitated data interpretation. Additionally, I constantly reflected about the empirical context and kept notes of my reflections in the research log. The research log was also used for data interpretation.

Before the retrospective analysis begins, here is a quick summary of motivation, aim, design principles and quality characteristics identified in the ADR cases.

Table 31. Summary of three ADR cases

#	Case	Description
1	**Data export feature**	
	Motivation	Data export is a crucial functionality for eHealth research software in an academic research context. The clinical researchers need to export the data for analysis, to interpret research results and draw conclusions. The data export feature design started at the U-CARE forming stage.
	Aim	The case aimed to develop design principles and quality characteristics for data export in eHealth research software in an academic research context.
	Design principles	The principle of simplicity The principle of modularity The principle of malleability The principle of accountability
	Quality characteristics	Malleability, decomposability, simplicity, and accountability
2	**Technology adaptation process**	
	Motivation	Software developers have to cope with a continuously changing design landscape, due to changes in user requirements, the organisation and the environment, while designing eHealth research software in an academic research context with limited resources. The technological-ecological changes posed challenges for software developers and required attention in the design process through the U-CARE maturing stage.
	Aim	The case aimed to develop design principles and quality characteristics, to guide the design process, and to support a continuously changing design landscape in an academic research context.
	Design principles	The principle of technological-ecological adaptation The principle of embracing proactive practices
	Quality characteristics	Malleability, decomposability, simplicity, and accountability
3	**Extending the artefact**	
	Motivation	Technological innovations in the surrounding environment can affect the usefulness of eHealth research software. eHealth research software's access/availability is essential for the end-users (i.e., research participants in a research context). In the matured stage of U-CARE, several research studies were ongoing. Extending the artefact to mobile devices increased reach toward research participants.
	Aim	The case aimed to develop design principles and quality characteristics, to guide the design process, and to support a continuously changing design landscape in an academic research context.
	Design principles	The principle of engagement with stakeholders The principle of co-design with stakeholders
	Quality characteristics	Simplicity and embedded in the design system

In this dissertation, the fitness-utility model (Gill & Hevner, 2013) is considered a point of departure for the evaluation method, and the presented set of candidate fitness characteristics was found to be useful in the U-CARE context. In the following, I will reflect on the quality characteristics by means of a retrospective analysis.

Quality Characteristics

The stakeholders of U-CARE observed and perceived the following as the essential characteristics that impacted on sustaining the usefulness of the eHealth research system: decomposability, malleability, openness and embedded in design system. Also, simplicity and accountability were prevalent in the empirical context, but they are not explicitly addressed in the utility-fitness model (ibid.). Quality characteristics, such as novelty, interestingness and elegance were perceived as important considering the U-CARE software system's appeal to various research groups, funding agencies, research participants, and academic journals. However, there were only a few traces found in the empirical context; thus, these quality characteristics require further investigation on if and how they impacted on sustaining the usefulness of the eHealth research system. Decomposability, malleability, embedded in design system, simplicity, and accountability were explicitly identified during the ADR cases (see Table 31) and retrospective analysis validated them further. However, openness was identified during the retrospective analysis.

The retrospective analysis drew on the empirical material, for example, looking at the IT meeting minutes content for traces of fitness characteristics, their emerging patterns and their impact at later stages during the evolution of the artefact. The quality characteristics of eHealth research software identified as likely to impact on sustaining its usefulness in the academic research context, based on the synthesis of learnings gained through the retrospective analysis, are presented in detail below. As an illustration of each particular quality characteristic identified, how it came into play and affected the U-CARE software system and design process is exemplified and discussed for each case. The quality characteristic is then discussed in relation to the full system context.

Decomposability

Any system tends to evolve from nearly[89] decomposable subsystems (Simon, 1996). A system that is composed of independent subsystems, modules or components is easier to construct, since work on individual parts can be conducted separately (Gill & Hevner, 2013). When a system is designed using a

[89] Herbert Simon (1996) explained that in the complex systems there is always some interdependence between independent modules, despite the designer's effort to fully decompose the system, what he calls near decomposability.

number of independent components (or building blocks) the "chances that a particular component will evolve into new useful future versions increase" (Coenen *et al.*, 2015, p. 4034).

In case I, the generic data export feature exhibited the decomposability characteristic. The decomposable generic data export feature was made up of many independent modules, for example, the reflection utility and the export utility. The export utility module was reused in the one-click data export feature. The decomposability of the generic data export feature increased its capacity to evolve into a useful future version (i.e., one-click); without decomposability it would have been an all-or-nothing affair. It can be argued that the data export feature construction using the proposed conceptual architecture of two-stage periodic data export would have been simpler thanks to its two-stage decomposability.

In case II, an example of decomposability of the U-CARE software system, can be observed in the form of a layered architecture. This allowed changes related to the technology upgrade to be limited to one layer (i.e., the presentation layer). During case II, the testing framework was also improved to facilitate the design and construction of advanced tests of the U-CARE software system. The development team divided testing framework into recorded tests (RecTest), programmed tests (ProTest), generic tests (GenTest), and database consistency tests (DbTest).

The decomposability of the artefact, in the form of a layered architecture, also facilitated the adaptation to mobile devices in case III. Although changes happened in the multiple layers in case III, they were easier to manage thanks to decomposability. The analysis of the source code revealed that there were very few changes in the business logic layer once it was mature (around halfway through 2015). The testability of the software system enabled a smooth transition in case III.

Most importantly, the core business logic of the U-CARE software system was divided into individual modules, such as randomisation, intervention, indicator, et cetera. This enabled a much simpler unit testing of core modules, as they were separately testable (i.e., independent from the testing of the entire system) and could be tested in isolation from other system modules. The modules were well devised, cohesive, self-contained, fully tested, and each with a distinct function. Modular design, conversely, facilitated loose coupling and made the source code easier to maintain.

Looking at the three cases combined, it is evident that the development team had given weight to decomposability in different ways at different stages of the development process. The level of decomposability also varied. In case I, it was the decomposability of feature that was of significant importance, whereas it in case II was that of the new technologies. Still, the system's overall decomposability facilitated the design process in all three cases. In conclusion, the decomposability quality characteristic was found to affect the design

artefact greatly as regards the component and system level, the design product and design process level, and the design and use.

Malleability

Malleability refers to the ability of the users to mould the artefact to their needs. The malleability of an artefact represents the degree to which it can be adapted by its users and respond to changing environments and user needs (Williams *et al.*, 2008; Gill & Hevner, 2013). If the artefact is malleable, the chances are higher that it will survive in future generations, as users adopt it and adapt it to their needs (Coenen *et al.*, 2015). Malleable artefacts, that are designed for multiple contexts of use, can also satisfy diverse user groups (Lund, 2014). An artefact with high malleability means changes are easier, with less risk and lower expenses (Williams *et al.*, 2008). The context-specific adaptation is also reflected as *mutability* and considered an important component of design theory in Information Systems (Gregor & Iivari, 2007; Gregor & Jonas, 2007). Gill & Hevner (2013) proposed three levels of user-malleability, i.e., customisation (the ability of an artefact to be tailored to a user's preferences), integration (the ability to conveniently share the capabilities of one artefact with another), and extension (adding new capabilities to an artefact). Through customisation, the users can modify the default version of the artefact into something that fits their needs, in effect creating a new instantiation of the artefact (Coenen *et al.*, 2015). Sjöström *et al.* (2011) used configuration (adapting the artefact to new situations), instead of customisation, to denote user-malleability. They suggested that configurable features (in contrast to hard-coded features) made a system mutable.

In case I, the early design of the data export feature was generic and configurable by the clinical researchers themselves. The main idea was that the data export feature should be useful in new situations insofar as possible, without further software development involved beyond user configurations. However, maintaining the generic data export feature turned out to be a challenge. Nevertheless, the user-malleability was considered in the proposed conceptual architecture of the two-stage periodic data export feature: for example, data filtering and a data export template would enable the clinical researchers to customise their data export requests. Privacy, security, and accountability were managed for the data export feature through configuring the malleable authorisation feature of the U-CARE software system.

In case II, the technology modernisation indirectly led to enhancement of the artefact malleability through the malleability of the design process, for example, minimisation of JS and CSS files based on configurable settings to activate environments like production or debug. The Razor view engine allowed the development team to develop different UI themes for the system.

In case III, responsive design using the Bootstrap CSS framework, CSS3 and HTML5 enabled flexibility in layouts, image sizes, text blocks, et cetera. This flexibility combined with smart use of CSS media queries resulted in the

malleability of UI fluidity, which enabled for the U-CARE software system to adapt to fit any container, based on mobile device screen sizes.

Several features of the U-CARE software system were designed for malleability and offered configuration possibilities for U-CARE stakeholders, such as software developers and clinical researchers. Malleability (a proactive design effort by Information Systems researchers – an upfront investment) enabled for the U-CARE software system to be configured to the design of any RCT study within clinical psychology (arguably even in other related contexts) (Sjöström *et al.*, 2011). In the U-CARE context, a large number of stakeholders in the design process increased the complexity in making sense of customer (i.e., the clinical researchers) needs, as they intended to conduct multiple and diverse studies. As did Carroll (2004), I observed that a malleable design could be a means of overcoming the inability of stakeholders to articulate their needs and determine the requirements on the system amid such complexity. The clinical researchers were provided with a malleable system which they could adapt to meet their needs. However, the actual needs, which were more refined than the initial requirements, were revealed over time when the system was used by clinical researchers and other stakeholders.

The system malleability led to extending the U-CARE research programme to that of a service provider by allowing other research groups and researchers to design and run their clinical studies through the U-CARE software system. On one hand, funds received from associated research groups for using the system enabled the U-CARE management to use additional development resources, for example, one full-time developer. On the other hand, the execution of more research studies resulted in getting more funding, credibility, and (re-)use in multiple situations. Indirectly, the malleability of the system aided in sustaining the U-CARE software system.

In case III, the possibility to configure research studies enabled switching of the mobile-adapted UI through *theme* settings to suit the target mobile devices of research participants. Also, theme settings allowed some studies to enable a mobile-adapted UI, while others could retain a desktop-adapted UI. Another example could be found in the malleable design of psychological interventions, which allowed the psychologists to set up mobile adapted interventions using existing interventions through a sort of *save as* functionality. This allowed the easy configuration of a new mobile-adapted intervention by adapting the contents only. Similarly, a questionnaire design tool, which allowed clinical researchers to create custom questionnaires, enabled the saving of questionnaires in mobile-adapted versions which could subsequently be redesigned with minimal changes in the question design. Likewise, the translation module malleability allowed for quickly fixing any translation errors.

A final reflection is that the U-CARE software system was malleable, which allowed it to survive, as the clinical researchers adapted and tweaked it to their needs. However, the malleable design had consequences for the development team in maintaining the system. One example that was observed

was related to configuration management. It was found that the configuration was different in the production system than in the developers' system and on the test servers. This meant that the system had a different behaviour based on the database state in different working environments. In turn, this had implications for the operational management of the system; for example, the work of developers did not end upon publishing code to production; they also had to examine the configuration for consistency. Furthermore, to test any bugs, they had to establish, on their local workstations, the same configuration that the server had. In the U-CARE software system, configurations were stored in the database; thus, synchronisation (at least the configuration data) of the production database with a local database was required for establishing data consistency. This synchronisation of databases, in turn, had implications for privacy and accountability (which are discussed under the *accountability* characteristic).

The U-CARE software system was built so that the clinical researchers could design and configure their studies and questionnaires themselves. Also, the overall U-CARE software system was designed so that it could be (re-)used by other U-CARE stakeholder categories, for example, those working with associated studies. However, if we, as design researchers, had designed the system in such a way that other system designers could (re-)use it for designing their systems, maybe we should have gone to the API or Platform concept (which was discussed while adapting the U-CARE system for mobile devices in case III).

In conclusion, decomposability and malleability appear to be of similar importance in sustaining the usefulness of the U-CARE software system in all three cases.

Embedded in design system

Gill & Hevner (2013) argue that when artefacts are part of systems where design and changes are common, it can be expected that they evolve more rapidly than when design and changes are uncommon. They suggest that "a *design system* can also manifest itself as a community of users and designers, providing contributors with intrinsic motivation to contribute" (ibid.). In U-CARE, from its inception, the design of various artefacts, such as the design of research protocols (research studies and trails), patient treatment (CBT interventions), and the technology to provide treatment and do research online (the U-CARE software system), was initiated by the community of multi-disciplinary researchers. This community established a *design system* in which incremental but regular changes took place in the U-CARE software system, as well as through continuous improvements in the design process. The design processes were feeding off each other; for example, when the development team designed a part of the system and demonstrated it to the clinical researchers, who were working on designing the research studies, the latter either realised that this was not what they meant or suggested to the development team

to do it in another way instead. Similarly, the development team gave feedback on the treatment design. Changes in the treatment design led to changes in technology and vice versa. The U-CARE software system (the artefact) being embedded in this *design system* fostered the long-term survival of the system.

The Information Systems researchers wanted to have domain knowledge regarding the area for which they were building the software. Being developers, they always came up with different innovative ideas as they were familiar with the software and knew what it could do for the clinical researchers. There were countless collaborations between the development team, the clinical researchers, and other stakeholders, as to what should be built. These groups talked about all aspects of software development. It was not a one-way feeding of requirements. Instead, it was a two-way collaboration (Section 3.5 provides a detailed description of the design process and an emerging *design system* in the U-CARE empirical context).

In case I, anticipating the continuous design of intervened design artefacts (the study protocols, the interventions and the system), the data export feature was developed to be generic in order to adapt to the changing U-CARE software system. In case II, it was observed that the stakeholders, such as software developers, clinical researchers and psychologists were engaged, but to lesser extent than before, in the *design system*, which was one of the reasons for the decline in the functioning of the U-CARE software system.

During case II (2014–2015), the U-CARE software system was in a development and operations (DevOps) working mode, which resulted in fewer team members in the development team able to take part in working with the *design system*. Likewise, most of the Information Systems researchers were moved from artefact design studies to follow-up and evaluation studies. The clinical researchers were initially engaged in designing pilot studies and then (re)designing full-scale studies based on pilot study results. Then, the clinical researchers moved from design mode to research study execution mode, which resulted in less activity in the *design system*. The psychologists, who had been part of *design system*, also changed their focus, from designing CBT treatment and creating CBT contents, to delivering the treatment to research participants. On the whole, the stakeholders remained active in the *design system*, but grew steadily less engaged. From 2016 onward, the development team also had to provide support, maintenance, and monitoring of the U-CARE software system.

During case III, all stakeholders were once again engaged in the *design system*. The U-CARE software system, as well as the software development resources, were in a stable state, which allowed the development team to take part in the *design system*. The clinical researchers were engaged in redesigning studies to adapt them to mobile devices. Similarly, the psychologists were engaged in adapting the treatment (CBT) and its contents to be delivered on

mobile devices. The clinical researchers also designed new studies; for example, those working on U-CARE ParentsCan considered adapting their RCT to be used on mobile devices from the start. Information Systems researchers who were interested in mobile adaptation also engaged in the design of the U-CARE software system.

Additional important lessons from the retrospective analysis of the artefact being embedded in the *design system* include that the stakeholders remained engaged as long as there was something for them to interact with, preferably something visual, such as an early version of the artefact, a test coverage report, a system status report, et cetera. Another lesson from the retrospective analysis revealed that the addition of lived experience representatives also strengthened the *design system*; for example, young people with lived experience of cancer. Overall, U-CARE involved people with lived experience in developing and testing interventions. The participatory approach affected the development of the U-CARE software system by considering the user experience when designing new features.

In conclusion, based on the retrospective analysis, embedded in the *design system* appeared to be the most important characteristic of all in sustaining the usefulness of the U-CARE software system.

Simplicity

Simplicity is the degree to which an artefact has a straightforward and easy to understand (comprehensible) design and implementation (ISO/IEC/IEEE, 2017). The definition of simplicity was adapted by Prat *et al.* (2015) as "the degree to which the structure of the artefact contains the minimal number of elements and relationships between elements." The key components of simplicity in a software system are code simplicity (following a coding standard), structural simplicity (modular architecture), and functional simplicity (minimum necessary to meet the requirements) (Pressman & Maxim, 2014). Generally, innovations that are simpler to understand are adopted faster than innovations that require the adopter to develop new skills and understanding (Rogers, 2003). Beck and Andres (2004) mentioned simplicity as one of the five values at the heart of the extreme programming (XP) agile method. In XP, simplicity refers to keeping the design of the system as simple as possible, so that it is easier to maintain, support, and revise.

Gill & Hevner (2013) also mentioned simplicity while associating it to the elegance characteristic. They related the elegance to the artefact's form (i.e., aesthetic elements such as appearance) only, whereas simplicity in the software engineering literature, as mentioned earlier, is related to both form and function. As observed in the U-CARE context, the quality characteristic of simplicity was considered to be twofold: concerning simplicity as an aesthetic quality of the artefact (form) and simplicity as a design quality embedded in the artefact (function). Simplicity is important in relation to decomposability as described by Parnas:

> The system is divided into a number of modules with well-defined interfaces; each one is small enough and simple enough to be thoroughly understood and well programmed. (1972, p. 1054)

In case I, the generic data export feature was developed considering the mutability and relative malleability of the technology for a more scalable solution. It was disheartening to see the generic data export feature fell into disuse. As a result, the development team started to pay more attention to design fitness issues, to complement the existing focus on usefulness. Hence, the development team chose simplicity over scalability during development of the one-click data export feature. Also, the UI of the one-click data export feature was very simple as compared with the generic data export feature.

In case II, the development team had carried out multiple refactoring of the U-CARE software system to simplify it through reduction of complexity, documentation, decomposition to multiple layers, modularity and use of proper coding practices to enhance code readability, code syntax cleanness, reusability, and maintainability. As a result, the efficiency and performance of the system increased.

In case III, the development team kept the design simple but attractive while adapting the U-CARE software system to mobile devices. For example, there were multiple scenarios in the system where the research participants had to choose either 'save,' 'submit' or 'cancel.' It was easy for the research participants to get confused and click on the *save* button, thinking that it would submit their information as well. To keep things simple the development team removed the *cancel* button and gave two options to the research participants, to click on either 'submit' or 'save and continue.' Similarly, the layout of the menu, header, footer, questionnaires, library, forum, chat, et cetera, was made simpler and more intuitive.

The important lesson related to building a simpler artefact was that it required the development team to always keep the code as clear and simple as possible. The development team had to balance malleability and simplicity. It was observed that if an artefact was malleable, it could be too difficult for newcomers to learn as only expert users could comprehend the malleable artefacts. Conversely, if an artefact was simple, it was easy for users to understand and follow, as it could only be used for a single purpose.

In conclusion, based on the retrospective analysis, simplicity appears to be important in sustaining the usefulness of the U-CARE software system, but achieving it in the U-CARE context required significant extra resources (e.g., experienced team members[90]).

[90] Team members experienced in the redesign and big refactoring to re-architecture the U-CARE software system, considering ideas discussed within team, such as the use of reusable stand-alone components, platform, API, microservices, et cetera.

Accountability

The accountable artefact is defined as an artefact that adheres to information accountability. Information accountability has attracted the attention of Information Systems researchers as one among several desirable properties of design artefacts (Pearson & Charlesworth, 2009; Boos & Grote, 2012; Sjöström, Ågerfalk, *et al.*, 2014; Sjöström *et al.*, 2017). Weitzner *et al.* described information accountability as follows:

> The use of information should be transparent, so it is possible to determine whether a particular use is appropriate under a given set of rules and that the system enables individuals and institutions to be held accountable for misuse. (2008, p. 84)

System and software quality models (ISO/IEC Standard 25010: 2011) define accountability as the "degree to which the actions of an entity can be traced uniquely to the entity." Accountability is one of the most important aspects of the health care industry. In an eHealth research context, accountability concerns information privacy and avoiding misuse of research participants' information (Sjöström, von Essen, *et al.*, 2014). Similarly, in the academic research context, accountability is conceptualised as being answerable to the academic community (e.g., ethical approval boards, academic journals, government agencies, funding agencies, et cetera) in conducting research in an ethical manner. eHealth research software deals with particularly sensitive personal information; thus, it needs to account for privacy, confidentiality, integrity, and protection of sensitive data. It is particularly evident in the eHealth research context that accountability is a factor that one would expect to be necessary in an eHealth system to ensure privacy and transparency in dealing with sensitive data. The eHealth research software requires the implementation and use of technological measures to provide accountability and enable audit organisations to confirm compliance with legislation and ethics.

In the U-CARE context, researchers were also aware of and complied with the legal and ethical aspects of the management of research participants' data. The U-CARE software system provided double (two-factor) authentication, role-based privileges (i.e., authorisation) to access information and monitor and log all research participants' information-use events (establishing audit trails) relevant to the assessment of accountability. Also, there were information accountability mechanisms in place at the organisational level, for example, security and privacy breach monitoring and auditing to ensure compliance with existing regulatory requirements.

In case I, the data export feature was made to adhere to information accountability by restricting access and use of research data (e.g., on a need-to-know basis). Regulatory requirements, for example, data privacy, transparency, and accountability, required that data export events were logged to en-

able regulatory audits and compliance reporting. The U-CARE software system's authorisation and logging features enabled accountability and traceability of data export events, in addition to other uses. Data export feature logged information on data export events, i.e., when (timestamp), who (user identity – role), and what (data specification), to facilitate follow-ups by the study owner. The proposed two-stage periodic data export architecture would further extend accountability to the data export of reminders, logs, RBL, and dashboard reporting of data.

In case II, the technology upgrade had consequences in sustaining the accountability characteristic. Accountability required provenance of data, that is the history or record of transactions performed on a data object. In the U-CARE software system, the log must be kept to reveal the provenance of the system use and to ensure accountability. Thus, during the technology adaptation process, the development team invested efforts in keeping the log feature intact. The development team often needed to synchronise the production database with the local database to reproduce error states (e.g., during the system failure). This synchronisation of databases, in turn, had implications for privacy and accountability. These were resolved by designing an additional tool which not only anonymised (de-identified) data, but also only synchronised limited data from a specific time interval (i.e., the data required to analyse the error). There were additional measures taken to avoid human errors in updating the database, for example, restricting access and use of specific SQL commands that could result in data record updates.

In case III, the responsive design was chosen because of its advantages in compliance with accountability requirements, in addition to other usefulness; not only did it not require a user to install apps (and store data) on mobile devices, but it also allowed the use of existing accountability routines and functionalities.

The overall U-CARE software system was built with traceability of actions insofar as possible. The system architecture was designed to explicitly log all activities (who did what and when). Also, there were two separate databases, one for research data and one for research participants' personal information. The access to research participants' personal information database was highly restricted, and only the U-CARE software system accessed the data. Following ethical considerations, any access to research participants' personal information through the U-CARE software system was considered a privacy breach and logged. The clinical researchers were only shown the research participants' nicknames (i.e., a system-generated unique user name or user name chosen by the research participant), to protect the research participants' privacy. Only if there was a need (e.g., when suicide was feared, when required to access medical records) were psychologist (with a therapist role in the system) allowed to see a research participant's full name and phone number; this was logged as a privacy breach. The external communication messages (e.g., SMS and email) were carefully designed so they did not include any personal

information that could identify the research participants, for example, personal identity number or phone number. Similarly, health care professionals in the hospital (with a registrar role in the system) could enter data regarding research participants using personal identity numbers, but were not able to connect it to the nicknames used in the U-CARE software system. The information that was collected through the system (e.g., chat conversations, private messages, diary entries, et cetera) was processed based on the research participants' signed informed consent.

In the U-CARE context, the stakeholders encountered various privacy issues related to legislative, cultural, conceptual, technological, organisational, and methodological concerns. The development team ensured, insofar as possible, that the U-CARE software system was accountable and conformed with the Swedish legislation in force. The accountability characteristic in the U-CARE software system was another reason for extending the U-CARE research programme to that of a service provider, allowing other research groups and researchers to design and run studies with confidence in that the U-CARE software system fulfilled all accountability requirements. Similarly, the accountability characteristic indirectly resulted in securing additional funding based on its credibility and (re-)use in multiple situations. The logging mechanism also led to rich data which resulted in new insights.

The development of eHealth behaviour interventions should comply with existing regulatory frameworks with consideration for emerging standards around ethics (Michie *et al.*, 2017). eHealth research software not only needs to satisfy our functional requirements but also needs to satisfy our societal, moral, and legal requirements. This is quite unusual and software developers have traditionally not been trained for it. Integrity and ethical conduct are required on the part of software developers to maintain the confidentiality of research participants' data. The accountability mechanisms are necessary to build trust in the system (Weitzner *et al.*, 2008). However, heavily regulated industries that demand accountability, transparency, and documentation may require additional development sprint(s) after a few iterations of software development, with the goal of enhancing rigour that may be lacking in regular sprints (Fitzgerald *et al.*, 2013).

In the DSR literature the accountability quality characteristic is discussed under ethicality (e.g., Prat *et al.*, 2015; Venable *et al.*, 2016). Similarly, Myers & Venable (2014) proposed a set of ethical principles for design science research, highlighting ethical and legal issues. Sjöström, von Essen, *et al.* (2014) suggested that accountability issues needed to be addressed in both the artefact and the design process. The U-CARE software system's accountability mechanisms not only built trust and maximised the possibility of accountability, but also prevented intentional misuse.

In conclusion, based on the retrospective analysis, accountability appears to be the most discussed and thus most pertinent characteristic in the U-CARE

software system, given the eHealth context. However, when it comes to sustaining usefulness, the most important characteristic was "embedded in the *design system*."

Openness

The degree to which an artefact is open to inspection, modification, and reuse has an impact on its sustenance (Gill & Hevner, 2013). The aspects of openness make the artefact more reusable and available for modification and analysis, especially when combined with decomposability and malleability. Openness makes it easier both to see how an artefact is constructed and to modify existing components. The contemporary discourse on openness, for example, open source, open data, open standards, open content, and open access, also referred to as *open science* (Nosek *et al.*, 2015; Aalst *et al.*, 2016; Munafò *et al.*, 2017), is playing a significant role in making academic research transparent and accessible. Openness requires new skills and competencies, for example, concerning intellectual property (IP) rights and other legislative issues (Sjöström, von Essen, *et al.*, 2014).

In the U-CARE context, the interventions were released under a Creative Commons licence, allowing anyone to use them for non-commercial purposes. This open content policy allowed the researchers, both within U-CARE and associated with U-CARE, to reuse, revise, remix, retain and redistribute the interventions for current and future (re-)use. In case III, this enabled psychologists to adapt contents without requiring permission from the original authors.

In order to promote openness and innovation, the U-CARE context strived toward open sourcing of the U-CARE software system (Sjöström, von Essen, *et al.*, 2014), but at the time of this dissertation, the licensing of the system has not yet been determined. Although the U-CARE software system is not currently open source, it was designed using multiple open source components that indirectly have had an impact on the evolution of the artefact design, as the system evolved whenever the open design of components evolved (e.g., jQuery, jQuery plugins, and NuGet packages). In cases II and III, the development team considered open source technologies in order to promote innovation and openness.

Recently, U-CARE has decided to make data collected through the U-CARE software system accessible[91], a move toward embracing open data. Also, "sharing of data from the clinical trials benefit patients by enabling new discoveries, meta-analyses, and confirmation of published results" (Lo & DeMets, 2016). In case I, with the two-stage periodic data export architecture, the development team's objective was to empower the clinical researchers to flexibly export data from the U-CARE software system with the intent of making data accessible. This feature will further foster data access, sharing, reuse,

[91] https://www.u-care.uu.se/collaboration/u-care-accessibility/ [accessed: November 15, 2018].

and reproducibility, which is extremely important in an academic research context (Murray-Rust, 2008; Peters *et al.*, 2012; Dallmeier-Tiessen *et al.*, 2014; Hettrick, 2016).

As a result of the retrospective analysis, it appears that openness remains under discussion in U-CARE in regards its impact on sustaining the usefulness of the U-CARE software system, but it is recognised for its value in the evolution of the U-CARE software system.

The appropriate selection of quality characteristics depends on the context (Hevner *et al.*, 2013). Paying attention to these quality characteristics during the development of an eHealth research software will provide a better future for the resulting artefact in terms of sustaining its usefulness.

In the following, the design principles are abstracted with regards to a broader class of solutions, taking into consideration the entire U-CARE context, artefact and design process.

Design Principles

Design principles are recommendations on how to address a specific class of problems or class of solutions in a range of settings (Markus *et al.*, 2002; Sein *et al.*, 2011; Mckenney & Reeves, 2012). Practitioners need concrete/detailed instructions on how to build an artefact, whereas researchers strive towards general knowledge about creating other instances of artefacts that belong to the same class (Chandra *et al.*, 2015; Chandra Kruse *et al.*, 2016). A more specific formulation of design principles (through concretisation) provides more practical instructions, but also narrows the class of problems and class of solutions that can be addressed. On the other hand, a more generalised formulation of the design principles (through abstraction) provides more general instructions, while broadening the class of problems and class of solutions that can be addressed. The objective at this stage is to have a broad enough level of abstraction to enable researchers and practitioners in other contexts to instantiate the design principles while designing their eHealth research software. So, considering lessons learned through the three ADR cases to generate more abstract design knowledge, the design principles formulated are presented in one place for analysis (see Table 32).

Table 32. Design principles from three ADR cases

Design principle	**Specification**
Case I: Data export feature – design principles for data export in eHealth research software	
The principle of simplicity	Provide easy-to-use data export functionality in order for [clinical] researchers to export data, preferably by a single click via a simple UI, given that such functionality should not require

	in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	Data export functionality should be divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	a) Customise: Data export functionality should be customisable in order for [clinical] researchers to tailor [their own] research data and descriptive metadata export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats, such as CSV or XML, or tailored for spreadsheets or common statistical packages, in a way that is useful for downstream applications. b) Filter: Data export functionality should allow data filtering in order for [experienced clinical] researchers to customise data export according to their preferences and needs, given that such functionality should guide the researcher to filter exportable data and allow the researcher to save and reuse their data exports as templates. c) Schedule: Data export functionality should allow scheduling data export requests in order to get data after specified intervals [based on study design] or when data is available [in cases where the volume of data would increase data export processing time].
The principle of accountability	a) Privacy: Data export functionality should anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: Data export functionality that enables the clinical researcher (i.e., study owner or principal investigator) to restrict data access in order to enforce governance policies, data extraction, and ethical guidelines, given that such data access restrictions can be researcher-specific (based on access privileges), time-specific (i.e., at multiple intervals with the same/refreshed/additional datasets, or one-off after the study completion or termination) and data-specific (i.e., partial, full, or selected dataset). c) Auditability: Data export functionality should log all activities related to data export in order for study owner to fulfil audit and regulatory requirements, given that such logs store all data export events [when (timestamp), who (user identity – role), how (encrypted/plain text), why (purpose specification and use) and what (data specification)] to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics.
Case II: The technology adaptation process – design principles for sustaining the usefulness of eHealth research software	
The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging technological landscape, in order to promote fitness to the changing design landscape, given that the development process is supported by adequate test coverage, automated and

	continuous/frequent test-deliver-feedback development practices, a set of appropriate tools, and continuous upskilling of the development team.
The principle of embracing proactive practices	The software developers of eHealth research software should embrace proactive practices in order to improve code readability, extensibility, testability, simplicity, and potential velocity increase, given that resources (time, money, and attention) are allocated for such practices.
Case III: Extending the artefact – design principles for sustaining the usefulness of eHealth research software	
The principle of engagement with stakeholders	The software developers of eHealth research software should continuously engage with stakeholders, in order to adapt the software in a direction which will satisfy stakeholders, given that the stakeholders are willing and committed to such engagement in the long term.
The principle of co-design with stakeholders	The software developers of eHealth research software should co-design with stakeholders, in order to obtain continuous and early feedback, knowledge and requirements elicitation, gain trust, increase relevance and usefulness, and enhance creativity, engagement, and collaboration, given that there are incentives or motivation for stakeholders to participate in co-design and an availability of resources to make co-design possible.

The similarities within the design principles made it evident that the design principles concerned the functionalities and features of the design product and activities in the design process. In other words, design principles governed the development or selection of system features and design principles guided the development process (Walls *et al.*, 1992). This was consistent with the views of Gregor & Jones (2007) and Sjöström & Ågerfalk (2009), that design theories are about products and processes (or methods).

The design principles that pertain to the design product were simplified by eliminating details related to the specific feature (i.e., data export functionality) and reformulated to make them applicable to a broader class of solutions (i.e., eHealth research software). Considering the full system view, knowledge abstraction was based on hands-on experience of designing, participatory observations, and retrospective analysis of features in the U-CARE software system. Appendix F.1 describes and presents the abstraction process in detail.

The design principles related to the design process emerged and were formulated for the entire U-CARE software system and thus are already generalised. In other words, these design principles are proposed to be useful as-is for a broader class of solutions (i.e., eHealth research software).

Table 33 presents the reformulated product-related design principles and as-is process-related design principles.

Table 33. Design principles for sustaining the usefulness of eHealth research software (final version)

Design principle	Specification
For design product	
P1: The principle of simplicity	Provide the eHealth research software with easy-to-use functionalities in order for researchers to use it in their [eHealth] research, preferably via a simple UI, given that such functionalities should not require in-depth technical knowledge and should not overwhelm the researchers with details.
P2: The principle of modularity	Provide the eHealth research software's functionalities in modules, to enable for maintenance and reuse by software developers, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
P3: The principle of malleability	Provide the eHealth research software with customisable functionalities in order for [experienced] researchers to tailor them according to their [potential] needs, preferences, or usage context, given that such functionalities guide the researcher during the customisation.
P4: The principle of accountability	a) Privacy: Provide the eHealth research software with functionality that anonymises data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: Provide the eHealth research software with functionality that enables the researcher (i.e., study owner or principal investigator) to restrict system and feature access in order to enforce governance policies and ethical guidelines, given that such access restrictions can be researcher-specific (based on access privileges), and data-specific (i.e., partial, full, or selected datasets). c) Auditability: Provide the eHealth research software with functionality to log activities related to research in order for study owner to fulfil audit and regulatory requirements, given that such logs store [accountability-related] events [when (timestamp), who (user identity – role) and what (specification)] to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics.
For design process	
P5: The principle of engagement with stakeholders	The software developers of eHealth research software should continuously engage with stakeholders, in order to adapt the software in a direction which will satisfy stakeholders, given that the stakeholders are willing and committed to such engagement in the long term.
P6: The principle of co-design with stakeholders	The software developers of eHealth research software should co-design with stakeholders, in order to obtain continuous and early feedback, knowledge and requirements elicitation, gain trust, increase relevance and usefulness, and enhance creativity, engagement, and collaboration, given that there are incentives or motivation for stakeholders to participate in co-design and an availability of resources to make co-design possible.
P7: The principle of technological-ecological adaptation	The eHealth research software should continuously be adapted by software developers, regarding both its compliance with new requirements from its stakeholders and its fitness to the emerging

	technological landscape, in order to promote fitness to the changing design landscape, given that the development process is supported by adequate test coverage, automated and continuous/frequent test-deliver-feedback development practices, a set of appropriate tools, and continuous upskilling of the development team.
P8: The principle of embracing proactive practices	The software developers of eHealth research software should embrace proactive practices in order to improve code readability, extensibility, testability, simplicity, and potential velocity increase, given that resources (time, money, and attention) are allocated for such practices.

From the multiple ADR cases, it was observed that the quality characteristics have an impact on or relationship with design principles and vice versa. In the following sections, the identified quality characteristics' relationships to the above design principles, i.e., the typology of sustaining usefulness, are investigated through retrospective analysis.

Typology of Sustaining Usefulness

The significant role of typologies has been recognised in Information Systems literature (e.g., Williams *et al.*, 2008; Walsh, 2015). Typologies allow researchers to postulate on the relationships between concepts (Nickerson *et al.*, 2013). During the retrospective analysis, following an empirical-to-conceptual (i.e., bottom-up) approach, the typology of sustaining usefulness was constructed, considering each design principle and grouping it with similar ones based on its relationships with each particular quality characteristic. To establish the relationships between the design principles and quality characteristics, the typology of sustaining usefulness, including the design principles (P1–P8) with their qualifications (i.e., design product or design process), are presented in Table 34. The identification of quality characteristics and the formulation of design principles are empirically grounded in the multiple BIE cycles and different ADR cases, resulting in a typology.

Table 34. The typology of sustaining usefulness

Quality characteristic	**Concerns product/process**	**Design principles**
Decomposability	Design product	The principle of modularity (P2)
Malleability	Design product	The principle of malleability (P3)
Embedded in design system	Design process	The principle of engagement with stakeholders (P5)
	Design process	The principle of co-design with stakeholders (P6)
	Design process	The principle of technological-ecological adaptation (P7)
	Design process	The principle of embracing proactive practices (P8)
Simplicity	Design product	The principle of simplicity (P1)
Accountability	Design product	The principle of accountability (P4)
Openness	Design product	*Tentative: The principle of embracing openness (P9)*

It is important to note that *the principle of technological-ecological adaptation* (P7) was initially perceived as a '*technological-ecological fit*' quality characteristic of the design product and was defined by Mustafa *et al.* as:

> Technological-ecological fit highlights the IT artefact's relations to emerging boundary objects (e.g., plug-ins and APIs) and the way that those relations constrain and/or enable the mutability of the IT artefact. (2014, p. 302)

During the formalisation of learning in case II (BIE cycle II), P7 was perceived as a quality characteristic that concerned the design process; later, it became a design principle (see P7 in Table 33). Gill and Hevner (2013) presented a preliminary list of characteristics related to the design artefact only. However, it was observed in the U-CARE context that the design process was equally important. The formalisation of design principle P7 initiated deliberation concerning other quality characteristics in the preliminary list. For example, '*embedded in design system*' was perceived as a quality characteristic that concerned the design process. Similarly, *accountability* was perceived as a quality characteristic that concerned both the design product and the design process. However, in the typology, the *accountability* quality characteristic and P4 were related to the design product only. The design process related aspects were handled in the U-CARE healthcare organisation which was not part of the U-CARE software system.

Typology (Re-)Construction in the U-CARE Context

Gill and Hevner (2013) stated that designers (in practice) base their designs on more or less explicit utility functions. They proposed an exaptation from economics, where utility is posited as a function to rank choices in the context of decision-making. Gill & Hevner suggest that:

> the fitness of a design artefact must be estimated using a utility function that considers the full range of characteristics that can impact the likelihood that the artefact will further be reproduced and evolve. (2013, p. 247)

The typology consists a set of quality characteristics and a set of design principles. Additionally, as described in Chapter 4, a design principle includes material property, the activity of the user (or group of users), and boundary conditions. In the U-CARE context, quality characteristics helped designers in better estimating artefact fitness through informing their utility function. It was observed that the design team was acting in line with the design principles (P1–P8). As a result, quality characteristics and design principles ensured that the functionality of the U-CARE software system remained available – improved and supported – and will be used (survives), reused, or extended with reasonable efforts (reproduced and evolved) in the future.

In this section, I illustrate (re-)construction of a typology for sustaining the usefulness of eHealth research software in the U-CARE context. Although the

typology was conceptualised in the retrospective analysis stage of this dissertation, over time, the enactment of quality characteristics and instantiation of design principles resulted in construction and refinement of a typology. For example, design principles P1–P4, related to quality characteristics of the design product, were instantiated in the U-CARE software system (see case I). Over time, the design process received more attention and design principles P7 and P8, related to quality characteristics of the design process, were instantiated in the design process of the U-CARE software system (see case II). At the intermediate maturity stage of the U-CARE software system, the typology could be perceived as a premature version. At a later matured stage of the U-CARE software system, design principles P7 and P8 were instantiated in case III. Case III resulted in additional design principles P5 and P6, which led to refinement of the typology.

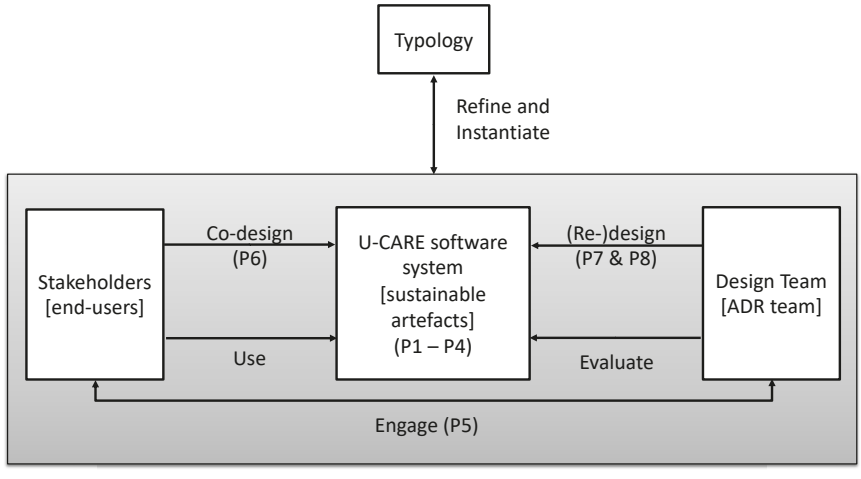


Figure 46. Typology (re-)construction in the U-CARE context.

Figure 46 shows the (re-)construction of typology in the U-CARE context. The figure represents the continuous improvement (refinement) of the typology (i.e., typology evolves as the context changes). During multiple ADR cases, quality characteristics were enacted and design principles were instantiated for sustaining the usefulness of the eHealth research software in the academic research context.

The proposed typology (in its current version, as presented above) was considered for further refinement and evaluation in subsequent BIE cycle(s). Over time, however, the design researcher may dynamically (re-)construct the typology, by adding or refining design principles and quality characteristics, as they interact with various stakeholders in the U-CARE context and through the design and use of the U-CARE software system. Refinement and appropriation of the typology result in sustaining the usefulness of the system (i.e., creating a sustainable artefact). The typology helps the design team (i.e., the

ADR team) in making design decisions, accumulating and sharing, and visualising design knowledge.

The typology is not yet evaluated for its utility in the U-CARE context. However, because I show a retrospective (re-)construction of the typology and present transparent and rich descriptions of three ADR cases, I argue that the use of a typology can facilitate a common understanding in the design team of the quality characteristics and design principles for sustaining usefulness, as well as providing a visual guide for reflecting on the design decisions made by the design team, and facilitating communication and collaboration among stakeholders.

The typology (re-)construction in Figure 46 can be generalised to suggest how a typology can be constructed, refined, and instantiated in another eHealth research software project. In this sense, the typology is prescriptive, as it tells a designer what ought to be done. The typology that consists of both a set of quality characteristic and a set of design principles can be generated by appropriation of action design research (in single or multiple cases). Also, quality characteristics and design principles might be related to the design product or the design process. The appropriation of typology is *estimation, inset and enactment of fitness characteristics and instantiation of design principles in the design process by a designer(s)*. Hence, I have derived an abstract (meta-)design principle for this dissertation:

> Design eHealth research software through the appropriation of a typology of sustaining usefulness, so that stakeholders can sustain software usefulness in the continuously changing design landscape in an academic research context.

Looking Back, Moving Forward – Re-visiting the Design Principles

With a point of departure in the typology (see Table 34), which encompasses the quality characteristics and design principles, it became apparent that there was no design principle linked to openness quality characteristic. The impact of openness on sustaining the usefulness of the U-CARE software system is evident in the U-CARE context, as previously discussed. However, the decision to open source the artefact is still under discussion. The open source model was discussed early on:

> Open source is a business model among others, and we need to think about our stakeholders when we decide whether the system is going to be open source or not […] we will use Microsoft servers to store the system, but otherwise, we have not used any other licensed software products. Consequently, at the moment, it is up to us whether we will make the [U-CARE software system] open source or not. (ISR-1, 2010, IT meeting minutes)

Similarly, during a number of SAB meetings, the pros and cons of open source code were discussed, such as:

> The decision to use or not to use open source may have implications on sustainability. [The panel] was not agreed on which decision would be best for sustainability. (Panel discussion on sustainability, 2013, SAB report)

> [From a] technological point of view, it was suggested that using open source could be one way to be innovative and to unleash the power of open innovations. (Panel discussion on innovation, 2013, SAB report)

> Another thing that we discussed a lot but [where] we never really [became] active […] is the open source issue. When we started this, we were very eager to get a working [software system] right away to start our studies. If I have done this again I would spend more time reflecting about existing open source products which you can build upon.
>
> […] if you look at this adaptation to mobile devices, if we had used an open source platform like Joomla it would have worked on mobile devices because that open source platform has such a big group of developers, so there are always people to make sure that things built for Joomla also work on mobile. (ISR-1, 2015, SAB meeting)

The primary legal concerns tend to be around licensing and code ownership. Open sourcing of the artefact from its inception would had helped the U-CARE stakeholders create an IP ownership structure early on (e.g., appropriate licensing), with the collaboration of stakeholders and legal experts, to avoid otherwise complex legal issues, such as IP rights. The situation regarding open sourcing of the U-CARE software system was described well by Jiménez *et al.* (2017, p. 5) in citing Fogel (2005): "the longer a project is run in a closed manner, the harder it is to open it later" and in the recommendation to "make source code publicly accessible from day one," while encouraging the adoption of best practices in software development to promote research software quality. Jiménez *et al.* suggest that embracing openness from day one has the following benefits:

(1) Promotes trust in the software and broader project
(2) Facilitates the discovery of existing software development projects
(3) Provides a historical public record of contributions from the start of the project and helps to track recognition
(4) Encourages contributions from the community
(5) Increases opportunities for collaboration and reuse
(6) Exposes work for community evaluation, suggestions and validation
(7) Increases transparency through community scrutiny
(8) Encourages developers to think about and showcase good coding practices
(9) Facilitates reproducibility of scientific results generated by all prior versions of the software

(10) Encourages developers to provide documentation, including a detailed user manual and clear in-code comments (2017, p. 5)

Doyle *et al.* (2019)[92] recently conducted a literature review, using an open science lens, to assess DSR literature. They found a few studies, such as Hariharan *et al.* (2017) and Coenen *et al.* (2018), which mention that the artefacts built were made available on open source platforms [Bitbucket[93] and GitHub[94], respectively]. Doyle *et al.* (2019, p. 1) suggested that the DSR community should strongly engage with open science as it "is facing significant challenges related to limited accessibility of knowledge and artefacts produced." They proposed 'open artefacts' for DSR researchers as a way to adopt open science practices and defined them, adapting Pontika *et al.* (2015) and Gill & Hevner (2013), as "DSR artefacts that can be accessed online for free, with an open license that allows use, inspection, modification, and reuse." Similarly, I also defined sustainable research software and argue about sustaining the usefulness of the artefact (i.e., eHealth research software).

Considering the above discussion, I have formulated a tentative design principle:

> The principle of embracing openness (P9):
> The stakeholders of eHealth research software should embrace openness (i.e., open science practices) from the start, in order to make the software, data, and research more (re-)usable and available for modification and analysis (inspection and use), given that IP rights are discussed, managed and communicated effectively to all stakeholders at an early stage.

Much eHealth research software is developed in the academic research context(s). Such software (i.e., eHealth interventions) is no longer supported when research funding expires (Glasgow *et al.*, 2014), as is the case with research data (Dallmeier-Tiessen *et al.*, 2014). Embracing openness can lower the need for start-up costs for future eHealth intervention. For example, Research Electronic Data Capture (REDCap) was initially developed and deployed in academic research context (at Vanderbilt University), "to provide scientific research teams intuitive and reusable tools for collecting, storing and disseminating project-specific clinical and translational research data" (Harris *et al.*, 2009, p. 377). Although REDCap is not open-source software, it is available at no charge for institutions that join the REDCap Consortium. I argue that availability source code of eHealth research software can allow research groups (like U-CARE) to reuse or extend rather than developing a complex system from scratch. Similarly, open sourcing U-CARE could allow it to be used (survives), reused, extended (reproduced and evolved) in the future.

---

[92] Paper accepted for the DESRIST 2019 conference. Preprint available at https://osf.io/ye6xp [accessed: April 03, 2019].
[93] https://bitbucket.org/kit-iism/experimenttool/src [accessed: April 03, 2019].
[94] https://github.com/d-pac [accessed: April 03, 2019].

8.2 Reflecting on ADR

DSR has emerged as a viable approach to information systems research. Iivari (2015) reflects on two strands of DSR: The laboratory approach and the practice approach. ADR, as originally articulated by Sein *et al.* (2011), is perhaps the most well-known practice approach to design research. ADR promotes research relevance as the ADR researcher works in concert with key stakeholders in an organisational context. Purao *et al.*, (2013) suggested that, in ADR, the "domain of intervention should be the ensemble artefact, i.e., not only the hardware-software instantiation but also the work practices of organisational participants relevant to the context in which the IT-artefact is realised." They also suggested that "the design process should not be limited to the researchers' own conscious decisions but should also be open to influences from the organisational practices and participants." The credibility of research is established with prolonged engagement in the field, thick and rich descriptions, and close collaboration with participants throughout the research process (Creswell & Miller, 2000).

Gill & Hevner (2013), in their fitness-utility model, suggested that the evolutionary fitness of a design artefact is more valuable than its immediate usefulness. Ågerfalk & Wiberg in their panel report[95] mentioned the following, in reference to the fitness-utility model Hevner suggested:

> As for how to move DSR thinking forward, [Alan Hevner] then turned to the argument that Gill and Hevner (2011) offer and suggested that "we need to move beyond just thinking about usefulness as the nature of utility of the artifact" and ask ourselves "How can I make that artifact sustainable? How can it adapt to change in an environment?". (2018, p. 70)

Ågerfalk & Wiberg also stated that:

> this [moving beyond immediate usefulness] ambition echoes the Scandinavian participatory design tradition's emphasis on "change and development, not only technological change and systems development, but change and development of people, organizations, and practices, occurring in changing socio-historical contexts (Gregory *et al.*, 2003, p. 63)". (2018, p. 70)

It is evident that the ADR team, in the U-CARE context, followed the Scandinavian participatory approach to design as Information Systems researchers worked in close co-operation with the software developers, the clinical researchers, and other stakeholders. In this participatory approach, the goal is not just to design the artefact, but also to improve practice (Ågerfalk &

[95] Panel held at the third Scandinavian Conference on Information Systems in Sigtuna, Sweden, in August 2012. It is important to note at that time only an early version of the fitness-utility model was published in DESRIST 2011.

Wiberg, 2018), which becomes evident over time as ADR team not only design artefacts, but also reflect on and intervene in the design process. Based on my experience in the U-CARE context, I agree with Sein & Rossi (2019) that ADR is bringing us towards a more participatory approach to design. Haj-Bolouri *et al.* (2016) also advocate a participatory design to involve stakeholders and engage them early on in ADR cycles, and throughout the process.

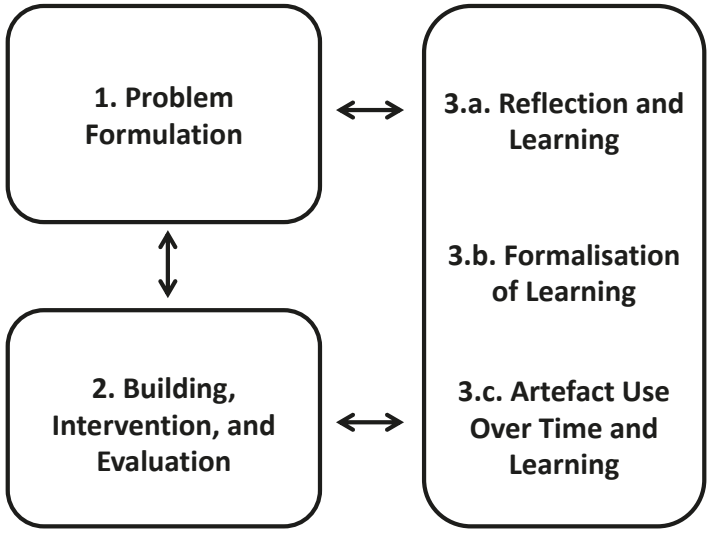


Figure 47. Longitudinal action design research.

Figure 47 illustrates how ADR was appropriated in this dissertation. As mentioned in Chapter 4, *reflection and learning* (Stage 3 in 'proper' ADR) and *formalisation of learning* (Stage 4 in 'proper' ADR) occurred continuously during the ADR BIE cycles. Hence, both stages were merged into Stage 3, as Stage 3.a and Stage 3.b (see Figure 47). "Problem re-interpretation in light of solution construction is the hallmark of wicked problems – and the kinds of problems we deal with during research do require such an approach" (Sandeep Purao, personal communication, October 17, 2014). Westin (2014, p. 97) has also proposed the modification to ADR by adding a feedback path from the *formalisation of learning* (Stage 4 in 'proper' ADR) to the *problem formulation* (Stage 1 in 'proper' ADR) to capture unanticipated consequences that are vital for new iterations of BIE. As the stages were merged, the proposed feedback path was established and it enabled reformulating of the problem or even identifying a class of problems in the broader context.

The longitudinal engagement allowed me to reflect on the artefact evolution over time after its implementation in the U-CARE context. Such reflections occurred between and after the BIE cycles and are presented in the respective sections under the heading *artefact use over time and learning*.

Mullarkey & Hevner (2018) have proposed evolution as a BIE cycle type, arguing that evolution of the artefact over time will continue to generate knowledge useful to the researcher and practitioner. Although I agree with them regarding the possibilities of evolution of the artefact and knowledge generation, I have, in my appropriation of ADR, presented *artefact use over time and learning* as a supplement to *reflection and learning* in Stage 3.c (see Figure 47). This was because I agreed with Sein & Rossi (2019) that it is not directly part of a full-fledged BIE cycle as it does not contribute to design of an artefact. In case I, Stage 3.c (*artefact use over time and learning*) led to both a BIE cycle (i.e., iv) and *formalisation of learning*. Stage 3.c enabled the ADR researchers, even after a long period had passed, to capture unanticipated consequences that might lead to *problem (re-)formulation,* another BIE iteration, or even a new ADR case. Hence, considering this 'Longitudinal Action Design Research' (LADR) appropriation is suggested for cases where an ADR researcher expects a prolonged engagement in the empirical context.

Through three ADR cases and multiple BIE cycles, the artefact in this study emerged through interaction between design and use. The research should be guided and evaluated based on explicit quality criteria (Sarker *et al.*, 2013). Table 35 highlights how research in this dissertation adheres to ADR DPs (Sein *et al.*, 2011).

Table 35. Appropriation of ADR principles

ADR principles (ADR stage)	**The actualisation of ADR principles**
DP1. Practice-inspired research (Problem Formulation)	Research was started due to the need for designing and sustaining an eHealth research software system in U-CARE context.
DP2. A theory-ingrained artefact (Problem Formulation)	The design and development of the artefact was informed by scientific theories, as described in the section on reflection and learning.
DP3. Reciprocal shaping (BIE)	The IT artefact was designed in the real organisation settings. Multiple versions of the artefact were deployed in the organisation and used over a long period of time.
DP4. Mutually influential roles (BIE)	The ADR team consisted of researchers, practitioners and representatives of various stakeholder groups. The lead designer was an Information Systems researcher and supervisor of a PhD student who was a co-designer.
DP5. Authentic & concurrent evaluation (BIE)	The artefact was continuously evaluated as part of intervention in the empirical context.
DP6. Guided emergence (Reflection and Learning)	The ADR team guided the emergence of artefact by utilising concurrent evaluation (DP5). The artefact reflects the intentional design (DP2) as well as evolutionary shaping by organisational use (DP3 & DP4).
DP7. Generalised outcomes (Formalisation of Learning)	The research resulted in a generalised problem and solution, as well as design principles for sustaining the usefulness of eHealth research software, as described in the contributions section.

Co-creating knowledge (i.e., formalisation of learning in the ADR context) was a challenge. There were some academic courses organised in the U-CARE context to introduce U-CARE stakeholders, particularly PhD students, to various disciplines and their research philosophies, for example, psychology, caring sciences, economics, and Information Systems. These courses enabled us to build a basic understanding and a common language. However, the goal of jointly publishing interdisciplinary research was not achieved due to different disciplinary requirements, particularly for PhD dissertations. For example, there was a lack of author contribution mechanisms for interdisciplinary publications. Although there were many publications, mainly by senior researchers, these were in the eHealth domain with Information Systems researchers contributing, or cases where clinical researchers contributed in Information System researchers' publications. Similarly, motivating the U-CARE stakeholders to participate in co-creating and formalising the design knowledge was a challenge due to epistemological and ontological differences in how research is conducted in different disciplines. However, co-design was achieved mainly thanks to the U-CARE software system's central role in the U-CARE context. The multi-disciplinary composition of the research team meant that different individuals within the research team were busy at different times. Time prioritisation and accessibility to health care stakeholders were major difficulties in involving the stakeholders in ADR. Despite this, the multi-disciplinary ADR team contributed to my research by participating in the design workshops, providing feedback during evaluations, and allowing me to engage in and observe their research.

Reflecting and documenting continuously was a challenge, mainly as I was also participating in design workshops; for example, it was difficult to write observations during the IT meetings while also being actively involved, and vice versa. Another challenge was the lack of tools for keeping track of research progress in regards to DSR in general, and ADR in particular, such as tools for tracking or annotating the design process. Recently, the DSR community has realised this problem and designed artefacts for this purpose, for example, MyDesignProcess.com (Brocke *et al.*, 2017). It could have been helpful if such tools had been available when I began the research project. There is an emerging interest in the design of tools for use in design science research (Morana *et al.*, 2018).

Finally, the ADR researcher is, unlike in other research paradigms, not necessarily in control of the project's speed of progress, due to real-world problem situations encountered in a specific organisational setting (Drechsler & Hevner, 2016). An example would be variations in the data export feature and their use over time. The logged data provide additional opportunities that could not have been anticipated by the clinical researchers in advance.

8.3 ADR across Multiple Cases

As it was previously mentioned (in Section 4.2), the ADR method has been elaborated by scholars, such as Haj-Bolouri *et al.* (2016) and Mullarkey & Hevner, (2015; 2018). However, neither 'proper' ADR nor elaborated ADR approaches focus on how to conduct ADR across multiple cases – the focus is on design, implementation, and learning in a single organisational context. With a few exceptions (e.g., Hovorka & Pries-Heje, 2013), DSR methods do not explicitly emphasise knowledge abstraction from numerous design experiences in a more extensive design and development context. Most of the design literature describes knowledge abstraction from a single artefact instantiation at a time. Hovorka & Pries-Heje (2013) advocate that researchers should avoid 'ignoring the iceberg,' by conducting multiple studies to gain more in-depth knowledge about the phenomenon at hand, e.g., gaining better understanding of assumptions behind the design principles identified. Similarly, many design research practices face a situation where research aims are achieved through learning based on multiple design cases (Sjöström *et al.*, 2012; Nunamaker *et al.*, 2017). Thus, design researchers encounter variations in the design process, e.g., concerning organisational context, design goals, design pace, and design duration. Such multiple-case theorising situations are likely to occur in sizeable multi-disciplinary research environments addressing societal challenges. Nunamaker *et al.* (2017) argue that such environments hold increased importance for high-impact Information Systems research. As a consequce, our research approaches should factor in theory development taking place over more extended periods and draw from multiple design efforts. As argued by Baskerville & Pries-Heje (2016), the process of abstracting knowledge from design science research is non-trivial. Thus, there is a need for further discourse on how to develop abstract knowledge in design-oriented research approaches.

As is evident in Section 8.1 (and this Section 8.2), retrospective analysis and synthesis of learning across multiple ADR cases resulted in additional (new) design knowledge through reflection and abstraction. The cases relate in at least two ways. First, they were part of the same design context (outlined in Chapter 0), although in different stages of maturity. Second, they all concern designing key functionalities in eHealth research software and are likely to contribute to an overarching knowledge interest. In other words: all cases were relevant to a class of problems. Experiences from conducting ADR in multiple cases over a more extended period and retrospective analysis served to articulate 'Augmented Action Design Research' (AADR), an extension of ADR. This approach guides how to conduct multiple ADR projects that build towards an overarching knowledge aim. The following section elaborates how the AADR method, with an additional stage of augmented reflection and learning (ARL), facilitated multiple case design and interpretation.

Augmented Action Design Research

The traditional ('proper') ADR approach is based on interpretations and formalisations of experiences from BIE cycles. AADR adds another level of analysis, following the ideas of Lee & Baskerville (2003), where generalisations from empirical data to theory occur in several cases, thus providing insights to support an emerging overall abstraction process.

A crucial question is when and why augmented action design research is initiated. Two possible scenarios can be seen, both related to the articulation of an overarching knowledge aim. More specifically: (1) a research aim that cannot be sufficiently addressed through a single ADR project. In some cases, that aim is known from the start, e.g., in the context of establishing research environments around a complex topic (Nunamaker *et al.*, 2017). The overarching knowledge aim may then be the very reason to establish a new research environment. However, (2) a research interest may also be identified over time through reflections on design experiences (Sjöström, 2010). In the context of wicked problems, an essential aspect of the design process is that the problem domain is better understood over time, through the design activities taking place (e.g., Gregor & Hevner, 2013). The sooner researchers can articulate an overarching knowledge aim, the sooner they can appropriately adapt their data collection strategies.

In AADR, research may address multiple problem classes. As outlined above, the overarching knowledge aim may be articulated at a late stage. In such cases, several ADR projects may have been conducted, addressing problem classes that are only peripherally related to the overarching knowledge aim. Still, those projects may be beneficial to include in the AADR project due to documented design experiences that relate to the overarching knowledge aim. For instance, an individual ADR project may focus on the problem class of designing support for therapists to provide psychosocial counselling over the internet. The AADR knowledge aim, however, may concern the problem class of incentivising users (or stakeholders) to do beta-testing or engage in co-design activities. Despite an initial unawareness of the beta-testing issue, the first ADR project may have resulted in rich data that supports the AADR endeavour. The AADR model that I present here was conceptualised independently of when the overarching knowledge interest was identified. However, a late identification of the overarching knowledge aim may create a need for additional data collection from the individual ADR projects, to facilitate retrospective analyses.

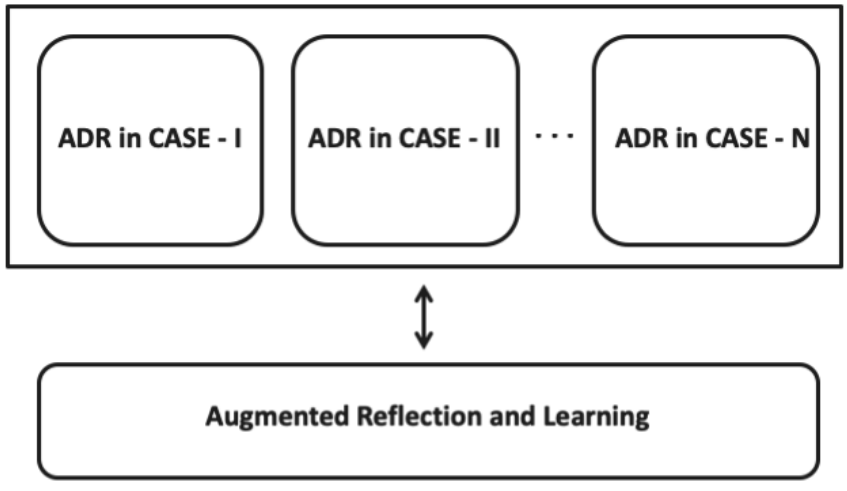


Figure 48. Augmented action design research.

Given the perspective outlined above, the AADR approach extends traditional ADR with a new stage (Figure 48): Augmented reflection and learning (ARL). Below, I further elaborate on the ARL extension.

Augmented Reflection and Learning

This research stage enables a holistic view of the problem domain and aims at a generalisation of results (e.g., design principles) drawing from the multiple ADR cases. It enables reformulating the problem and identifying the class of problems in a broader context or additional knowledge abstraction possibilities. Thus, the first step in ARL is problem re-interpretation as a more general class of problems. ADR suggests that reflection and learning occur continuously in ADR research, and conceptualises formalisation of learning as a discrete step. The AADR approach includes the formalisation of learning at the end of each BIE cycle (as discussed in the previous Section 8.2, Reflecting on ADR). The ARL stage is introduced to synthesise learning across multiple ADR cases, possibly resulting in a higher abstraction level. In other words, ARL combines both 'reflection and learning' and 'formalisation of learning' as one stage.

Haj-Bolouri *et al.* (2017) stated that "identifying the class of problems requires making a choice." Focusing on a single problem instance in every ADR case not only allows the involvement of stakeholders, but also remains the very essence of ADR: the single-entry point. ARL allowed me to cast the multiple problem instances as a broader class of solutions and abstract design knowledge. Here is a tentative list of tasks for the ARL stage, based on my experience:

1. Perform retrospective analysis and reflect on artefact evolution during the project.
2. Synthesise learning across cases.
3. Abstract the learning into a broader class of problems or class of solutions.
4. Abstract design knowledge.

So, the first step in ARL was retrospective analysis. This ARL stage was required to synthesise learning across multiple ADR cases to generate more abstract design knowledge. Figure 48 illustrates the individual case that followed the four stages of ADR (or LADR) as explained in Table 6, while ARL supplemented as a fifth stage (fourth stage, in the case of LADR).

Appropriation of ARL in this Dissertation

Unclear research interests coalesce into well-specified research questions over time through increased understanding of the problem domain. In this situation, there may be a need for retrospective analysis to reconstruct the design process so as understand the steps through which the design has evolved (Sjöström, 2017). As stated above, Gill & Hevner (2013) argue that to understand fitness, researchers need to look back in time in order to trace the evolution of an artefact. Hence, in the ARL stage, considering the successive evolution of the empirical context and research interests (i.e., sustaining the usefulness of eHealth research software), I looked back in time and analysed the past (2010–2019) to identify potential quality characteristics. The level of abstraction used was broad enough to enable researchers and practitioners in other contexts to materialise the design principles in use while designing their research software. The entire U-CARE context, the artefact and design processes (particularly the cases discussed in this dissertation) were considered, highlighting and formalising the reflection and learning of the U-CARE context that had the potential to benefit both research communities (e.g., design science) and practitioners (e.g., research software engineers) in the academic research context as a whole. The retrospective analysis required reconstructing the whole design process and the interventions that researchers had made. Here, the field notes with reflections (research log), IT meeting minutes, product backlog history, code repository, et cetera were quite helpful in reconstruction and establishing a chain of evidence.

Haj-Bolouri *et al.* (2017) indicated that ADR is compatible with retrospective framing. ARL in this sense is retrospective framing of the U-CARE software system and the design process. On the one hand, ARL is not part of ADR, as I was not designing anything (Sein & Rossi, 2019), but on the other hand, the ARL stage was focused around designing the artefact and design knowledge abstraction. Also, considering the full system view, knowledge abstractions were based on my own experiences of the development of features

in the artefact and participatory observations in other feature development by the development team. In a retrospective framing sense, I would argue that ARL was not just an interpretive step, but a supplement to the ADR stages. The ARL stage can also be considered as an ex-post summative evaluation. It is important to note that the retrospective analysis during the BIE cycles in the ADR cases was performed mainly for evaluation of the artefact.

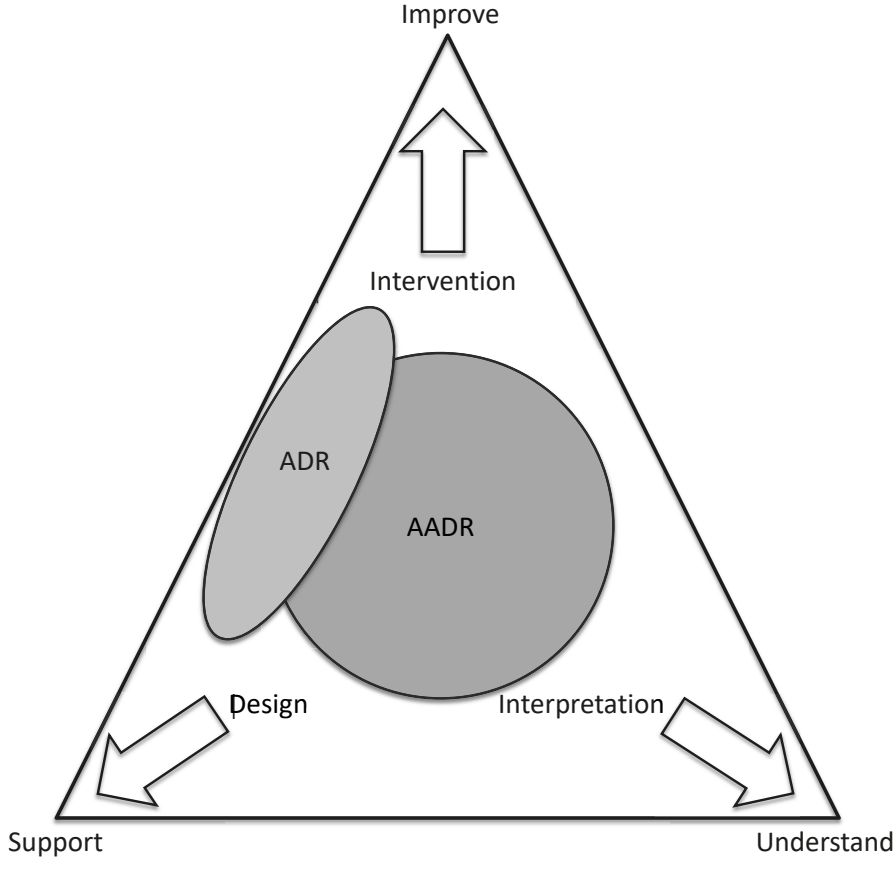


Figure 49. Positioning AADR (adapted from Westin, 2014).

Figure 49 illustrates how the proposed research method AADR fits with an ADR approach. The figure was originally adapted by Mathiassen (2002, p. 327), to illustrate different types of knowledge goals and activities, from Vidgen & Braa (1997, p. 527) and then adapted by Westin (2014, p. 31) for positioning ADR in the framework. Here, the positioning of AADR has been added. The arrows inside the triangle represent distinct research activities (i.e., interpretation, design, and intervention) through which knowledge is developed (Mathiassen, 2002). Mathiassen explained the framework as:

> First, to develop our understanding of systems development we engage in interpretations of practice; […] Second, to build new knowledge that can support practice, we design normative propositions or artefacts; […] Third, to learn what it takes actually to improve practice we engage in different forms of social and technical intervention. (2002, p. 327)

ADR strongly emphasises combining design and intervention to build an artefact and to formulate design principles through intervention in the organisational settings. A variety of research approaches may be used and combined within a larger project (Mathiassen, 2002), such as this dissertation. Chapters 5–7 illustrate the design and intervention that resulted in new knowledge which can support and improve practice (i.e., design principles). Furthermore, they illustrate the learning through engagement in the organisational context that resulted in increased knowledge about practice, and insights into barriers to and enablers of design process improvement efforts. Chapter 8 presents the retrospective analysis of collected data about practices and interprets these using the fitness-utility model. The reflection upon practices resulted in additional insights, for example, typology of sustaining usefulness and the AADR method.

In conclusion, it is worth considering the augmented action design research (AADR) method with an additional stage of augmented reflection and learning (ARL), to facilitate multiple case design and interpretation.

Part V: Conclusion

9 Concluding Discussion

In this final chapter, it is explained, in Section 9.1, how the research questions are addressed. The research contributions are discussed in Section 9.2. The implications of the research results are presented in Section 9.3. Lastly, in Section 9.4, future work is presented.

9.1 Re-visiting the Research Questions

The objective of this dissertation was to create an understanding of sustaining the usefulness of eHealth research software in an academic research context. The overall research aim was to explore *what Information Systems researchers and practitioners need to be aware of for sustaining the usefulness of eHealth research software in the academic research context.* The primary research aim has been divided into three sub-questions. This section is a summary of how I addressed the research questions, what I achieved, and how results are enacted.

> **RQ1:** What are the quality characteristics of eHealth research software that impact on sustaining its usefulness in the academic research context?

The first question was addressed by looking at the DSR literature to identify whether any list of such characteristics existed. The fitness-utility model of Gill & Hevner (2013) provided a preliminary list of such quality characteristics. However, keeping an open mind, I took part in the design of the artefact and observed the design processes in the empirical setting to identify context-specific characteristics. Decomposability, malleability, openness, embedded in design system, simplicity, and accountability were prevalent in the empirical context and perceived by the stakeholders as essential quality characteristics that impacted on sustaining the usefulness of the eHealth research system in an academic research context. Four quality characteristics, i.e., decomposability, malleability, openness and embedded in design system, were mentioned by Gill & Hevner (ibid.). Simplicity and accountability, inductively derived from the empirical material, have been proposed as revisions to the preliminary list of quality characteristics.

The second research question concerned the use of quality characteristics by Information Systems researchers and practitioners in sustaining the usefulness of eHealth research software in the academic research context.

> **RQ2:** How do Information Systems researchers and practitioners approach the quality characteristics for sustaining the usefulness of eHealth research software in the academic research context?

Chapters 5–7 illustrate how quality characteristics were enacted in three ADR cases while Chapter 8 presents how they developed over time considering the evolution of the artefact, design landscape, and context. The three ADR cases were carried out with real users (developers, researchers and focus group participants), using the real artefact, and real problems in an academic research context and in a real organisation's settings (as opposed to in isolated experiments).

> **RQ3:** What design principles should guide Information Systems researchers and practitioners in sustaining the usefulness of eHealth research software in the academic research context?

Design principles were inductively articulated during multiple BIE cycles in three ADR cases. Four of the design principles (regarding the design process) serve to guide researchers and practitioners in sustaining usefulness of eHealth research software, i.e., *the principle of engagement with stakeholders*, *the principle of co-design with stakeholders*, *the principle of technological-ecological adaptation*, and *the principle of embracing proactive practices*. In addition, five design principles (regarding the design product) serve to guide researchers and practitioners in designing sustainable eHealth research software, i.e., *the principle of simplicity*, *the principle of modularity*, *the principle of malleability*, *the principle of accountability*, and *the principle of embracing openness*.

The *class of problems* addressed by this dissertation is *coping with the evolving design landscape* in an academic research context. This is addressed by providing the researchers and practitioners with a set of design principles, a list of quality characteristics, a typology of sustaining usefulness, and an augmented ADR method. The dissertation provides an understanding of design theories for sustaining eHealth research software usefulness.

9.2 Research Contributions

Design theories give explicit prescriptions regarding *how to do something* and correspond almost exactly to Gregor's (2006, p. 620) *design and action* theory type and the *design theories* of Walls *et al.* (1992; 2004). According to Rossi

et al. (2013) "a significant portion of research on IS had little to do with what the actual information system functioned, and how it was developed." Some design research methodologies lack empirical grounding, since they are based on reconstructions of studies conducted for other purposes (Cronholm & Göbel, 2015; Cronholm & Göbel, 2016). Iivari (2015) has observed that most publications on ADR remain rather theoretical and calls for more actual ADR papers to evaluate the efficacy of the method. Empirical studies of design research have started to appear, but the descriptions of the development and evaluation of design artefacts and iterative emergence are often kept short. This dissertation fills this gap, answering the call of Gill & Hevner (2013) and Iivari (2015): enacting and augmenting theory by doing action design research, focusing on both practice and design research in information systems design. In this pursuit, I draw on a longitudinal in-depth contextualised multiple ADR study in the academic research context. The study generated descriptive (e.g., rich descriptions of design processes) and prescriptive (e.g., design principles, typology) knowledge of the problem space through empirical insights. This dissertation, therefore, contributes to a more comprehensive description of the development of design artefacts, allowing other researchers and practitioners to take part in the reasoning regarding design artefacts, design process and research design (see Table 36).

Table 36. Research contributions

Contribution	**Description**
Design principles	*Prescriptive knowledge:* Design principles (regarding both design process and design product) for sustaining the usefulness of eHealth research software in an academic research context.
Quality characteristics	*Prescriptive (normative) knowledge:* Extended list of fitness characteristics for fitness-utility model.
Typology	*Prescriptive (explanatory) knowledge:* The quality characteristics and their relationships to the design principles.
AADR method	*Prescriptive knowledge:* The method guides on how to conduct multiple action design research projects. Also: Reflections on the use of ADR.
Instantiation	*Descriptive knowledge:* Data export feature (*improvement*), Technology adaptation process (*improvement*), U-CARE adaptation to mobile devices (*exaptation*).
U-CARE practice	*Descriptive knowledge:* Extensive and rich descriptions of the development of a sophisticated eHealth research software.

Design Principles, Quality Characteristics and Typology

The dissertation resulted in a set of design principles that may guide designers in sustaining the usefulness of eHealth research software in an academic research context. By following best practices and design guidelines in software development, sustainable research software can be created, and as a result, the reproducibility and reusability of research can be improved (Goble, 2014; Crusoe & Brown, 2016). Groen *et al.* (2015) also concluded that use of best practices and design guidelines improves the quality of research software and leads to an increase in the publication output. The usefulness of the design principles was evaluated as the design principles were actionable by the development team and instantiated into the artefact – and the artefact indeed afforded the action described by the design principle. In essence, the design principles guide the interaction between design and use when sustaining the usefulness of eHealth research software.

This dissertation also answers the calls to Information Systems researchers to provide empirical instantiations of the fitness-utility model and extend the candidate fitness characteristics list. To the best of my knowledge, this is the first empirical instantiation and actual use of the fitness-utility model since it was introduced by Gill & Hevner (2013).

As sustaining research software usefulness refers to the appropriation (estimation, inset, and enactment) of fitness characteristics in the design process by designer(s) to ensure that the functionality of the research software remains available – improved and supported – and used (*survives*), reused, or extended with reasonable efforts (*reproduced and evolved*) in the future, this enactment is used for empirical grounding of the (re-)construction of a typology for sustaining the usefulness of eHealth research software in the academic research context.

Augmented Action Design Research

This dissertation also contributed to the method space. One of the key contributions to the DSR is the introduction of augmented action design research (AADR), an extension of ADR. The approach guides on how to conduct multiple action design research projects that build towards an overarching knowledge aim.

Instantiation

The innovative artefact is also an empirical contribution with a rich descriptions of the empirical context and evolution of the artefact (Ågerfalk, 2014). The *class of solution* addressed by this dissertation is sustainable eHealth research software. The U-CARE software system was designed to address real-life organisational problems, concerning the design and use of the data export

feature, the process of technology adaptation due to continuous technological-ecological changes in the design landscape, and the adaptation of the U-CARE software system to mobile devices. Sustaining the usefulness of eHealth research software is an important factor in achieving sustained organisational or societal impact.

9.3 Implications

In this section, I present how results from the dissertation about sustaining the usefulness can be used to improve the design of eHealth research software in academic research contexts.

Implication for Practice

Quality characteristics, design principles, typology, and rich descriptions (design examples) of the empirical context have implications for researchers and practitioners in design practice. This research has shown the importance of paying attention to quality characteristics for sustaining the usefulness of eHealth research software in an academic research context and described how they impact during software development. The practitioners in various design disciplines can benefit from the proposed quality characteristics, identified in this dissertation which include decomposability, malleability, openness, embedded in design system, simplicity, and accountability, to sustain their design artefacts over time. The various design situations in the ADR cases of sustaining the usefulness of eHealth research software illustrate the consequences of these six quality characteristics in the U-CARE software system. These rich descriptions will make practitioners aware of the importance of identification of quality characteristics in their contexts. Similarly, practitioners who are interested in designing similar research software for use in an academic research context will benefit from the design principles.

The typology of sustaining usefulness, in its current version, suggests that one should pay attention to both the design product and the design process. Furthermore, continuous formulation and refinement of design principles and their instantiation into a concrete artefact and design process could enact quality characteristics. Given that the proposed typology, is based on a single academic research context, it should be emphasised that a different context might require other characteristics and design principles than those included here. Hence, design practitioners need to embrace routines in their design processes to continually address sustaining usefulness, for example, using typology as an agile wall to discuss the enactment and instantiation of quality characteristics and design principles into their eHealth research software. Including various stakeholders in discussions will enable the practitioners to communicate which resources are required for sustaining usefulness in the long

run. The typology will allow practitioners and stakeholders to specify quality requirements and associate them with quality characteristics and design principles. As a result, the typology will enable stakeholders to prioritise the quality-related requirements (i.e., user stories).

In practice, agile Scrum teams actively use and maintain physical/and or electronic agile walls (i.e., JIRA, Trello, et cetera) in discussing, tracking, and managing their projects. Agile walls also display and communicate the project activities and progress status (e.g., in backlog, in progress, done), and are used during the various meetings (e.g., the daily stand-up, sprint planning, retrospective, et cetera). Given this use of agile walls, keeping the typology as a wall[96] could support design team collaboration and awareness, as it could act as a team's external memory. A Typology wall alongside the traditional agile walls would help the design team(s) to analyse and link their design decisions to a larger overall context of sustaining usefulness. The typology can provide an avenue for co-design and co-creating design knowledge, clarity, and transparency in the design process for design team(s) and stakeholders.

Although the scope of this dissertation is limited to eHealth research software, the quality characteristics, design principles and typology are broadly applicable, thanks to rich descriptions of sustaining the usefulness of the research software over time. Considering the academic research context and research software, this dissertation might be of interest to both researchers and practitioners in other fields and could potentially serve as a guide for research projects that design software for research, particularly those with a small development team, and provide empirically grounded design principles, quality characteristics, and a typology for sustaining the usefulness of eHealth research software. In particular, the dissertation demonstrates action design research as a useful approach for developing research software. With regards to the software development practice, the dissertation highlights practices that encourage sustained usefulness.

Implication for Research

The extended list of quality characteristics, the typology for the fitness-utility model, the instantiation and validation ADR in the empirical context, with rich presentations of and reflections on the research process, and the AADR have implications for researchers and practitioners in design research. It was argued by Gill & Hevner (2013) that any design practice needs fitness [quality] characteristics that involve adaptation and evolution of design artefacts to sustain usefulness. The long-term fitness of the artefact can be extrapolated from a talk by David Lorge Parnas, way back in the 1994, as follows:

[96] 'As a wall' means that the typology would be available in public, where all stakeholders can access it, collaborate around it and provide feedback on it.

> A sign that the Software Engineering profession has matured will be that we lose our preoccupation with the first release and focus on the long-term health of our products. (1994)

The insides gathered and presented in this research establishes the importance of the quality characteristics in sustaining the usefulness of eHealth research software in an academic research context. Based on the learnings from the empirical grounding of the fitness-utility model in the multiple ADR cases in the U-CARE context and the retrospective analysis, it is proposed that the fitness-utility model may benefit from a revision of its candidate characteristics to include 'simplicity' and 'accountability' as new candidates, particularly in the eHealth research context. The proposed revisions are arguably too specific to be incorporated into the existing general model, considering that the current empirical evidence is based on only one of the possible IT artefacts (i.e., instantiations). Still, this empirical account of quality characteristics constitutes a contribution to the discourse on fitness-utility in DSR, primarily through a rich illustration of how quality characteristics and the U-CARE stakeholders influenced the emergence of an eHealth research software in an academic research context. Furthermore, this illustration is evidence of the validity of the fitness-utility model.

Recent advances in the DSR discourse – such as agile design science (Conboy *et al.*, 2015), the emergent nature of design science (Markus *et al.*, 2002; Pirkkalainen, 2015) and the four-cycle model, adding change and impact (Drechsler & Hevner, 2016) – have proposed changes/adaptations in order for the DSR process to cope with dynamics and time-related aspects. Alismail *et al.* (2017) demonstrated a framework for identifying DSR objectives. They represented the utility function, which considers either short- and medium-term or long-term planning horizons for deciding the artefact trait (i.e., quality characteristics) as:

> That is, if the researchers/practitioners are focusing on short-term and medium-term planning horizons, then they will be employing the usefulness evaluation model. On the other hand, the fitness-utility model is more compatible if they are focusing on a long-term planning horizon. (Alismail *et al.*, 2017, p. 225)

Utility for the short or medium term relates only to usefulness in fitness-utility model. For the long term, utility is the choice mechanisms guiding the artefact design and fitness (reproduction and evolution). This dissertation contributes to the DSR discourse by taking an ensemble view and showing that the researchers and practitioners need to consider sustaining usefulness which means that they have to balance short-term (usefulness) and long-term (fitness) goals in iterative BIE cycles in the organisational context, following the ADR method. In other words, ADR can be used for stratifying immediate usefulness needs in the organisation, as well as the long-term fitness of an ensemble artefact.

9.4 Future Work

This research provides insights from a longitudinal ADR project, with three ADR cases, on how quality characteristics and their enactment influence sustaining the usefulness of eHealth research software in an academic research context. Moreover, this research also suggests empirically grounded design principles for sustaining the usefulness of eHealth research software, based on a formalisation of learning in the three ADR cases. Furthermore, an augmented action design research (AADR) method, an extension of ADR, regarding how to conduct multiple ADR projects, was discussed in this research. In future research, the results of this project concerning sustaining the usefulness of eHealth research software can be studied, validated, and extended in different empirical context(s).

The future research can, for instance, investigate the quality characteristics and design principles for sustaining the usefulness of research software in general and taking into account wider groups of stakeholders, not only researchers and practitioners in the original context, such as secondary designers who would like to (re-)use and extend the research software. The future research can also concern a broader class of solutions such as research infrastructure, and study larger contexts, such as academic institutions at a national or even global organisational level. Furthermore, future research can be a collaboration between Information Systems researchers and designers in other fields (or design communities), such as computer science and engineering, as suggested by Gill and Hevner:

> we will have a strong incentive to collaborate with these [design] communities if we are to exert impact. Where we may be able to contribute most effectively is in our understanding of the potential unintended consequences of artefacts employed in an organizational setting. (2013, p. 17)

(☺)

References

AALST WMP van der, BICHLER M and HEINZL A (2016) Open Research in Business and Information Systems Engineering. *Business & Information Systems Engineering* **58(6)**, 375–379. Available at: http://link.springer.com/10.1007/s12599-016-0454-0.

ÅGERFALK PJ (2014) Insufficient theoretical contribution: a conclusive rationale for rejection? *European Journal of Information Systems* **23(6)**, 593–599. Available at: http://link.springer.com/10.1057/ejis.2014.35 (accessed 29/12/14).

ÅGERFALK PJ (2019) Stimulating academic discourse: a call for response. *European Journal of Information Systems* **28(1)**, 1–5. Available at: https://www.tandfonline.com/doi/full/10.1080/0960085X.2019.1557853.

ÅGERFALK PJ and WIBERG M (2018) Pragmatizing the normative artifact: Design science research in scandinavia and beyond. *Communications of the Association for Information Systems* **43(1)**, 68–77.

VAN AKEN JE (2004) Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. *Journal of Management Studies* **41(2)**, 219–246.

AKHLAGHPOUR S, WU J, LAPOINTE L and PINSONNEAULT A (2013) The ongoing quest for the IT artifact: Looking back, moving forward. *Journal of Information Technology* **28(2)**, 150–166. Available at: http://www.palgrave-journals.com/doifinder/10.1057/jit.2013.10 (accessed 18/06/14).

ALCAÑIZ M, BOTELLA C, BAÑOS RM, ZARAGOZA I and GUIXERES J (2009) The Intelligent e-Therapy system: a new paradigm for telepsychology and cybertherapy. *British Journal of Guidance & Counselling* **37(3)**, 287–296. Available at: http://www.tandfonline.com/doi/abs/10.1080/03069880902957015 (accessed 12/02/14).

ALISMAIL S, ZHANG H and CHATTERJEE S (2017) A Framework for Identifying Design Science Research Objectives for Building and Evaluating IT Artifacts. In *DESRIST 2017 Proceedings, LNCS 10243* pp 218–230. Available at: http://link.springer.com/10.1007/978-3-319-59144-5_13.

ALSHEIKH-ALI AA, QURESHI W, AL-MALLAH MH and IOANNIDIS JPA (2011) Public Availability of Published Research Data in High-Impact Journals (BOUTRON I, Ed). *PLoS ONE* **6(9)**, e24357. Available at: http://dx.plos.org/10.1371/journal.pone.0024357.

ALTURKI A and GABLE GG (2014) Theorizing in Design Science Research: An Abstraction Layers Framework. In *PACIS 2014 Proceedings* Available at: http://aisel.aisnet.org/pacis2014/126.

ALTURKI A, GABLE GG and BANDARA W (2013) The design science research roadmap: In progress evaluation. In *PACIS 2013 Proceedings* Available at: http://aisel.aisnet.org/pacis2013/160.

AMBATI V and KISHORE SP (2004) How can academic software research and open source software development help each other? In *The 4th Workshop on Open Source Software Engineering W8S Workshop - 26th International Conference on Software Engineering* pp 5–8, IEE, Edinburgh, UK. Available at: http://digital-library.theiet.org/content/conferences/10.1049/ic_20040256.

ANDER M, WIKMAN A, LJÓTSSON B, GRÖNQVIST H, LJUNGMAN G, WOODFORD J, LINDAHL NORBERG A and VON ESSEN L (2017) Guided internet-administered self-help to reduce symptoms of anxiety and depression among adolescents and young adults diagnosed with cancer during adolescence (U-CARE: YoungCan): a study protocol for a feasibility trial. *BMJ Open* **7(1)**, 1–11.

ANDERSSON G (2009) Using the Internet to provide cognitive behaviour therapy. *Behaviour Research and Therapy* **47(3)**, 175–80. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19230862 (accessed 25/08/14).

ANDERSSON G, CARLBRING P, LJÓTSSON B and HEDMAN E (2013) Guided Internet-Based CBT for Common Mental Disorders. *Journal of Contemporary Psychotherapy* **43(4)**, 223–233.

ANDERSSON G and TITOV N (2014) Advantages and limitations of Internet-based interventions for common mental disorders. *World Psychiatry* **13(1)**, 4–11.

ARZBERGER P, SCHROEDER P, BEAULIEU A, BOWKER G, CASEY K, LAAKSONEN L, MOORMAN D, UHLIR P and WOUTERS P (2004) Promoting Access to Public Research Data for Scientific, Economic, and Social Development. *Data Science Journal* **3(29)**, 135–152. Available at: http://www.ics.uci.edu/~gbowker/promoting access.pdf.

ASPIN A (2012) Exporting Data from SQL server. In *SQL Server 2012 Data Integration Recipes: Solutions for Integration Services and Other ETL Tools* pp 343–424, Apress, New York.

BAASTERLAR KMP, POUWER F, CUIJPERS P, RIPER H and SNOEK FJ (2011) Web-Based Depression Treatment for Type 1 and Type 2 Diabetic Patients. *Diabetes Care* **34**, 320–325.

BARAK A and GROHOL JM (2011) Current and Future Trends in Internet-Supported Mental Health Interventions. *Journal of Technology in Human Services* **29(3)**, 155–196. Available at: http://www.tandfonline.com/doi/abs/10.1080/15228835.2011.616939 (accessed 07/11/12).

BASKERVILLE RL, KAUL M and STOREY VC (2015) Genres of inquiry in design-science research: justification and evaluation of knowledge production. *MIS Quarterly* **39(3)**, 541–564.

BASKERVILLE RL, KAUL M and STOREY VC (2011) Unpacking the duality of design science. In *ICIS 2011 Proceedings* Shanghai. Available at: https://aisel.aisnet.org/icis2011/proceedings/generaltopics/10.

BASKERVILLE RL and MYERS MD (2002) Information Systems as a Reference Discipline. *MIS Quarterly* **26(1)**, 1–14.

BASKERVILLE RL and PRIES-HEJE J (2016) Discovering the Significance of Scientific Design Practice: New-Science Wrapped in Old-Science? In *European Conference on Information Systems (ECIS 2016)* Fraunhofer Publica, İstanbul,Turkey. Available at: https://aisel.aisnet.org/ecis2016_rp/135.

BASTAWROUS A and ARMSTRONG MJ (2013) Mobile health use in low- and high-income countries: an overview of the peer-reviewed literature. *Journal of the Royal Society of Medicine* **106(4)**, 130–42. Available at: http://www.ncbi.nlm.nih.gov/pubmed/23564897 (accessed 16/08/13).

BAXTER R, HONG NC, GORISSEN D, HETHERINGTON J and TODOROV I (2012) The Research Software Engineer. In *Digital Research* Oxford, UK. Available at: http://purl.org/net/epubs/work/63787.

DE BAYSER M, AZEVEDO LG and CERQUEIRA R (2015) ResearchOps: The case for DevOps in scientific applications. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* pp 1398–1404, IEEE. Available at: http://ieeexplore.ieee.org/document/7140503/.

BECK K and ANDRES C (2004) *Extreme Programming Explained: Embrace Change* (2nd edition). Addison-Wesley.

BECK K, GRENNING J, MARTIN RC, BEEDLE M, HIGHSMITH J, MELLOR S, BENNEKUM A van, HUNT A, SCHWABER K, COCKBURN A, JEFFRIES R, SUTHERLAND J, CUNNINGHAM W, KERN J, THOMAS D, FOWLER M and MARICK B (2001) *Manifesto for Agile Software Development*. [Online] Available at: http://agilemanifesto.org/.

BHIDE A, SHAH PS and ACHARYA G (2018) A simplified guide to randomized controlled trials. *Acta Obstetricia et Gynecologica Scandinavica* **97(4)**, 380–387.

BOOS D and GROTE G (2012) Designing controllable accountability of future internet of things applications. *Scandinavian Journal of Information Systems* **24(1)**, 3–28.

BROCKE J, FETTKE P, GAU M, HOUY C, MAEDCHE A, MORANA S and SEIDEL S (2017) *Tool-Support for Design Science Research: Design Principles and Instantiation*. [Online] Available at: https://ssrn.com/abstract=2972803.

CANNING PS, COOK WR, HILL WL and OLTHOFF WG (1989) Interfaces for strongly-typed object-oriented programming. *ACM SIGPLAN Notices* **24(10)**, 457–467. Available at: http://portal.acm.org/citation.cfm?doid=74878.74924.

CARROLL J (2004) Completing Design in Use: Closing the Appropriation Cycle. In *ECIS 2004 Proceedings* p 11. Available at: http://aisel.aisnet.org/ecis2004/44.

CARVER JC, GESING S, KATZ DS, RAM K and WEBER N (2018) Conceptualization of a US Research Software Sustainability Institute (URSSI). *Computing in Science and Engineering* **20(3)**, 4–9.

CHANDRA KRUSE L, SEIDEL S and PURAO S (2016) Making Use of Design Principles. In *DESRIST 2016 Proceedings, LNCS 9661* (PARSONS J., TUUNANEN T., VENABLE J., DONNELLAN B., HELFERT M. KJ, Ed), pp 37–51, Springer, Cham, St. John, Canada. Available at: http://link.springer.com/10.1007/978-3-319-39294-3_3.

CHANDRA L, SEIDEL S and GREGOR S (2015) Prescriptive Knowledge in IS Research: Conceptualizing Design Principles in Terms of Materiality, Action, and Boundary Conditions. In *48th Hawaii International Conference on System Sciences* pp 4039–4048, IEEE. Available at: http://ieeexplore.ieee.org/document/7070304/.

CHO H, GRAY J and SUN Y (2012) Quality-Aware Academic Research Tool Development. In *19th Asia-Pacific Software Engineering Conference* pp 66–72, IEEE, Hong Kong. Available at: http://ieeexplore.ieee.org/document/6462783/.

CLOUGH BA and CASEY LM (2011) Technological adjuncts to enhance current psychotherapy practices: a review. *Clinical Psychology Review* **31(3)**, 279–92.

COCKBURN A (2001) *Agile Software Development* (1st edition). Addison-Wesley Professional.

COENEN T, COERTJENS L, VLERICK P, LESTERHUIS M, MORTIER AV, DONCHE V, BALLON P and DE MAEYER S (2018) An information system design theory for the comparative judgement of competences. *European Journal of Information Systems* **27(2)**, 248–261. Available at: http://doi.org/10.1080/0960085X.2018.1445461.

COENEN T, DONCHE V and BALLON P (2015) LL-ADR : Action Design Research in Living Labs. In *Proceedings of the 48th Hawaii International Conference on System Sciences* pp 4029–4038, IEEE Computer Society, Kauai, Hawai.

COHN AM, HUNTER-REEL D, HAGMAN BT and MITCHELL J (2011) Promoting Behavior Change from Alcohol Use Through Mobile Technology: The Future of Ecological Momentary Assessment. *Alcoholism: Clinical and Experimental Research* **35**, 2209–2215.

CONBOY K, GLEASURE R and CULLINA E (2015) Agile Design Science Research. In *DESRIST 2015 Proceedings, LNCS 9073* pp 168–180. Available at: http://link.springer.com/10.1007/978-3-319-18714-3_11.

CRAIG P, DIEPPE P, MACINTYRE S, MICHIE S, NAZARETH I and PETTICREW M (2008) Developing and evaluating complex interventions: the new Medical Research Council guidance. *BMJ* **50(5)**, a1655. Available at: http://www.bmj.com/lookup/doi/10.1136/bmj.a1655.

CRESWELL JW and MILLER DL (2000) Determining Validity in Qualitative Inquiry. *Theory Into Practice* **39(3)**, 124–130. Available at: http://www.tandfonline.com/doi/abs/10.1207/s15430421tip3903_2.

CRICK T, HALL BA and ISHTIAQ S (2017) Reproducibility in Research: Systems, Infrastructure, Culture. *Journal of Open Research Software* **5**, 1–12. Available at: http://openresearchsoftware.metajnl.com/articles/10.5334/jors.73/.

CRONHOLM S and GÖBEL H (2015) Empirical Grounding of Design Science Research Methodology. In *New Horizons in Design Science: Broadening the Research Agenda: 10th International Conference, DESRIST 2015 Proceedings* pp 471–478. Available at: http://link.springer.com/10.1007/978-3-319-18714-3_40.

CRONHOLM S and GÖBEL H (2016) Evaluation of the Information Systems Research Framework: Empirical Evidence from a Design Science Research Project. *The Electronic Journal Information Systems Evaluation* **19(3)**, 158–168.

CROUCH S, HONG NC, HETTRICK S, JACKSON M, PAWLIK A, SUFI S, CARR L, DE ROURE D, GOBLE C and PARSONS M (2013) The Software Sustainability Institute: Changing Research Software Attitudes and Practices. *IEEE Computing in Science & Engineering* **15(6)**, 74–80. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6731384 (accessed 20/01/15).

CRUSOE MR and BROWN CT (2016) Walking the Talk: Adopting and Adapting Sustainable Scientific Software Development processes in a Small Biology Lab. *Journal of Open Research Software* **4(1)**, 4–9. Available at: http://openresearchsoftware.metajnl.com/articles/10.5334/jors.35/.

CRYER J (2013) *Resemble.js : Image analysis and comparison*. [Online] Available at: http://huddle.github.io/Resemble.js/.

CUIJPERS P, DONKER T, VAN STRATEN A, LI J and ANDERSSON G (2010) Is guided self-help as effective as face-to-face psychotherapy for depression and anxiety disorders? A systematic review and meta-analysis of comparative outcome studies. *Psychological Medicine* **40(12)**, 1943–1957.

CUIJPERS P, VAN STRATEN A and ANDERSSON G (2008) Internet-administered cognitive behavior therapy for health problems: A systematic review. *Journal of Behavioral Medicine* **31(2)**, 169–177.

DADGAR M, SAMHAN B and JOSHI KD (2013) Mobile Health Information Technology and Patient Care: A Literature Review and Analysis. In *AMCIS 2013 Proceedings* pp 1–10, Chicago, Illinois.

DALLMEIER-TIESSEN S, DARBY R, GITMANS K, LAMBERT S, MATTHEWS B, MELE S, SUHONEN J and WILSON M (2014) Enabling Sharing and Reuse of Scientific Data. *New Review of Information Networking* **19(1)**, 16–43. Available at: http://www.tandfonline.com/doi/abs/10.1080/13614576.2014.883936.

DAVISON RM and MARTINSONS MG (2016) Context is king! Considering particularism in research design and reporting. *Journal of Information Technology* **31(3)**, 241–249.

DELONE WH and MCLEAN ER (1992) Information Systems Success: The Quest for the Dependent Variable. *Information Systems Research* **3(1)**, 60–95.

DELONE WH and MCLEAN ER (2003) The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems* **19(4)**, 9–30.

DEWEY J (1938) *Logic: The Theory of Inquiry*. H. Holt and Company, New York.

DONKER T, PETRIE K, PROUDFOOT J, CLARKE J, BIRCH M-R and CHRISTENSEN H (2013) Smartphones for Smarter Delivery of Mental Health Programs: A Systematic Review. *Journal of Medical Internet Research* **15(11)**, e247. Available at: http://www.jmir.org/2013/11/e247/ (accessed 19/11/13).

DOUPI P, RENKO E, GIEST S and DUMORTIER J (2010) *Country Brief: Sweden*.

DOWNS RR, LENHARDT WC, ROBINSON E, DAVIS E and WEBER N (2015) Community Recommendations for Sustainable Scientific Software. *Journal of Open Research Software* **3(11)**, e11.

DOYLE C, LUCZAK-ROESCH M and MITTAL A (2019) *We need the open artefact: Design Science as a pathway to Open Science in Information Systems research*. Available at: https://osf.io/ye6xp.

DRECHSLER A and HEVNER A (2016) A four-cycle model of IS design science research: capturing the dynamic nature of IS artifact design. In *Breakthroughs and Emerging Insights from Ongoing Design Science Projects: Research-in-progress papers and poster presentations DESRIST* pp 1–8, St. John, Canada. Available at: https://cora.ucc.ie/handle/10468/2560.

EU (2016) *All European scientific articles to be freely accessible by 2020 – The Netherlands EU presidency press release*. [Online] Available at: http://english.eu2016.nl/documents/press-releases/2016/05/27/all-europeanscientific-articles-to-be-freely-accessible-by-2020 (accessed 27/10/16).

EYSENBACH G (2011) CONSORT-EHEALTH: Improving and Standardizing Evaluation Reports of Web-based and Mobile Health Interventions. *Journal of Medical Internet Research* **13(4)**, e126. Available at: http://www.jmir.org/2011/4/e126/.

EYSENBACH G (2001) What is e-health. *Journal of Medical Internet Research* **3(2)**, e20.

FITZGERALD B, STOL K-J, O'SULLIVAN R and O'BRIEN D (2013) Scaling agile methods to regulated environments: An industry case study. In *35th International Conference on Software Engineering (ICSE)* pp 863–872, IEEE, San Francisco. Available at: http://ieeexplore.ieee.org/document/6606635/.

FOGEL K (2005) *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly Media, Inc. Available at: https://producingoss.com/en/producingoss.pdf.

FOWLER M (2018) *Refactoring: Improving the Design of Existing Code* (2nd edition). Addison-Wesley Professional.

FOWLER M, BECK K, BRANT J, OPDYKE W and ROBERTS D (1999) *Refactoring: Improving the Design of Existing Code* (1st edition). Addison-Wesley Professional.

FREE C, PHILLIPS G, GALLI L, WATSON L, FELIX L, EDWARDS P, PATEL V and HAINES A (2013) The effectiveness of mobile-health technology-based health behaviour change or disease management interventions for health care consumers: a systematic review. (CORNFORD T, Ed). *PLoS Medicine* **10(1)**, e1001362. Available at: http://dx.plos.org/10.1371/journal.pmed.1001362 (accessed 18/10/13).

FREE C, PHILLIPS G, WATSON L, GALLI L, FELIX L, EDWARDS P, PATEL V and HAINES A (2013) The effectiveness of mobile-health technologies to improve health care service delivery processes: a systematic review and meta-analysis. (CORNFORD T, Ed). *PLoS Medicine* **10(1)**, e1001363. Available at: http://dx.plos.org/10.1371/journal.pmed.1001363 (accessed 18/10/13).

GAMMA E, HELM R, JOHNSON R and VLISSIDES J (1995) *Design Patterns: Elements of Reusable Object-Oriented Software* (1st edition). Addison-Wesley Professional.

VAN GEMERT-PIJNEN JEWC, NIJLAND N, VAN LIMBURG M, OSSEBAARD HC, KELDERS SM, EYSENBACH G and SEYDEL ER (2011) A holistic framework to improve the uptake and impact of eHealth technologies. *Journal of Medical Internet Research* **13(4)**, 1–26.

GENTLEMAN R and LANG DT (2004) *Statistical Analyses and Reproducible Research*. Available at: http://biostats.bepress.com/bioconductor/paper2.

GILL TG and HEVNER AR (2011) A fitness-utility model for design science research. In *DESRIST 2011 Proceedings, LNCS 6629* (JAIN H, SINHA AP, & VITHARANA P, Eds), pp 237–252, Berlin Heidelberg.

GILL TG and HEVNER AR (2013) A Fitness-Utility Model for Design Science Research. *ACM Transactions on Management Information Systems* **4(2)**.

GLASGOW RE, PHILLIPS SM and SANCHEZ MA (2014) Implementation science approaches for integrating eHealth research into practice and policy. *International Journal of Medical Informatics* **83(7)**, e1–e11. Available at: http://dx.doi.org/10.1016/j.ijmedinf.2013.07.002.

GOBLE C (2014) Better Software, Better Research. *IEEE Internet Computing* **18(5)**, 4–8.

GREGOR S (2002) Design theory in Information Systems. *Australian Journal of Information Systems* **9(Special Edition)**, 14–22.

GREGOR S (2006) The nature of theory in information systems. *MIS Quarterly* **30(3)**, 611–642.

GREGOR S and HEVNER AR (2013) Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly* **37(2)**, 337–355.

GREGOR S and IIVARI J (2007) Designing for Mutability in Information Systems Artifacts. In *Information Systems II: Theory, Representation and Reality* (GREGOR S & HART D, Eds), pp 3–24, ANU Press. Available at: https://www.jstor.org/stable/j.ctt24hdw8.5.

GREGOR S and JONAS D (2007) The Anatomy of a Design Theory. *Journal of the Association for Information Systems* **8(5)**, 312–335.

GREGORY J, NURMINEN M, ORR J, ROBERTSON T, STAR SL and SUCHMAN L (2003) Scandinavian Approaches to Participatory Design. *International Journal of Engineering Education* **19(1)**, 62–74.

GROEN D, GUO X, GROGAN JA, SCHILLER UD and OSBORNE JM (2015) Software development practices in academia: a case study comparison. In *Proceedings of Cornel University Library* p 22. Available at: http://arxiv.org/abs/1506.05272.

GRÖNQVIST H, OLSSON EMG, JOHANSSON B, HELD C, SJÖSTRÖM J, LINDAHL NORBERG A, HOVÉN E, SANDERMAN R, VAN ACHTERBERG T and VON ESSEN L (2017) Fifteen Challenges in Establishing a Multidisciplinary Research Program on eHealth Research in a University Setting: A Case Study. *Journal of Medical Internet Research* **19(5)**, e173. Available at: http://www.jmir.org/2017/5/e173/.

GURMAN TA, RUBIN SE and ROESS AA (2012) Effectiveness of mHealth Behavior Change Communication Interventions in Developing Countries: A Systematic Review of the Literature. *Journal of Health Communication* **17(sup1)**, 82–104. Available at: http://www.tandfonline.com/doi/abs/10.1080/10810730.2011.649160 (accessed 15/07/12).

HAJ-BOLOURI A, BERNHARDSSON L and ROSSI M (2016) PADRE: A Method for Participatory Action Design Research. In *Tackling Society's Grand Challenges with Design Science: 11th International Conference, DESRIST 2016* (PARSONS J, TUUNANEN T, VENABLE J, DONNELLAN B, HELFERT M, & KENNEALLY J, Eds), pp 19–36, Springer International Publishing. Available at: http://link.springer.com/10.1007/978-3-319-39294-3_2.

HAJ-BOLOURI A, PURAO S, ROSSI M and BERNHARDSSON L (2017) Action Design Research as a Method-in-Use : Problems and Opportunities. In *DESRIST 2017 Proceedings, LNCS 10243* pp 1–9.

HARIHARAN A, ADAM MTP, DORNER V, LUX E, MUELLER MB, PFEIFFER J and WEINHARDT C (2017) Brownie: A Platform for Conducting NeuroIS Experiments. *Journal of the Association for Information Systems* **18(4)**, 264–296.

HARRIS PA, TAYLOR R, THIELKE R, PAYNE J, GONZALEZ N and CONDE JG (2009) Research electronic data capture (REDCap)--a metadata-driven methodology and workflow process for providing translational research informatics support. *Journal of biomedical informatics* **42(2)**, 377–81. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2700030&tool=pmcentrez&rendertype=abstract (accessed 18/03/13).

HARRISON V, PROUDFOOT J, WEE PP, PARKER G, PAVLOVIC DH and MANICAVASAGAR V (2011) Mobile mental health: Review of the emerging field and proof of concept study. *Journal of Mental Health* **20**, 509–524.

HASTINGS J, HAUG K and STEINBECK C (2014) Ten recommendations for software engineering in research. *GigaScience* **3(1)**, 31. Available at: http://gigascience.biomedcentral.com/articles/10.1186/2047-217X-3-31.

HEDMAN E, LJOTSSON B and LINDEFORS N (2012) Cognitive behavior therapy via the Internet: a systematic review of applications, clinical efficacy and cost-effectiveness. *Expert Rev Pharmacoecon Outcomes Res* **12**, 745–64.

HELFERT M, DONNELLAN B and OSTROWSKI L (2012) The Case for Design Science Utility and Quality - Evaluation of Design Science Artifact within the Sustainable ICT Capability Maturity Framework. *Systems, Signs and Actions* **6(1)**, 46–66. Available at: http://www.sysiac.org/.

HETTRICK S (2016) *Research Software Sustainability: Report on a Knowledge Exchange Workshop*. Available at: http://digitalcommons.unl.edu/scholcom/6/.

HEVNER A and CHATTERJEE S (2010) *Design Research in Information Systems*. (SHARDA R & VOß S, Eds). Springer US, Boston, MA. Available at: http://link.springer.com/10.1007/978-1-4419-5653-8.

HEVNER AR (2007) A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems* **19(2)**, 87–92.

HEVNER AR, DONNELLAN B and ANDERSON J (2013) The DRIVES (Design Research for Innovation Value, Evaluation, and Sustainability) Model of Innovation. In *Design Science: Perspectives from Europe (EDSS 2012). Communications in Computer and Information Science* (HELFERT M & DONNELLAN B, Eds), pp 144–154, Springer International Publishing. Available at: http://link.springer.com/10.1007/978-3-319-04090-5_13.

HEVNER AR, MARCH ST, PARK J and RAM S (2004) Design science in information systems research. *MIS Quarterly* **28(1)**, 75–105.

HOFMANN SG, ASNAANI A, VONK IJJ, SAWYER AT and FANG A (2012) The Efficacy of Cognitive Behavioral Therapy: A Review of Meta-analyses. *Cognitive Therapy and Research* **36(5)**, 427–440. Available at: http://link.springer.com/10.1007/s10608-012-9476-1.

HOVORKA DS and PRIES-HEJE J (2013) Don't Ignore the Iceberg: Timely Revelation of Justification in DSR. In *DESRIST 2013 Proceedings, LNCS 7939* (BROCKE J VOM, Ed), pp 228–241, Springer, Helsinki, Finland. Available at: http://link.springer.com/10.1007/978-3-642-38827-9_16.

IIVARI J (2015) Distinguishing and contrasting two strategies for design science research. *European Journal of Information Systems* **24(1)**, 107–115. Available at: http://link.springer.com/10.1057/ejis.2013.35 (accessed 14/05/15).

ISO/IEC/IEEE (2017) ISO/IEC/IEEE International Standard - Systems and software engineering--Vocabulary. In *ISO/IEC/IEEE 24765:2017(E)* pp 1–541, IEEE. Available at: https://ieeexplore.ieee.org/servlet/opac?punumber=8016710.

ISO/IEC (2011) Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. In *ISO/IEC 25010:2011* p 34, ISO.

JÄRVINEN P (2007) Action Research is Similar to Design Science. *Quality & Quantity* **41(1)**, 37–54. Available at: http://www.springerlink.com/index/10.1007/s11135-005-5427-1 (accessed 12/11/12).

JIMÉNEZ RC, KUZAK M, ALHAMDOOSH M, BARKER M, BATUT B, BORG M, CAPELLA-GUTIERREZ S, CHUE HONG N, COOK M, CORPAS M, FLANNERY M, GARCIA L, GELPÍ JL, GLADMAN S, GOBLE C, GONZÁLEZ FERREIRO M, GONZALEZ-BELTRAN A, GRIFFIN PC, GRÜNING B, HAGBERG J, HOLUB P, HOOFT R, ISON J, KATZ DS, LESKOŠEK B, LÓPEZ GÓMEZ F, OLIVEIRA LJ, MELLOR D, MOSBERGEN R, MULDER N, PEREZ-RIVEROL Y, PERGL R, PICHLER H, POPE B, SANZ F, SCHNEIDER M V., STODDEN V, SUCHECKI R, SVOBODOVÁ VAŘEKOVÁ R, TALVIK H-A, TODOROV I, TRELOAR A, TYAGI S, VAN GOMPEL M, VAUGHAN D, VIA A, WANG X, WATSON-HAIGH NS and CROUCH S (2017) Four simple recommendations to encourage best practices in research software. *F1000Research* **6**, 876. Available at: https://f1000research.com/articles/6-876/v1.

KATZ DS, CHOI S-CT, LAPP H, MAHESHWARI K, LÖFFLER F, TURK M, HANWELL MD, WILKINS-DIEHR N, HETHERINGTON J, HOWISON J, SWENSON S, ALLEN GD, ELSTER AC, BERRIMAN B and VENTERS C (2014) Summary of the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1). *Journal of Open Research Software* **2(1)**, 1–21.

KATZ DS, CHOI S-CT, WILKINS-DIEHR N, CHUE HONG N, VENTERS CC, HOWISON J, SEINSTRA F, JONES M, CRANSTON K, CLUNE TL, DE VAL-BORRO M and LITTAUER R (2016a) Report on the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2). *Journal of Open Research Software* **4(e7)**, 1–23.

KATZ DS, CHOI S-CT, WILKINS-DIEHR N, CHUE HONG N, VENTERS CC, HOWISON J, SEINSTRA F, JONES M, CRANSTON K, CLUNE TL, DE VAL-BORRO M and LITTAUER R (2016b) Report on the Third Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE3). *Journal of Open Research Software* **4(e37)**, 1–23.

KLEIN HK and MYERS MD (1999) A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly* **23(1)**, 67–94.

KRAFT P, DROZD F and OLSEN E (2009) ePsychology: Designing theory-based health promotion interventions. *Communications of the Association for Information Systems* **24(1)**, 399–426.

KRIPPENDORFF K (2006) *The Semantic Turn: A New Foundation for Design*. CRC Press, Boca Raton, FL.

KRUCHTEN P, NORD RL and OZKAYA I (2012) Technical Debt: From Metaphor to Theory and Practice. *IEEE Software* **29(6)**, 18–21. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6336722 (accessed 20/07/14).

LAINE C, GOODMAN SN, GRISWOLD ME and SOX HC (2007) Reproducible Research: Moving toward Research the Public Can Really Trust. *Annals of Internal Medicine* **146(6)**, 450. Available at: http://annals.org/article.aspx?doi=10.7326/0003-4819-146-6-200703200-00154.

LAKEW N (2013) Sustaining IT Usefulnes – Re-defining end users' role as contextual designers. In *European Design Science Symposium (EDSS)* pp 123–134, Springer, Dublin, Ireland.

LEE AS and BASKERVILLE RL (2003) Generalizing Generalizability in Information Systems Research. *Information Systems Research* **14(3)**, 221–243. Available at: http://isr.journal.informs.org/cgi/doi/10.1287/isre.14.3.221.16560.

LEONARDI PM (2012) Materiality, Sociomateriality, and Socio-Technical Systems: What Do These Terms Mean? How Are They Related? Do We Need Them? In *Materiality and Organizing: Social Interaction in a Technological World* (LEONARDI PM, NARDI BA, & KALLINIKOS J, Eds), pp 25–48, Oxford: Oxford University Press.

LIPSITZ LA (2012) Understanding Health Care as a Complex System. *Journal of American Medical Association* **308(3)**, 243–244. Available at: http://archsurg.jamanetwork.com/data/Journals/JAMA/24475/jvp120042_243_244.pdf.

LIU DLD, XU SXS and BROCKMEYER M (2008) Investigation on Academic Research Software Development. In *International Conference on Computer Science and Software Engineering* pp 626–630.

LO B and DEMETS DL (2016) Incentives for Clinical Trialists to Share Data. *New England Journal of Medicine* **375(12)**, 1112–1115. Available at: http://www.nejm.org/doi/10.1056/NEJMp1608351.

LOCHAN R (2012) Technology and the Internet as a key component in psychosocial care of Somatic diseases. In *AMCIS 2012 Proceedings* Seattle, Washington. Available at: http://aisel.aisnet.org/amcis2012/proceedings/ISHealthcare/18.

LUMSDEN J and MORGAN W (2005) Online-Questionnaire Design : Establishing Guidelines and Evaluating Existing Support. In *Proceedings of the 16th Annual International Conference of the Information Resources Management Organization (IRMA 2005)* (KHOSROW-POUR M, Ed), p 1358, San Diego, California. Available at: http://www.irma-international.org/viewtitle/32623/.

LUND J (2014) Activities to address challenges in digital innovation. In *Working Conference on Information Systems and Organizations, IFIP Advances in Information and Communication Technology* pp 115–131.

LUNDBERG N, KOCH S, HÄGGLUND M, BOLINA P, DAVOODY N, ELTES J, JARLMAN O, PERLICH A, VIMARLUND V and WINSNES C (2013) My care pathways - Creating open innovation in healthcare. *Studies in Health Technology and Informatics* **192(1–2)**, 687–691.

LUXTON DD, MCCANN RA, BUSH NE, MISHKIND MC and REGER GM (2011) mHealth for mental health: Integrating smartphone technology in behavioral healthcare. *Professional Psychology: Research and Practice* **42(6)**, 505–512. Available at: http://doi.apa.org/getdoi.cfm?doi=10.1037/a0024485 (accessed 15/07/12).

MAGNUSSON J and BYGSTAD B (2014) Technology debt: toward a new theory of technology heritage. In *ECIS 2014 Proceedings* pp 0–15.

MARCH ST and SMITH GF (1995) Design and natural science research on information technology. *Decision Support Systems* **15(4)**, 251–266. Available at: http://www.sciencedirect.com/science/article/pii/0167923694000412 (accessed 18/09/14).

MARKUS ML, MAJCHRZAK A and GASSER L (2002) A design theory for systems that support emergent knowledge processes. *MIS Quarterly* **26(3)**, 179–212.

MARTIN RC (2000) *Design Principles and Design Patterns*. http://www.objectmentor.com.

MATHIASSEN L (2002) Collaborative practice research. *Information Technology & People* **15(4)**, 321–345. Available at: http://www.emeraldinsight.com/doi/10.1108/09593840210453115 (accessed 01/11/12).

MATTSSON S, ALFONSSON S, CARLSSON M, NYGREN P, OLSSON E and JOHANSSON B (2013) U-CARE: Internet-based stepped care with interactive support and cognitive behavioral therapy for reduction of anxiety and depressive symptoms in cancer - a clinical trial protocol. *BMC Cancer* **13(1)**, 414. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3848442.

MCKENNEY S and REEVES TC (2012) *Conducting Educational Design Research*. Routledge, London.

MESIROV JP (2010) Accessible Reproducible Research. *Science* **327(5964)**, 415–416. Available at: http://www.sciencemag.org/cgi/doi/10.1126/science.1179653.

MEWTON L, SMITH J, ROSSOUW P and ANDREWS G (2014) Current perspectives on Internet-delivered cognitive behavioral therapy for adults with anxiety and related disorders. *Psychology research and behavior management* **7**, 37–46. Available at: http://www.dovepress.com/current-perspectives-on-internet-delivered-cognitive-behavioral-therap-peer-reviewed-article-PRBM (accessed 17/02/14).

MICHIE S, YARDLEY L, WEST R, PATRICK K and GREAVES F (2017) Developing and Evaluating Digital Interventions to Promote Behavior Change in Health and Health Care: Recommendations Resulting From an International Workshop. *Journal of Medical Internet Research* **19(6)**, e232. Available at: http://www.jmir.org/2017/6/e232/.

MILLER G (2012) The Smartphone Psychology Manifesto. *Perspectives on Psychological Science* **7(3)**, 221–237. Available at: http://pps.sagepub.com/lookup/doi/10.1177/1745691612441215 (accessed 02/03/13).

MINISTRY OF HEALTH AND SOCIAL AFFAIRS (2010) *National eHealth (Sweden) - the strategy for accessible and secure information in health and social care*. Available at: http://www.government.se/content/1/c6/16/79/85/8d4e6161.pdf.

MOHR DC, SCHUELLER SM, RILEY WT, BROWN CH, CUIJPERS P, DUAN N, KWASNY MJ, STILES-SHIELDS C and CHEUNG K (2015) Trials of intervention principles: Evaluation methods for evolving behavioral intervention technologies. *Journal of Medical Internet Research* **17(7)**.

MORANA S, VOM BROCKE J, MAEDCHE A, SEIDEL S, ADAM MTP, BUB U, FETTKE P, GAU M, HERWIX A, MULLARKEY MT, NGUYEN HD, SJÖSTRÖM J, TOREINI P, WESSEL L and WINTER R (2018) Tool Support for Design Science Research—Towards a Software Ecosystem: A Report from a DESRIST 2017 Workshop. *Communications of the Association for Information Systems* **43(1)**, 237–256.

MORIN A, URBAN J and SLIZ P (2012) A quick guide to software licensing for the scientist-programmer. *PLoS Computational Biology* **8(7)**.

MULLARKEY MT and HEVNER AR (2018) An elaborated action design research process model. *European Journal of Information Systems* **9344**, 1–15. Available at: http://doi.org/10.1080/0960085X.2018.1451811.

MULLARKEY MT and HEVNER AR (2015) Entering Action Design Research. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* pp 121–134. Available at: http://link.springer.com/10.1007/978-3-319-18714-3_8.

MUNAFÒ MR, NOSEK BA, BISHOP DVM, BUTTON KS, CHAMBERS CD, PERCIE DU SERT N, SIMONSOHN U, WAGENMAKERS E, WARE JJ and IOANNIDIS JPA (2017) A manifesto for reproducible science. *Nature Human Behaviour* **1(1)**, 0021. Available at: http://www.nature.com/articles/s41562-016-0021.

MURRAY-RUST P (2008) Open data in science. *Serials Review* **34(1)**, 52–64.

MUSTAFA MI and SJÖSTRÖM J (2013) Design Principles for Research Data Export: Lessons Learned in e-Health Design Research. In *DESRIST 2013 Proceedings, LNCS 7939* (BROCKE J, HEKKALA R, RAM S, & ROSSI M, Eds), pp 34–49, Springer Berlin Heidelberg. Available at: http://dx.doi.org/10.1007/978-3-642-38827-9_3.

MUSTAFA MI, SJÖSTRÖM J and ERIKSSON-LUNDSTRÖM J (2014) An Empirical Account of Fitness-Utility: A Case of Radical Change towards Mobility in DSR Practice. In *DESRIST 2014 Proceedings, LNCS 8463* (TREMBLAY MC, VANDERMEER D, ROTHENBERGER M, GUPTA A, & YOON V, Eds), pp 289–303, Springer International Publishing.

MYERS MD and VENABLE JR (2014) A set of ethical principles for design science research in information systems. *Information & Management* **51(6)**, 801–809. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0378720614000081 (accessed 16/12/14).

NATURE (2008) Beta blockers? *Nature* **455(October)**, 708.

NICKERSON RC, VARSHNEY U and MUNTERMANN J (2013) A method for taxonomy development and its application in information systems. *European Journal of Information Systems* **22(3)**, 336–359.

NIEDERMAN F and MARCH ST (2012) Design science and the accumulation of knowledge in the information systems discipline. *ACM Transactions on Management Information Systems* **3(1)**, 1–15. Available at: http://dl.acm.org/citation.cfm?doid=2151163.2151164.

NIJLAND N (2011) *Grounding eHealth - Towards a holistic framework for sustainable eHealth technologies*. University of Twente: Enschede. Available at: http://doc.utwente.nl/75576/1/thesis_N_Nijland.pdf.

NORLUND F, OLSSON EM, BURELL G, WALLIN E and HELD C (2015) Treatment of depression and anxiety with internet-based cognitive behavior therapy in patients with a recent myocardial infarction (U-CARE Heart): study protocol for a randomized controlled trial. *Trials* **16(1)**, 1–8. Available at: http://www.trialsjournal.com/content/16/1/154.

NOSEK BA, ALTER G, BANKS GC, BORSBOOM D, BOWMAN SD, BRECKLER SJ, BUCK S, CHAMBERS CD, CHIN G, CHRISTENSEN G, CONTESTABILE M, DAFOE A, EICH E, FREESE J, GLENNERSTER R, GOROFF D, GREEN DP, HESSE B, HUMPHREYS M, ISHIYAMA J, KARLAN D, KRAUT A, LUPIA A, MABRY P, MADON T, MALHOTRA N, MAYO-WILSON E, MCNUTT M, MIGUEL E, PALUCK EL, SIMONSOHN U, SODERBERG C, SPELLMAN BA, TURITTO J, VANDENBOS G, VAZIRE S, WAGENMAKERS EJ, WILSON R and YARKONI T (2015) Promoting an open research culture. *Science* **348(6242)**, 1422–1425. Available at: http://www.sciencemag.org/cgi/doi/10.1126/science.aab2374.

NUNAMAKER JF, TWYMAN NW, GIBONEY JS and BRIGGS RO (2017) Creating High-Value Real-World Impact through Systematic Programs of Research. *MIS Quarterly* **41(2)**, 335–351.

NYLANDER S, LUNDQUIST T, BRÄNNSTRÖM A and KARLSON B (2009) "It's Just Easier with the Phone" – A Diary Study of Internet Access from Cell Phones. In *Pervasive 2009, LNCS 5538* pp 354–371, Springer. Available at: http://link.springer.com/10.1007/978-3-642-01516-8_24.

ORLIKOWSKI WJ (2010) The sociomateriality of organisational life: considering technology in management research. *Cambridge Journal of Economics* **34(1)**, 125–141. Available at: http://cje.oxfordjournals.org/cgi/doi/10.1093/cje/bep058 (accessed 04/11/12).

ORLIKOWSKI WJ and BAROUDI JJ (1991) Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research* **2(1)**, 1–28. Available at: http://isr.journal.informs.org/cgi/doi/10.1287/isre.2.1.1.

ORLIKOWSKI WJ and IACONO CS (2001) Research commentary : Desperately seeking 'IT' in IT research - A call to theorizing the IT artifact. *Information Systems Research* **12(2)**, 121–134.

PAGLIARI C, SLOAN D, GREGOR P, SULLIVAN F, DETMER D, KAHAN JP, OORTWIJN W and MACGILLIVRAY S (2005) What is eHealth (4): a scoping exercise to map the field. *Journal of Medical Internet Research* **7(1)**, e9. Available at: http://www.jmir.org/2005/1/e9/ (accessed 25/04/15).

PARNAS DL (1972) On the criteria to be used in decomposing systems into modules. *Communications of the ACM* **15(12)**, 1053–1058. Available at: http://dl.acm.org/citation.cfm?id=361598.361623 (accessed 17/07/14).

PARNAS DL (1994) Software aging. In *Proceedings of the 16th international conference on Software engineering* pp 279–287, IEEE Computer Society Press. Available at: http://dl.acm.org/citation.cfm?id=257734.257788 (accessed 30/08/14).

PARNAS DLL (1976) On the Design and Development of Program Families. *IEEE Transactions on Software Engineering* **SE-2(1)**, 1–9.

PEARSON S and CHARLESWORTH A (2009) Accountability as a Way Forward for Privacy Protection in the Cloud. In *Cloud Computing, LNCS 5931* (JAATUN M, ZHAO G, & RONG C, Eds), Lecture Notes in Computer Science. pp 131–144, Springer Berlin Heidelberg. Available at: http://dx.doi.org/10.1007/978-3-642-10665-1_12.

PEFFERS K, TUUNANEN T, ROTHENBERGER MA and CHATTERJEE S (2007) A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems* **24(3)**, 45–77. Available at: http://mesharpe.metapress.com/openurl.asp?genre=article&id=doi:10.2753/MIS0742-1222240302 (accessed 26/10/12).

PENG RD (2011) Reproducible Research in Computational Science. *Science* **334(6060)**, 1226–1227. Available at: http://www.sciencemag.org/cgi/doi/10.1126/science.1213847.

PETERS GY, ABRAHAM C and CRUTZEN R (2012) Full disclosure : doing behavioural science necessitates sharing. *The European Health Psychologist* **14(4)**, 77–84.

PHAM Q, WILJER D and CAFAZZO JA (2016) Beyond the Randomized Controlled Trial: A Review of Alternatives in mHealth Clinical Trial Methods. *JMIR mHealth and uHealth* **4(3)**, e107. Available at: http://mhealth.jmir.org/2016/3/e107/.

PIRKKALAINEN H (2015) Dealing With Emergent Design Science Research Projects in IS. In *DESRIST 2015 Proceedings, LNCS 9073* (DONNELLAN B, GLEASURE R, HELFERT M, KENNEALLY J, ROTHENBERGER M, CHIARINI TREMBLAY M, VANDERMEER D, & WINTER R, Eds), pp 61–68, Dublin, Ireland.

PLSEK P (2003) Complexity and the Adoption of Innovation in Health Care Complexity and the Adoption of Innovation in Health Care. In *Accelerating Quality Improvement in Health Care: Strategies to Speed the Diffusion of Evidence-Based Innovations* National Institute for Healthcare Management Foundation and National Committee for Quality in Health Care, Washington, D.C. Available at: https://www.nihcm.org/pdf/Plsek.pdf.

PONTIKA N, KNOTH P, CANCELLIERI M and PEARCE S (2015) Fostering open science to research using a taxonomy and an eLearning portal. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business - i-KNOW '15* pp 1–8, ACM Press, New York, USA. Available at: http://dl.acm.org/citation.cfm?doid=2809563.2809571.

PRAT N, COMYN-WATTIAU I and AKOKA J (2015) A Taxonomy of Evaluation Methods for Information Systems Artifacts. *Journal of Management Information Systems* **32(3)**, 229–267. Available at: http://www.tandfonline.com/doi/abs/10.1080/07421222.2015.1099390#.VqXhCYUrLmE.

PRAT N, COMYN-WATTIAU I and AKOKA J (2014) Artifact evaluation in information systems design science research – a holistic view. In *PACIS 2014 Proceedings* Chengdu, China. Available at: https://aisel.aisnet.org/pacis2014/23.

PREECE J, ROGERS Y and SHARP H (2002) *Interaction design: beyond human-computer interaction* (2nd edition). John Wiley & Sons, Inc. New York, USA. Available at: http://www.lavoisier.fr/livre/notice.asp?ouvrage=1338089 (accessed 06/03/13).

PRESSMAN RS and MAXIM BR (2014) *Software Engineering: A Practitioner's Approach* (8th edition). McGraw-Hill Education.

PURAO S, HENFRIDSSON O, ROSSI M and SEIN MK (2013) Ensemble Artifacts : From Viewing to Designing in Action Design Research. *Systems, Signs and Actions* **7(1)**, 73–81.

RAPTIS DA, METTLER T, FISCHER MA, PATAK M, LESURTEL M, ESHMUMINOV D, DE ROUGEMONT O, GRAF R, CLAVIEN P-A and BREITENSTEIN S (2014) Managing multicentre clinical trials with open source. *Informatics for Health and Social Care* **39(2)**, 67–80. Available at: http://www.tandfonline.com/doi/full/10.3109/17538157.2013.812647.

RINI C, WILLIAMS DA, BRODERICK JE and KEEFE FJ (2012) Meeting them where they are: Using the Internet to deliver behavioral medicine interventions for pain. *Translational Behavioral Medicine* **2**, 1–11.

ROGERS EM (2003) *Diffusion of innovations* (5th edition). Free Press, New York.

ROMME AGL (2003) Making a difference: Organization as design. *Organ. Sci* **14(5)**, 14558–573.

VAN ROOIJ T and MARSH S (2016) eHealth: past and future perspectives. *Personalized Medicine* **13(1)**, 57–70.

ROSS JS, LEHMAN R and GROSS CP (2012) The importance of clinical trial data sharing: Toward more open science. *Circulation: Cardiovascular Quality and Outcomes* **5(2)**, 238–240.

ROSSI M, HENFRIDSSON O, LYYTINEN K and SIAU K (2013) Design Science Research: The Road Traveled and the Road That Lies Ahead. *Journal of Database Management* **24(3)**, 1–8. Available at: http://www.igi-global.com/article/design-science-research/94541 (accessed 25/01/14).

SARKER S, XIAO X and BEAULIEU T (2013) Qualitative Studies in Information Systems : A Critical Review and Some Guiding Principles. *MIS Quarterly* **37(4)**, iii–xviii.

SCHWAB M, KARRENBACH N and CLAERBOUT J (2000) Making scientific computations reproducible. *Computing in Science & Engineering* **2(6)**, 61–67. Available at: http://ieeexplore.ieee.org.ezp-prod1.hul.harvard.edu/ielx5/5992/19077/00881708.pdf?tp=&arnumber=881708&isnumber=19077%5Cnhttp://ieeexplore.ieee.org/document/881708/.

SEIN MK, HENFRIDSSON O, PURAO S, ROSSI M and LINDGREN R (2011) Action Design Research. *MIS Quarterly* **35(1)**, 37–56.

SEIN MK and ROSSI M (2019) Elaborating ADR while drifting away from its essence: A commentary on Mullarkey and Hevner. *European Journal of Information Systems* **28(1)**, 21–25.

SENAPATHI M and SRINIVASAN A (2012) Understanding post-adoptive agile usage: An exploratory cross-case analysis. *Journal of Systems and Software* **85(6)**, 1255–1268. Available at: http://www.sciencedirect.com/science/article/pii/S0164121212000489 (accessed 11/08/14).

SHULL F (2011) Perfectionists in a World of Finite Resources. *IEEE Software* **28(2)**, 4–6. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5720703 (accessed 07/08/14).

SIMON HA (1996) *The Sciences of the Artificial* (3rd edition). MIT Press, Cambridge, MA.

SJÖSTRÖM J (2017) DeProX: A Design Process Exploration Tool. In *DESRIST 2017 Proceedings, LNCS 10243* pp 447–451. Available at: http://link.springer.com/10.1007/978-3-319-59144-5_29.

SJÖSTRÖM J (2010) *Designing Information Systems: A Pragmatic Account*. Uppsala University: Uppsala. Available at: http://uu.diva-portal.org/smash/get/diva2:350209/FULLTEXT01.pdf.

SJÖSTRÖM J and ÅGERFALK PJ (2009) An Analytic Framework for Design-Oriented Research Concepts. In *AMCIS 2009 Proceedings* San Francisco, California. Available at: http://aisel.aisnet.org/amcis2009/302.

SJÖSTRÖM J and ÅGERFALK PJ (2013) Architecting Social Interaction : Experiences From E-Health Design Research in the U-Care Program. In *SIG Prag Workshop on IT Artefact Design & Workpractice Improvement* pp 1–14, Tilburg, the Netherlands.

SJÖSTRÖM J, AGERFALK PJ and HEVNER AR (2017) Scrutinizing Privacy and Accountability in Online Psychosocial Care. *IT Professional* **19(3)**, 45–51.

SJÖSTRÖM J, ÅGERFALK PJ and HEVNER AR (2014) The Design of a Multi-layer Scrutiny Protocol to Support Online Privacy and Accountability. In *DESRIST 2014 Proceedings, LNCS 8463* (TREMBLAY MC, VANDERMEER D, ROTHENBERGER M, GUPTA A, & YOON V, Eds), pp 85–98, Springer International Publishing, Cham.

SJÖSTRÖM J, ÅGERFALK PJ and LOCHAN R (2011) Mutability Matters: Baselining the Consequences of Design. In *Proceedings of MCIS 2011* Limassol, Cyprus. Available at: http://aisel.aisnet.org/mcis2011/33.

SJÖSTRÖM J and ALFONSSON S (2012) Supporting the therapist in online therapy. In *ECIS 2012 Proceedings* Available at: http://aisel.aisnet.org/ecis2012/69/ (accessed 06/03/13).

SJÖSTRÖM J, CHANDRA KRUSE L and HAJ-BOLOURI A (2016) *A Design Theory for Built-in Evaluation Support*.

SJÖSTRÖM J, DONNELLAN B and HELFERT M (2012) Product Semantics in Design Research Practice. In *Shaping the Future of ICT Research. Methods and Approaches* (BHATTACHERJEE A & FITZGERALD B, Eds), IFIP Advances in Information and Communication Technology. pp 35–48, Springer Berlin Heidelberg, Berlin, Heidelberg. Available at: http://www.springerlink.com/index/10.1007/978-3-642-35142-6 (accessed 16/01/14).

SJÖSTRÖM J, ERIKSSON O and ÅGERFALK PJ (2013) CoDisclose: An Approach to Disclosing Design Rationale. In *Proceedings of the SIGPrag 2013 workshop at ICIS 2013* Milan. Available at: http://media.sigprag.net/2016/03/SIGPrag-2013-Sjostrom-et-al.pdf.

SJÖSTRÖM J, VON ESSEN L and GRÖNQVIST H (2014) The Origin and Impact of Ideals in eHealth Research: Experiences From the U-CARE Research Environment. *JMIR Research Protocols* **3(2)**, e28. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4051743&tool=pmcentrez&rendertype=abstract (accessed 25/08/14).

SJÖSTRÖM J and GOLDKUHL G (2010) A Stakeholder-centric Approach to Information Systems Design. In *eChallenges* (CUNNINGHAM P & CUNNINGHAM M, Eds), pp 1–8, International Information Management Corporation, Warsaw.

SJÖSTRÖM J and HERMELIN M (2013) In-Place Translation in Information Systems Drawing from the Knowledge Base. In *Design Science: Perspectives from Europe: European Design Science Symposium, EDSS 2012, CCIS 388* (HELFERT M & DONNELLAN B, Eds), pp 88–98, Springer International Publishing, Leixlip, Ireland.

SJÖSTRÖM J, KRUSE LC, HAJ-BOLOURI A and FLENSBURG P (2018) Software-Embedded Evaluation Support in Design Science Research. In *Designing for a Digital and Globalized World. DESRIST 2018. Lecture Notes in Computer Science, vol 10844* (CHATTERJEE S, DUTTA K, & SUNDARRAJ R, Eds), pp 348–362, Springer, Cham.

SJÖSTRÖM J, RAHMAN MH, RAFIQ A, LOCHAN R and ÅGERFALK PJ (2013) Respondent Behavior Logging: An Opportunity for Online Survey Design. In *DESRIST 2013 Proceedings, LNCS 7939* pp 511–518. Available at: http://link.springer.com/10.1007/978-3-642-38827-9_44.

SMEDBERG Å and SANDMARK H (2012) Design of a Mobile Phone App Prototype for Reflections on Perceived Stress. In *The Fourth International Conference on eHealth, Telemedicine, and Social Medicine (eTELEMED 2012)* pp 243–248, Valencia, Spain.

SOJER M and HENKEL J (2010) Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments. *Journal of the Association for Information Systems* **11(12)**, 868–901.

SOMMERVILLE I (2011) *Software Engineering* (9th edition). Addison-Wesley.

SONNENBERG C and VOM BROCKE J (2012) Evaluation patterns for design science research artefacts. In *Practical Aspects of Design Science: European Design Science Symposium* (HELFERT M & DONNELLAN B, Eds), pp 71–83, Springer-Verlag Berlin Heidelberg, Leixlip, Ireland.

SØRENSEN C and LANDAU J (2014) We've Got 99 Problems, but a Phone Ain't One: Mobile ICT and Academic Agility in Information Systems Research. In *ECIS 2014 Proceedings* AISeL. Available at: http://aisel.aisnet.org/ecis2014/proceedings/track03/7.

STODDEN V (2010) The Scientific Method in Practice: Reproducibility in the Computational Sciences. *MIT Sloan Research Paper*, 1–33. Available at: http://ssrn.com/abstract=1550193.

STODDEN V, GUO P and MA Z (2013) Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. *PLoS ONE* **8(6)**, 2–9.

STODDEN V, MIGUEZ S and SEILER J (2015) ResearchCompendia.org: Cyberinfrastructure for reproducibility and collaboration in computational science. *Computing in Science and Engineering* **17(1)**, 12–19.

STÖRRLE H, KRISTENSEN K and MADSEN J (2016) Taming your clients, or: Defining and managing requirements in an academic research context. In *Proceedings of first conference of research software engineers* Manchester, UK. Available at: https://ukrse.github.io/conf2016_talks.

TASCHUK M and WILSON G (2017) Ten simple rules for making research software more robust. *PLOS Computational Biology* **13(4)**, e1005412. Available at: https://dx.plos.org/10.1371/journal.pcbi.1005412.

TERNSTRÖM E, HILDINGSSON I, HAINES H, KARLSTRÖM A, SUNDIN Ö, EKDAHL J, SEGEBLAD B, LARSSON B, RONDUNG E and RUBERTSSON C (2017) A randomized controlled study comparing internet-based cognitive behavioral therapy and counselling by standard care for fear of birth – A study protocol. *Sexual & Reproductive Healthcare* **13**, 75–82. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1877575617300356.

TOM E, AURUM A and VIDGEN R (2013) An exploration of technical debt. *Journal of Systems and Software* **86(6)**, 1498–1516. Available at: http://www.sciencedirect.com/science/article/pii/S0164121213000022 (accessed 22/08/14).

TREMBLAY MC, HEVNER AR and BERNDT DJ (2010) Focus Groups for Artifact Refinement and Evaluation in Design Research. *Communications of the AIS* **26(June)**, 599–618.

TRUEX DP, BASKERVILLE RL and KLEIN H (1999) Growing systems in emergent organizations. *Communications of the ACM* **42(8)**, 117–123. Available at: http://dl.acm.org/ft_gateway.cfm?id=310984&type=html (accessed 17/10/14).

VAISHNAVI V and KUECHLER B (2004) *Design Science Research in Information Systems*. [Online] Available at: http://www.desrist.org/design-research-in-information-systems/ (accessed 15/11/15).

VENABLE J (2006) A Framework for Design Science Research Activities. In *Emerging Trends and Challenges in Information Technology Management* pp 184–187, Idea Group Publishing, Washington, DC.

VENABLE J, PRIES-HEJE J and BASKERVILLE R (2015) FEDS2: A Practical Tutorial on the Framework for Evaluation in Design Science Research (v. 2). In *CAiSE 2015 Proceedings (LNCS 9097)* pp 527–528, Springer International Publishing Switzerland, Stockholm, Sweden. Available at: https://link.springer.com/content/pdf/bbm%3A978-3-319-19069-3%2F1.pdf.

VENABLE JR, PRIES-HEJE J and BASKERVILLE R (2016) FEDS: a Framework for Evaluation in Design Science Research. *European Journal of Information Systems* **25(1)**, 77–89. Available at: http://dx.doi.org/10.1057/ejis.2014.36.

VENABLE JR, PRIES-HEJE J and BASKERVILLE RL (2012) A Comprehensive Framework for Evaluation in Design Science Research. In *DESRIST 2012 Proceedings, LNCS 7286* (PEFFERS K, ROTHENBERGER M, & KUECHLER B, Eds), pp 423–438, Springer Verlag, Berlin Heidelberg.

VIDGEN R and BRAA K (1997) Balancing Interpretation and Intervention in Information System Research: The Action Case Approach. In *Information Systems and Qualitative Research* (A.S. L, J. L, & J.I. D, Eds), pp 524–541, Springer US, Boston, MA. Available at: http://link.springer.com/10.1007/978-0-387-35309-8_26.

WAC K (2012) Smartphone as a personal, pervasive health informatics services platform: literature review. *IMIA Yearbook of medical informatics* **7(1)**, 83–93. Available at: http://www.ncbi.nlm.nih.gov/pubmed/22890347 (accessed 02/03/13).

WALLS JG, WIDMEYER GR and EL-SAWY OA (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. Available at: http://iris.nyit.edu/~kkhoo/Spring2008/Topics/DS/DtheoryAssessing_JITTA2004.pdf (accessed 06/03/13).

WALLS JG, WIDMEYER GR and EL-SAWY OA (1992) Building an Information System Design Theory for Vigilant EIS. *Information Systems Research* **3(1)**, 36–59. Available at: http://www.jstor.org/stable/23010780.

WALSH I (2015) Using quantitative data in mixed-design grounded theory studies: An enhanced path to formal grounded theory in information systems. *European Journal of Information Systems* **24(5)**, 531–557.

WALSHAM G (2006) Doing interpretive research. *European Journal of Information Systems* **15(3)**, 320–330. Available at: http://www.palgrave-journals.com/doifinder/10.1057/palgrave.ejis.3000589 (accessed 28/10/12).

WALSHAM G (1995) Interpretive case studies in IS research : nature and method. *European Journal of Information Systems* **4(2)**, 74–81.

WEAVER A, YOUNG AM, ROWNTREE J, TOWNSEND N, PEARSON S, SMITH J, GIBSON O, COBERN W, LARSEN M and TARASSENKO L (2007) Application of mobile phone technology for managing chemotherapy-associated side-effects. *Annals of Oncology* **18(11)**, 1887–1892. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17921245 (accessed 08/12/12).

WEITZNER DJ, ABELSON H, BERNERS-LEE T, FEIGENBAUM J, HENDLER J and SUSSMAN GJ (2008) Information accountability. *Communications of the ACM* **51(6)**, 82–87.

WESTIN S (2014) *Managing data and information quality in construction engineering: a system design approach*. University of Agder: Agder.
WILLIAMS K, CHATTERJEE S and ROSSI M (2008) Design of emerging digital services: a taxonomy. *European Journal of Information Systems* **17(5)**, 505–517. Available at: http://link.springer.com/10.1057/ejis.2008.38.
WOODFORD J, WIKMAN A, CERNVALL M, LJUNGMAN G, ROMPPALA A, GRÖNQVIST H and VON ESSEN L (2018) Study protocol for a feasibility study of an internet-administered, guided, CBT-based, self-help intervention (ENGAGE) for parents of children previously treated for cancer. *BMJ Open* **8(6)**.
XU X, VENKATESH V, TAM KY and HONG S-J (2010) Model of Migration and Use of Platforms: Role of Hierarchy, Current Generation, and Complementarities in Consumer Settings. *Management Science* **56(8)**, 1304–1323.

Part VI: Appendices

Appendix A: Quality Characteristics

A.1 Product Quality Characteristics

Table A.1-1. Product quality characteristics

#	(*Sub*-)Characteristics	Definition
1	Functional suitability	Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions
1.2	*Functional completeness*	Degree to which the set of functions covers all the specified tasks and user objectives
1.3	*Functional correctness*	Degree to which a product or system provides the correct results with the required degree of precision
1.4	*Functional appropriateness*	Degree to which the functions facilitate the accomplishment of specified tasks and objectives
2	Performance efficiency	Performance relative to the number of resources used under specified conditions
2.1	*Time behaviour*	Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements
2.2	*Resource utilisation*	Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements
2.3	*Capacity*	Degree to which the maximum limits of a product or system parameter meet requirements
3	Compatibility	Degree to which a product, system or component can exchange information with other products, systems or components, and perform its required functions while sharing the same hardware or software environment
3.1	*Co-existence*	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product
3.2	*Interoperability*	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged
4	Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified use context
4.1	*Appropriateness recognisability*	Degree to which users can recognise whether a product or system is appropriate for their needs
4.2	*Learnability*	Degree to which a product or system can be used by specified users to achieve specified learning goals to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified use context

4.3	*Operability*	Degree to which a product or system has attributes that make it easy to operate and control
4.4	*User error protection*	Degree to which a system protects users against making errors
4.5	*User interface aesthetics*	Degree to which a user interface enables pleasing and satisfying interaction for the user
4.6	*Accessibility*	Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use
5	Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period
5.1	*Maturity*	Degree to which a system, product or component meets needs for reliability under normal operation
5.2	*Availability*	Degree to which a system, product or component is operational and accessible when required for use
5.3	*Fault tolerance*	Degree to which a system, product or component operates as intended despite the presence of hardware or software faults
5.4	*Recoverability*	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the product or system
6	Security	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation
6.1	*Confidentiality*	Degree to which a product or system ensures that data are accessible only to those authorised to have access
6.2	*Integrity*	Degree to which a system, product or component prevents unauthorised access to, or modification of, computer programs or data
6.3	*Non-repudiation*	Degree to which actions or events can be proven to have taken place so that the actions or events cannot be repudiated later
6.4	*Accountability*	Degree to which the actions of an entity can be traced uniquely to that entity
6.5	*Authenticity*	Degree to which the identity of a subject or resource can be proved to be the one claimed
7	Maintainability	Degree of effectiveness and efficiency with which the intended maintainers can modify a product or system
7.1	*Modularity*	Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
7.2	*Reusability*	Degree to which an asset can be used in more than one system, or in building other assets
7.3	*Analysability*	Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose deficiencies in a product or causes of failures, or to identify parts to be modified
7.4	*Modifiability*	Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product or system quality

7.5	*Testability*	Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met
8	Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another
8.1	*Adaptability*	Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
8.2	*Installability*	Degree of effectiveness and efficiency with which a product or system can be successfully installed and uninstalled in a specified environment
8.3	*Replaceability*	Degree to which a product can replace another specified software product for the same purpose in the same environment

A.2 Quality-in-Use Characteristics

Table A.2-1. Quality-in-use characteristics

#	(*Sub*-)Characteristics	Definition
1	Effectiveness	Accuracy and completeness with which users achieve specified goals
2	Efficiency	Resources expended on the accuracy and completeness with which users achieve goals
3	Satisfaction	Degree to which user needs are satisfied when a product or system is used in a specified use context
3.1	*Usefulness*	Degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use
3.2	*Trust*	Degree to which a user or other stakeholder has confidence that a product or system will behave as intended
3.3	*Pleasure*	Degree to which a user obtains pleasure from fulfilling their personal needs
3.4	*Comfort*	Degree to which the user is satisfied with [feels] physical comfort
4	Freedom from risk	Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment
4.1	*Economic risk mitigation*	Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended use contexts
4.2	*Health and safety risk mitigation*	Degree to which a product or system mitigates the potential risk to people in the intended contexts of use
4.3	*Environmental risk mitigation*	Degree to which a product or system mitigates the potential risk to property or the environment in the intended use contexts
5	Context coverage	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction

		in both specified use contexts and contexts beyond those initially explicitly identified
5.1	*Context completeness*	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified use contexts
5.2	*Flexibility*	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements

Appendix B: eHealth Research Context

B.1 Randomised Controlled Trial

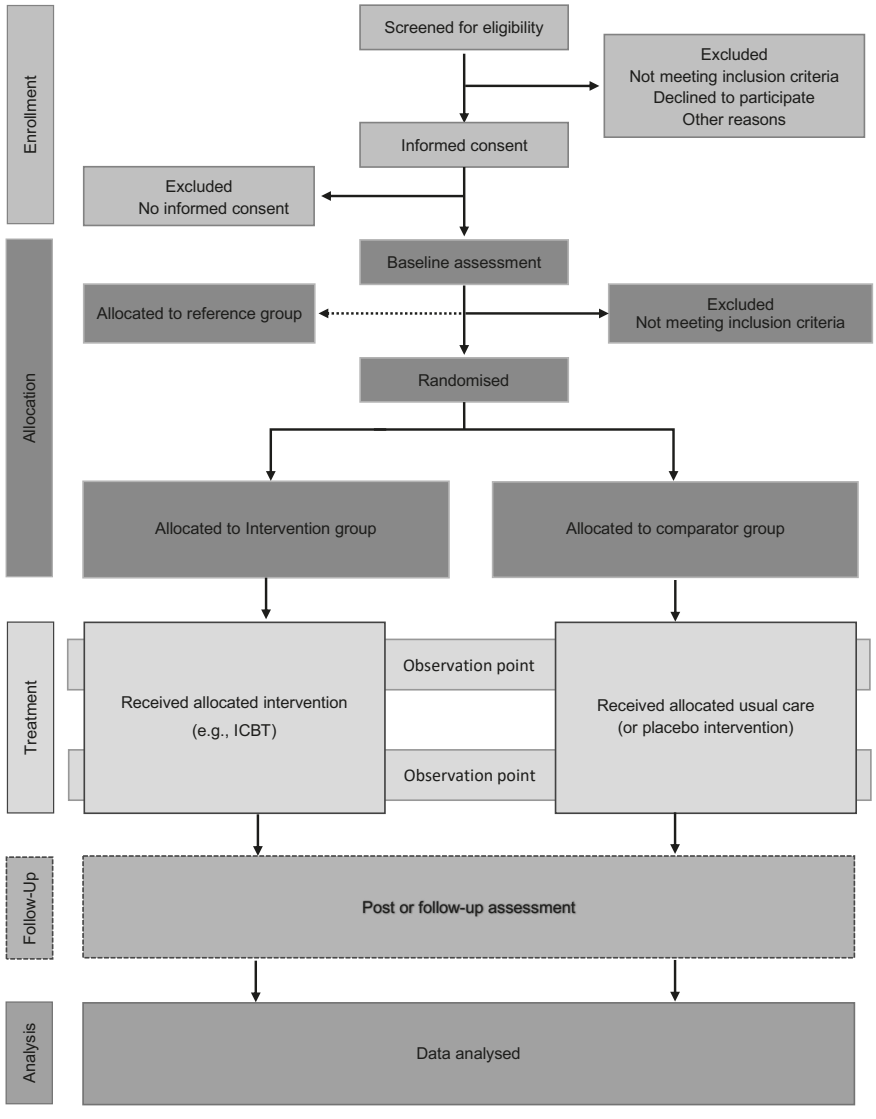


Figure B.1-1.An illustration of an RCT flow diagram

The gold standard for evaluating the effectiveness of a clinical intervention is the randomised controlled trial (RCT). Trial participants are randomly allocated to a group receiving the intervention, or to comparator group(s), enabling for the researcher to compare the groups. Random allocation of participants to each group minimises potentially confounding variables (e.g., systematic differences between participants in the respective groups), maximising the chance that a difference in outcome is due to the intervention, as opposed to other factors (Bhide *et al.*, 2018). Before conducting an RCT, a study protocol is developed. The study protocol describes the background, rationale, objectives, design, methodology, details of the treatment, details about how, when and what information (data) will be collected, and statistical considerations. Examples from the U-CARE setting are Mattsson *et al.* (2013), Norlund *et al.* (2015), Ander *et al.* (2017), and Woodford *et al.* (2018). The study protocol requires approval from an ethics authority and registration in a trial database for clinical studies before commencing the trial.

According to best practices, a pilot or feasibility study is conducted before the full trial to test the feasibility and acceptability of the treatment as well as the planned study procedures (Craig *et al.*, 2008). Any changes in the study design after the pilot/feasibility study need to be approved by the ethics authority.

As illustrated in Figure B.1-1, an RCT can be divided into several phases. In the enrolment phase, potential participants are identified based on inclusion and exclusion criteria pre-defined in the study protocol. Eligible and interested participants are asked to provide informed consent. In the allocation phase, the participants are asked to perform a baseline assessment. The assessment often consists of a number of self-report questionnaires where participants are asked to provide data regarding their health as well as background demographics. After the assessment, participants are randomised either into a group receiving treatment (e.g., ICBT) or to a comparator group (for example, receiving standard treatment or placebo treatment). Random allocation and allocation concealment (meaning study personnel and participants do not know whether the next participant will be in the intervention or comparator group) minimises selection bias and allows the researchers to evaluate the treatment effect. Observation Points (OP) refer to data collection at predefined time points during the treatment phase. OP can be relative to study inclusion, randomisation, diagnostic date, and another observation point. The follow-up phase refers to data collection after the treatment is finished. Follow-up data should be collected by study personnel who are blinded to which trial arm (intervention or comparator) participants are allocated to. One way of ensuring blinding is for follow-up data to be collected via self-report online, rather than by study personnel. The baseline assessment is used as a reference point in the analysis phase to evaluate the effect of the treatment.

B.2 Related Artefacts in the U-CARE Ecology

There are a number of somewhat similar projects related to internet-based psychology (based on RCT in CBT interventions) going on in different places, for example, Sweden, the United Kingdom, the Netherlands, Australia and Europe. Table B.2-1. lists a few interesting projects at different stages of implementation, to give an idea of ongoing research, without an attempt to be exhaustive.

Table B.2-1. Internet-based psychology projects

Name	Description	Geographic area
U-CARE	http://www.u-care.uu.se	Sweden
Internet Psykiatri	http://web.internetpsykiatri.se	Sweden
KBT Online	http://www.kbtonline.se	Sweden
Iterapi	http://iterapi.se	Sweden
Beating the Blues	http://www.beatingtheblues.co.uk	The United Kingdom
Mood Cafe	http://www.moodcafe.co.uk	The United Kingdom
E-COMPARED	European Comparative Effectiveness Research on internet-based Depression Treatment http://www.e-compared.eu/	The Netherlands
MoodGYM	https://moodgym.anu.edu.au/welcome	Australia
ThisWayUp	https://thiswayup.org.au/	Australia
Mental Health Online	https://www.mentalhealthonline.org.au/	Australia
Beating the Blues	http://www.beatingthebluesus.com	Australia
MasterMind	The MasterMind Consortium consists of partners from Denmark, Scotland, Wales, the Netherlands, Germany, Estonia, Belgium, Spain, Italy, Turkey, Norway, and Greenland http://mastermind-project.eu)	Europe

Table B.2-2 lists a few interesting CTMS projects, to give an idea of ongoing research, without an attempt to be exhaustive.

Table B.2-2. CTMS in the U-CARE ecology

Name	Description	Type
OpenClinica	https://www.openclinica.com/	Free – Open Source
REDCap	http://project-redcap.org	Free* – Close Source
TrialDB	https://trialdb.med.yale.edu/**	Free – Open Source*

*For academics only.

** TrialDB (an open-source software for the management of clinical trials) developed at Yale Center for Medical Informatics (YCMI). The documentation and code were freely available to investigators in academia. The system and code are not available anymore as they have been decommissioned. It is important to note that the user/developer documentation regarding the system was received from Prakash Nadkarni, MD (Yale University, prakash.nadkarni@yale.edu) [November 7, 2017] on personal request.

Appendix C: Data Export

C.1 Reflection Design Pattern

Many programming languages including C# provide a built-in facility for reflection. Reflection allows inspection of classes, interfaces, properties, and methods at runtime. It also allows dynamic creation of an instance of a type and invocation of methods. Here is an example in C#:

```
/* Invoking a method without reflection */
Study study = new Study ();
study.listParticipants();

/* Invoking a method with reflection */
Object study = Activator.CreateInstance("complete_classpath.Study");
MethodInfo method = study.GetType().GetMethod("listParticipants");
method.Invoke(study, null);
```

C.2 Authorisation Feature

The authorisation feature (a.k.a., action framework) allows configuring of all actions in the U-CARE software system. The authorisation feature allows the U-CARE software developers to focus on the core task of development controller actions[97] in the source code of the software, while leaving the responsibility for authorisation and logging to the authorisation feature. The authorisation feature requires multiple steps. First, when a developer creates an action in any controller following the MVC design pattern and publishes code in the production environment, the authorisation mechanism at runtime registers the action and controller. This registration is based on a reflection design pattern similar to what is used in the generic data export feature. This automatic and dynamic registration of actions enables flexibility for developers in managing the software and hence making software more malleable. Second, the developers have to configure user roles: who is allowed to access the action and the action's activity (e.g., *research analysis*), the action's type (e.g., *export data*), and whether or not the action requires authorisation. Third, configure if the

[97] Refers to a controller in an MVC design pattern.

action is required to be included in the log and if it is to appear on various software menus. The action metadata allows investigation of the character of a user request.

The authorisation feature enables changing the configuration at runtime without compiling the code. The authorisation feature also logs the actor (user) who is acting, the context (e.g., study), a timestamp, and the entire parameter list in the HTTP request. This log enables accountability and traceability of system events beside other use. The implications of log data are discussed in a few places in the dissertation.

C.3 Custom-made Data Export Applications

This section describes the development of custom-made data export applications. This development was begun mainly due to the diversity of data export requirements. Multiple C# console applications were developed to export data from DBMS using stored procedures, to manipulate data and TO generate results in the required format. There were multiple custom-made applications based on the specifications of individual data export requests. In the case of a new, similar type of data export request, the existing application was cloned and adapted. These custom-made applications were developed, maintained, and executed by a single developer. The source code of applications was neither stored centrally in any version control system (e.g., SVN or Git) nor shared with the rest of the development team. Furthermore, no one other than the specific developer knew how to operate each specific application and export data.

There were multiple studies in the U-CARE software system, each with a different study design (protocol). The custom-made application development was simple, as each request was specific to a single research study and the developer would focus on one study protocol at a time. The clinical researchers had to communicate with one specific developer directly and explain their needs to get the required results. It was also convenient for the developer to export requested data as he/she over time became well versed in the domain knowledge and its representation in the database. This was beneficial for U-CARE management as, over time, fewer resources (person-hours) were required to export data. Over time, many custom-made applications became obsolete, and some applications evolved and became more efficient. Sometimes, customisation was not possible and the clinical researchers themselves would manipulate data if necessary; this case is one example:

> [the clinical researchers in the] UPPS [study] want to start reviewing the collected observation point data from the study, including questionnaires filled in by staff on behalf of the patients. The most pressing needs are the survey ques-

> tionnaires filled in by the registrars for the patients. However, rather than rewrite the extraction program to look for individual quizzes, it would be easier to extract all questionnaires for all observation points, and the clinical researchers could then filter out what they do not want. (Dev-6, 2017, Product backlog)

Also, another problem occurred when data were not entered into the system directly (a kind of missing data):

> The [clinical] researchers in the UPPS filled in paper forms when [research] participants were unable to use the [system] to complete their questionnaires successfully. This is a situation similar to what we had with the U-CARE Heart study. We will need to provide a mechanism for UPPS staff to enter data after the fact for these study participants. (Dev-6, 2017, Product backlog)

Another problem occurred with external services such as EQ5D:

> There may be issues with the storage and observation point updating of EQ5D surveys. Keep in mind that we have to go to a different website for EQ5D, and the results are stored in their tables.
>
> Determine the following scenarios: a) User started EQ5D, but the completed survey date is not recorded with the user item, b) EQ5D was started, but there is no record of values in the database.
>
> Research EQ5D answers to see if there are irregularities: a) EQ5D is marked as being completed, but the answers are not found in the database, b) EQ5D was started, but was not completed anywhere in the observation point.
>
> Keep in mind that when we show the EQ5D results in the researcher view, all the accumulated EQ5D results for each user are shown at once, making it difficult to identify the most recent answer. (Dev-6, 2016, Product backlog)

Evaluation of custom-made applications was indirect, based on the discussion with the development team and feedback from the clinical researchers. The clinical researchers were satisfied with the data export process at that point. There were two key, large U-CARE studies which still had to export data. A custom-made application was used to export a sample extract. This sample extract enabled the clinical researchers to discuss the data and its format. Based on the feedback, the custom-made application was modified. Study-wide full-scale data will be extracted once the study is finished. This will be the most massive data extraction in U-CARE.

Appendix D: Technology Adaptation

D.1 A Comparison Tool for UI Testing

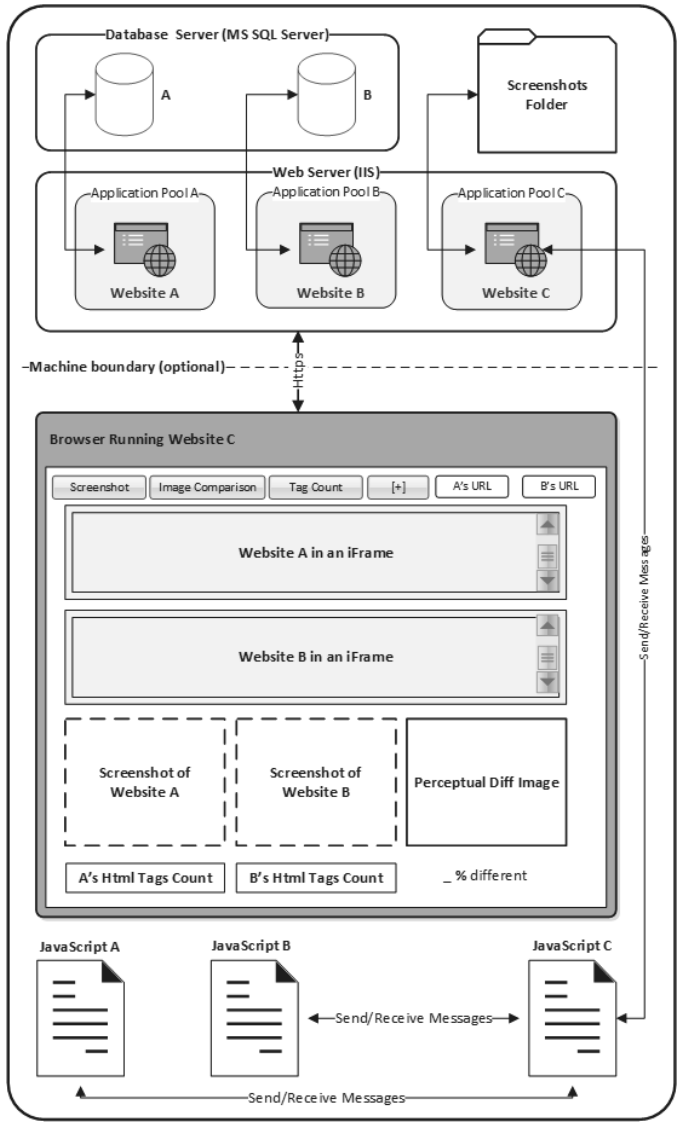


Figure D.1-1. A comparison tool for UI testing.

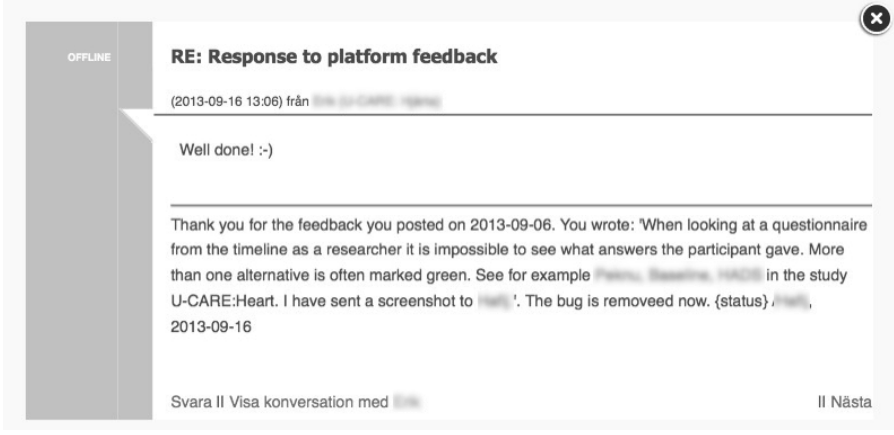


Figure D.1-2. Screenshot of web application ‘A’.

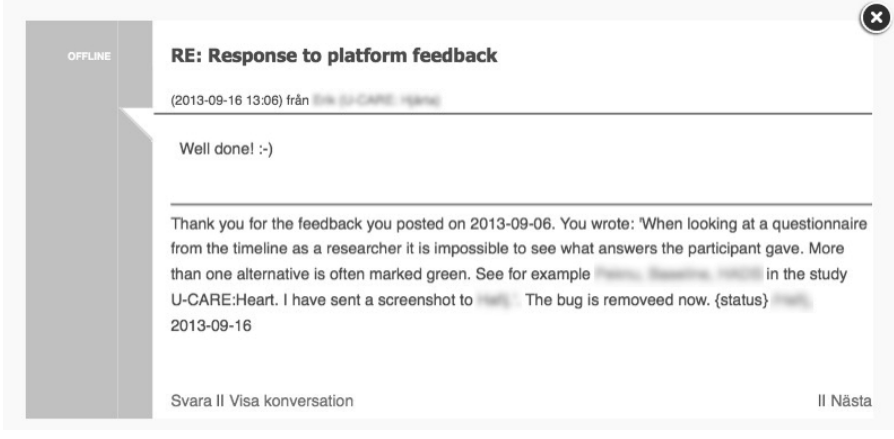


Figure D.1-3. Screenshot of web application ‘B’.

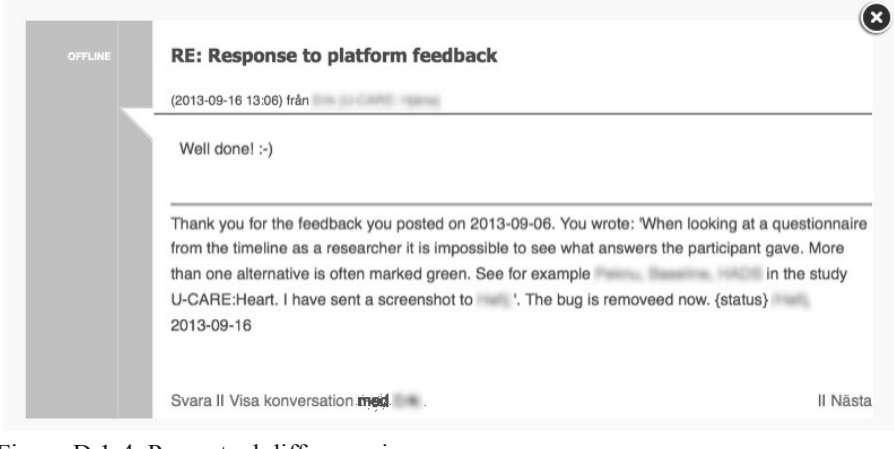


Figure D.1-4. Perceptual difference image.

The comparison tool contains two windows (iFrames). The first window displays the existing system (as web application A) and the second displays refactored system (as web application B). The test server was configured to support this comparison tool. Two web applications, A and B, were deployed with two databases having precisely the same structure and data, one for each application. The third web application was the comparison tool itself. The web applications were isolated by separate application pools on the web server. A JS library was created to communicate between the three applications (see Figure D.1-1).

The tool was improved in many iterations based on the feedback from testing workshops. The comparison tool was redesigned to load a URL (e.g., https://domain/index.cshtml#controller/action) in both windows at once. The scroll position synchronisation feature was added so that both windows showed the same area of interest. The perceptual difference analyser feature was added to discover differences in content, layout and information presented in both windows using image analysis library (resemble.js[98]) developed by Cryer (2013).

The tool took screenshots of current screens of both windows using the html2canvas[99] jQuery plugin and saved the screenshots in the screenshots folder on the web server. Then, the image analysis tool used these two screenshots to create and save a third difference image with altered areas, if any such exist, highlighted in distinct colour. In addition to visualisations, the tool presented the degree of discrepancy quantitatively as a percentage. For example, Figure D.1-2 is 0.13% different from Figure D.1-3 and the discrepancy is highlighted in Figure D.1-4. Another minor feature of the tool was to count and compare the HTML tags. This feature resulted in correcting a few syntax errors in the system. The naming convention of image files, as date_time_type_[url/diff].jpg, helped in error reporting and tracking the testing process. The tool supported the development team in identifying bugs caused by the front-end refactoring, specifically conversion of a large number of views to the Razor view engine standard.

[98] https://github.com/Huddle/Resemble.js [accessed: November, 13, 2013].
[99] https://html2canvas.hertzen.com/ [accessed: November 13, 2013].

D.2 First Developers' Workshop

Table D.2-1. Design decisions for software quality assurance

#	Decision	Rationale	Requirements	Follow-up
1	A quarterly progress report should be produced by the team leader, including progress made (regarding quality assurance) and goals for the next quarter.	Increase the overall transparency of the software development work and increase the orientation towards long-term goals.	The IT coordinator needs to allocate time to produce the report. Parts of the work will be delegated to the development team.	A report provided to U-CARE management every three months.
2	The new testing strategy should be implemented right away.	The rigour of quality assurance should be improved as soon as possible.	Increased support for the *definition of done* in the product backlog feature to ensure compliance with the process.	Implementation of the strategy should be reported at the next two sprint meetings, and followed up in the first quarterly report.
3	The development team should continuously refactor the software to better distribute work between DBMS and application in accordance with principles.	Such refactoring is a *lean* measure to increase the maintainability of the software and to improve performance.	Time is allocated to work with refactoring.	Refactoring progress will be included in the quarterly report.
4	A pedagogical summary will be written to increase the transparency of the software development process.	Our stakeholders need to better understand the measures taken to promote software quality.	The text is phrased and continuously updated on the www.u-care.uu.se web page [for dissemination of information and ongoing activities].	The status of the pedagogical summary will be reported in the quarterly report.
5	The test coverage of core business activities should steadily increase.	Shortcomings in test coverage increase the risk of introducing new bugs when revising the software.	More time must be allocated for testing in the development team.	As defined in the new testing strategy.
6	The testing strategy will be reviewed and refined each quarter.	New experiences and knowledge need to be incorporated into every aspect of our work.	There is a need for a development workshop every quarter.	The latest version of the testing strategy will always be part of the quarterly summary.

7	The business-oriented/end-user-oriented documentation needs to be improved.	More comprehensive documentation will make it easier for all parties to understand the U-CARE business logic and its requirements for development and testing.	There is a need to allocate time for documentation – among both developers and domain experts (e.g., psychologists and researchers).	Advancements in documentation will always be part of the quarterly summary.
8	Technical software documentation needs to be improved.	The existing documentation has proven to be too fragmented to be useful for the development team.	There is a need to allocate more time for documentation in the development team.	A documentation repository needs to be available so that relevant staff can always access the latest version of documents.
9	Software developers in U-CARE work half-time with proactive work, i.e., testing, refactoring and documentation.	The current approach/development process is not sustainable and constitutes a risk to the U-CARE operations.	A decision in the U-CARE management group that development resources need to be oriented more toward quality assurance.	The proactive work will be reported in various ways (see the rest of the table).

D.3 Second Developers' Workshop

Table D.3-1. Follow-up summary of progress on design decisions

#	Decisions of the first workshop	Comment
1	The team leader should produce a quarterly progress report, including progress made (e.g., regarding quality assurance) and goals for the next quarter.	This is the first issue of the quarterly report.
2	The new testing strategy should be implemented right away.	The new testing strategy has been partially implemented, as follows: i) Workflow improvements (the sprint planning and reporting now has a built-in annotation for testing for each task in the backlog); ii) Implementation of continuous integration (an automated routine to validate the quality of existing and new code have been set up). The automation means that all tests are executed at least once a day. Rigorous quality control requires a testing framework that ensures that any changes to the current system do not introduce unintended errors); iii) Infrastructure improvements (testing relies on a functional testing infrastructure which was improved to facilitate the design and construction of advanced tests of the U-CARE software system).
3	The development team should continuously refactor the software to better distribute work between	During the period Nov 2014–Jan 2015, [the development team] has emphasised testing, bug fixing and implementation of the decisions from the previous workshop. Given the increased focus on

	DBMS and application in accordance with principles.	quality assurance, the development of new features was conducted at a slower pace than before.
4	A pedagogical summary will be written to increase the transparency of the software development process.	It is not started yet.
5	The test coverage of core business activities should steadily increase.	The developers have continuously added tests of core business activities since the previous workshop. The test coverage document needs to be updated to reflect the changes made.
6	The testing strategy will be reviewed and refined each quarter.	This is not done, given the short period between the workshops and the first issue of the report. The revisions of the testing strategy should be on the agenda at the next workshop.
7	The business-oriented/end-user-oriented documentation needs to be improved.	A draft document has been created. ISR-1 is currently responsible for the editing of the document. The document should be delivered at the next workshop.
8	Technical software documentation needs to be improved.	No actions are taken yet.
9	Software developers in U-CARE work half-time with quality assurance work, i.e., testing, refactoring and documentation.	Effective since the November 2014 workshop.

D.4 Third Developers' Workshop

Table D.4-1. Follow-up summary of progress on quality assurance goals

#	**Quality assurance goals set in the second workshop**	**Comment**
1	Re-visit the decisions made at the first developers' workshop to ensure that the development process fully complies with the new testing strategy.	All completed items on the backlog for which a test is appropriate now include the date of the test with a brief description.
2	Implement routines to ensure a bi-weekly update of the test coverage document, so that there is always a test coverage snapshot available.	Work on the test document to describe the core business processes is ongoing.
3	The test coverage document should be updated each time new tests are committed to the code repository.	See comment for 2.
4	Finish version 1 of the end-user documentation (for staff) and disseminate it to the U-CARE [system] users.	This document will be completed by the end of quarter 4, 2015.
5	Finish pedagogical summary of the software development process and publish it on the U-CARE web page	Completed.

6	Identify and document specific needs for technical documentation of the U-CARE [system].	Postponed to quarter 4.
7	Prioritise refactoring needs.	The analysis of our existing code base, performed as we develop the mobile application interface, is the starting point of a process of identifying areas which would be appropriate for code refactoring.
8	Organise an additional developer workshop to evaluate and further improve the software development process.	The developer workshop was held on May 13, 2015, with a focus on formal procedures for test coverage, as well as quality assurance issues.

D.5 jQuery Upgrade

The U-CARE software system has many intuitive features to let the user experience a rich graphical interface with nicely presented menus, sliders, tooltips, sortable data tables, forms, popups, a calendar, an HTML editor, et cetera. These features were developed using jQuery.1.4 plugins. However, the jQuery plugins went out of date during the time that the development team continued to develop UIs. Moreover, the opportunity to use many third-party plugins became very limited as those third-party plugins were using the upgraded version of jQuery. Therefore, the need for upgrading jQuery to the latest version had become crucial, but at the same time very risky, as such a move could damage the already used components in the U-CARE software system.

In 2012, the development team set up a separate code branch and dedicated one software developer to take the initiative on upgrading the existing jQuery version from 1.4 to 1.9. The point of having a separate code branch was to test and solve all the bugs which might arise from the transformation of the jQuery plugin. However, there was no test strategy beyond stumbling over bugs and solving accordingly. The development team succeeded in solving most of the problems, but was not ready to publish the changes to the production server. Since there was pressure to work on new features that required the latest jQuery plugin, the development team had to publish insufficiently tested code to the production server. Thus, the development team faced many unintended bugs reported by the clinical researchers. However, in the end, the development team managed to clear up all the bugs within a month or two.

Similarly, during the mobile adaptation (case III), Bootstrap UI library implementation led to a change of jQuery plugins. Also, during the mobile adaptation, jQuery was upgraded, and as a result the development team once again had to upgrade jQuery plugins. For example, for a responsive table, a new plugin was required, but the new plugin required an upgrade in jQuery. The lesson learned from this situation was that the technology upgrade was a circular loop, an endless reciprocal cycle, which required careful planning.

Appendix E: Adaptation to Mobile Devices

E.1 Mobile Adaptation Choices

Table E.1-1. Mobile adaptation choices

Type	Description
Native App	Native apps are platform-specific, and in most cases, the developer has to create versions of their apps for multiple platforms. Android (Google), Blackberry (RIM), iOS (Apple), and Windows Phone (Microsoft) are some of the major app platforms. Apps may use hardware features like multi-touch, physical location identification, video and still images from the camera, along with audio and other capabilities.
Mobile Web App	The mobile web is the World Wide Web, which is accessed through a mobile device. The web app is a website that, in many ways, looks and feels like a native application. A browser runs it and it is typically written in HTML5.
Hybrid App	Hybrid apps are part native apps, part web apps. Like native apps, they can be installed from an app store and can take advantage of the many device features available. Hybrid apps are usually developed using cross-platform application frameworks (wrappers) like Appcelerator Titanium, PhoneGap, Sencha Touch, et cetera. These mobile apps offer cross-platform compatibility and can access the phone's hardware (e.g., camera, GPS, user contacts).
UI Framework	UI-JS mobile framework, for example, Kendo UI, jQuery Mobile, and Intel App Framework bring a native look and feel to Mobile Web Apps. The basic concept behind these frameworks is *write less, do more*.
Mobile Website	A mobile website which has been specifically designed for mobile viewing. Designing a mobile-specific website provides more freedom in the design, content, and structure of a portable webpage
Separate Mobile Theme	The separate mobile theme can be created geared specifically for mobile devices. This theme is additional to the one which is used for a regular website and is used when the user accesses the site from a mobile browser. The theme's stylesheet simplifies the layout and optimises the website experience for the small screen display. This is a quick way to develop a mobile presence without having to develop a separate website.
Custom Framework	Last, but not the least: developing our custom framework for converting an existing system to mobile devices.

E.2 Proof-of-Concept Prototype UI Design

Here are some screenshots of the proof-of-concept prototype's UI design.

Homework on Mobile Device (Part A)

Figure E.2-1. Proof-of-concept prototype – homework on mobile device (part a).

Figure E.2-2. Proof-of-concept prototype – homework on mobile device (part b).

Questionnaire on Tablet

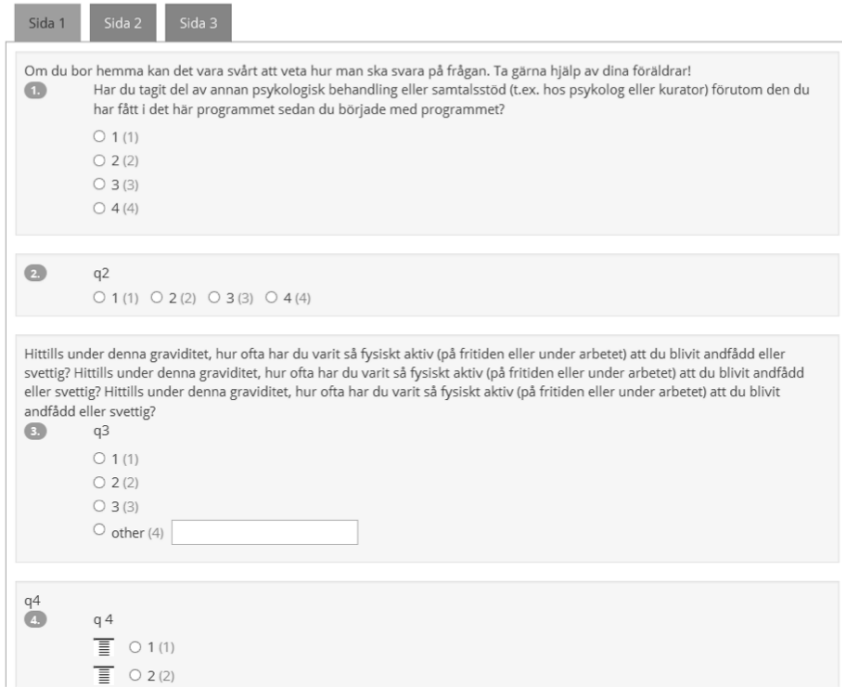
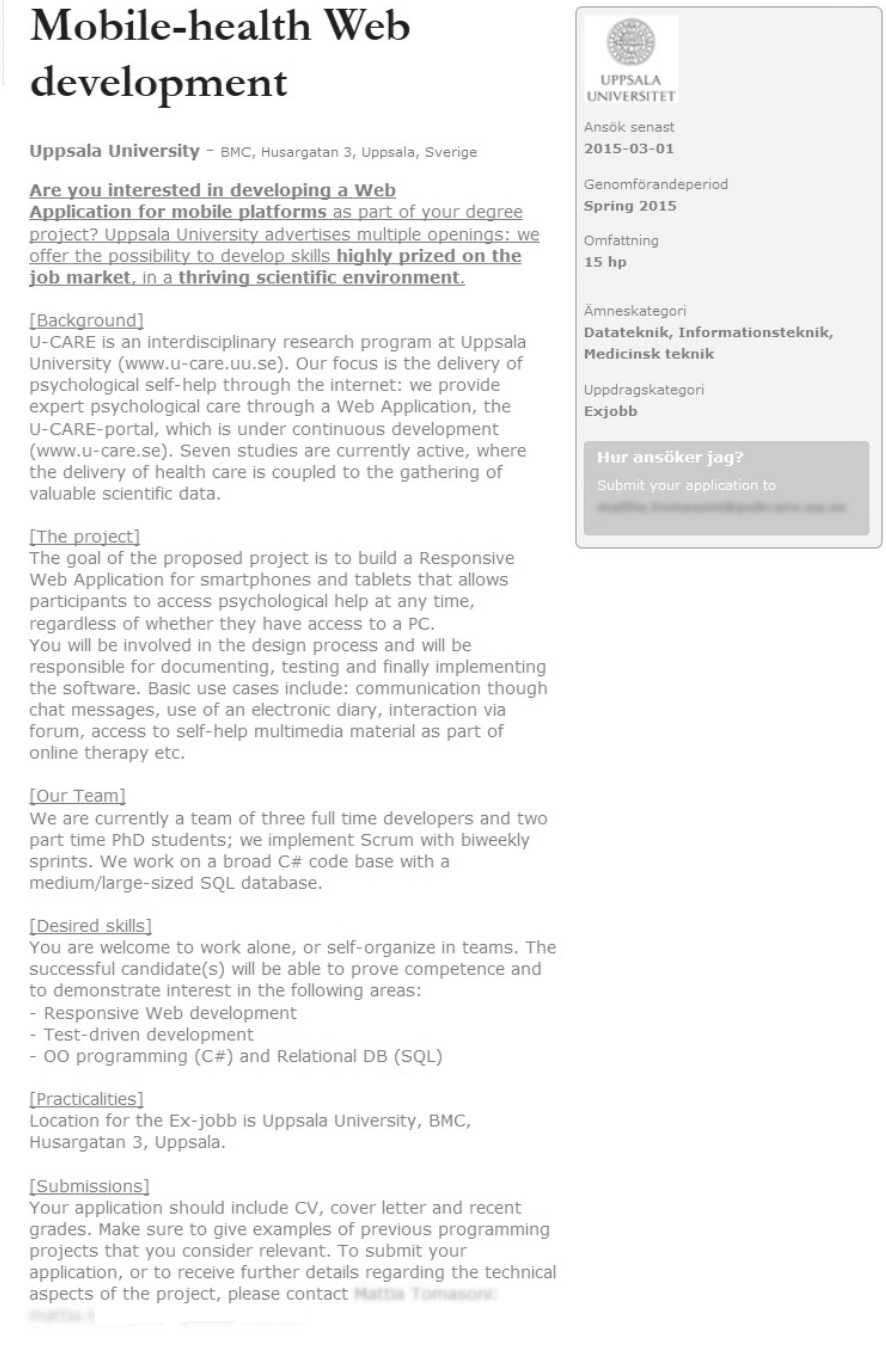


Figure E.2-3. Proof-of-concept prototype – questionnaire on tablet.

E.3 Advertisement for Mobile App Developer



Mobile-health Web development

Uppsala University – BMC, Husargatan 3, Uppsala, Sverige

Are you interested in developing a Web Application for mobile platforms as part of your degree project? Uppsala University advertises multiple openings: we offer the possibility to develop skills **highly prized on the job market**, in a **thriving scientific environment**.

[Background]
U-CARE is an interdisciplinary research program at Uppsala University (www.u-care.uu.se). Our focus is the delivery of psychological self-help through the internet: we provide expert psychological care through a Web Application, the U-CARE-portal, which is under continuous development (www.u-care.se). Seven studies are currently active, where the delivery of health care is coupled to the gathering of valuable scientific data.

[The project]
The goal of the proposed project is to build a Responsive Web Application for smartphones and tablets that allows participants to access psychological help at any time, regardless of whether they have access to a PC.
You will be involved in the design process and will be responsible for documenting, testing and finally implementing the software. Basic use cases include: communication though chat messages, use of an electronic diary, interaction via forum, access to self-help multimedia material as part of online therapy etc.

[Our Team]
We are currently a team of three full time developers and two part time PhD students; we implement Scrum with biweekly sprints. We work on a broad C# code base with a medium/large-sized SQL database.

[Desired skills]
You are welcome to work alone, or self-organize in teams. The successful candidate(s) will be able to prove competence and to demonstrate interest in the following areas:
- Responsive Web development
- Test-driven development
- OO programming (C#) and Relational DB (SQL)

[Practicalities]
Location for the Ex-jobb is Uppsala University, BMC, Husargatan 3, Uppsala.

[Submissions]
Your application should include CV, cover letter and recent grades. Make sure to give examples of previous programming projects that you consider relevant. To submit your application, or to receive further details regarding the technical aspects of the project, please contact Mattia Tomasoni:

UPPSALA UNIVERSITET

Ansök senast
2015-03-01

Genomförandeperiod
Spring 2015

Omfattning
15 hp

Ämneskategori
Datateknik, Informationsteknik, Medicinsk teknik

Uppdragskategori
Exjobb

Hur ansöker jag?
Submit your application to

Figure E.3-1. Advertisement for mobile app developer.

E.4 Mobile Adaptation – Design Workshop I

Table E.4-1. Design workshop I – task list and feedbacks

No	Task	Scenario	Feedback
1	Navigate on the homepage	Log in to the mobile version of the U-CARE portal via your mobile or tablet device and try to use the navigation panel to navigate through the portal.	Top buttons are too small. Padding needed on right side of intro text. Remove search icon from footer if study is not using it. Remove university logo and calendar from footer and move to top menu. Remove profile icon and move to top menu. Add home, forum, and library icons. Move top sandwich menu to the footer. Add confirm dialogue for log out button. Make footer icon informative. Interactive footer icons for IM, chat, and forum (display number of message unread, number of people online, and new posts respectively).
2	Try to see if you can chat with anyone	You want to chat with a peer. Locate the chat option and try to see if you can chat with a peer if anyone is online.	Change colours to green Make it clearer when moving between private vs. public chat You cannot see that there is the possibility to read past chat messages Too much grey space at the top Create separate page If you are online [your] name should be visible in the list of online users
3	Make a post on the forum	You want to post a message on the forum. Locate the forum icon, write your message, and then post it online.	Info button was not displayed Remove HTML tags in the post Too much space above, below, and to the right of posts Make *Posts* clickable and bigger Remove *Navigate forum*
4	Send an IM to User X	You want to send an IM. Locate the IM option. Click on the IM option, add the person to whom you want to send your IM, write your message and send it.	Send/Sent/Archive button should fill the whole page, and if one button is clicked on, the menus should disappear, and only the corresponding IM table should be shown Correct spacing between the columns Remove the pop-up dialogue when writing a new message On Android devices, after sending an IM, the footer buttons change colour You cannot see when you enter text to send a new message
5	Ask an expert a question	You want to ask an expert a question (not via instant messaging or forum). Locate this option, write down your question and send it.	Remove pop-up dialogue After asking a question, the user needs to be redirected to home page

E.5 Mobile Adaptation – Design Workshop II

Table E.5-1. Design workshop II – task list and feedback

No	Task	Scenario	Feedback
1	Fill in question-naires	Log in to the mobile version of the U-CARE website via your mobile or tablet device and try to fill in all the questionnaires (there are 5 of them).	Tables look ugly, and table labels are too wide (see Figure E.5-1) Can we use a legend like in another questionnaire, BADS-SF (see Figure E.5-2)? Questions options too close and small Need to see options when answering the questions EQ5D questionnaire has no mobile version Delay when filling in questions too long Need to research finding the mobile version of HADS Font size is a bit bigger Keep the text within screen in case of zooming
2	Report homework	You want to report on your homework. Locate your treat-ment and report at least one thing on each homework report sheet.	Homework that contains a big table: when I zoom in the footer grows too large Colour thing is bad Zooming required when you write Footer jumps up after a few questions, make it stick Text is a little bit small Report homework has various layout problems, for example, see Figure E.5-3, Figure E.5-4, and Figure E.5-5. The clinical researchers suggested three solutions to these problems a) Psychologist should change the report card, b) Turn matrix into sequential list of (scrollable) input fields, and c) Put questions on top of text field

Design Workshop II – Table Label Too Wide

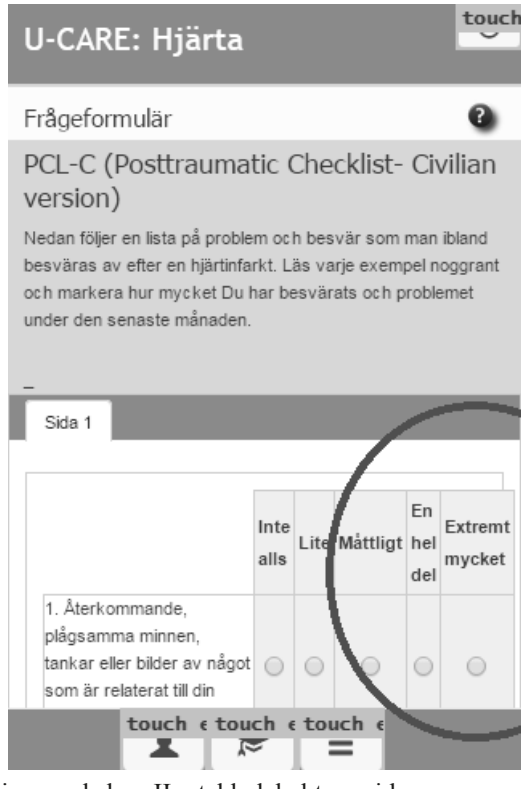


Figure E.5-1. Design workshop II – table label too wide.

Design Workshop II – Suggestion for Table Labels

Figure E.5-2. Design workshop II – suggestion for table labels.

Design Workshop II – Homework Layout Problem (a)

Figure E.5-3. Design workshop II – homework layout problem (a).

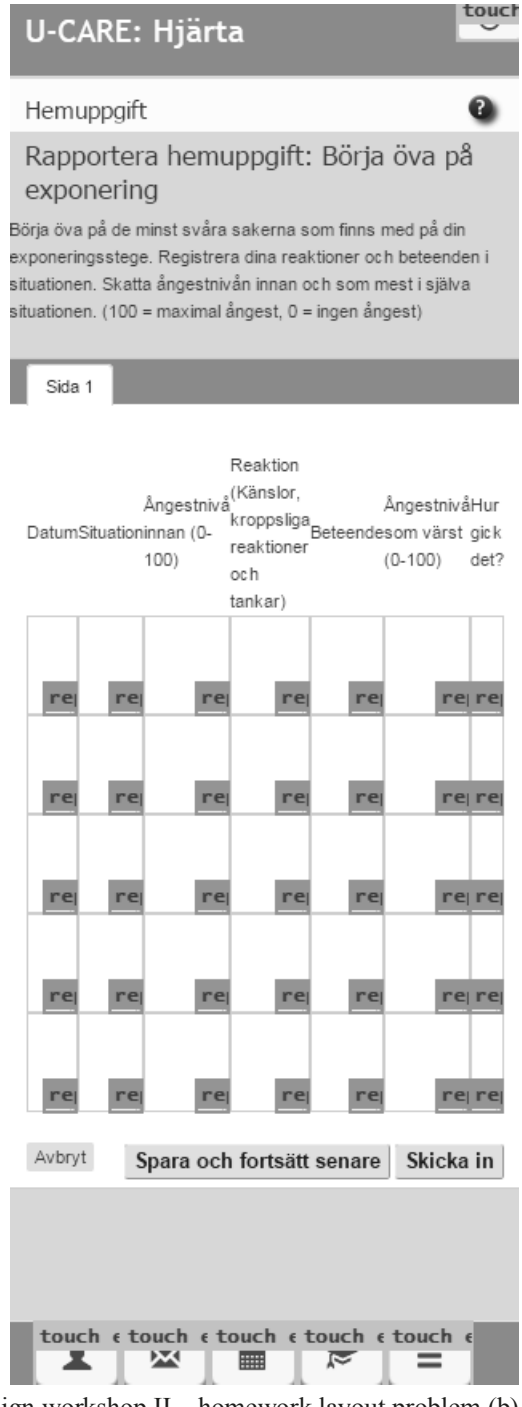


Figure E.5-4. Design workshop II – homework layout problem (b).

Figure E.5-5. Design workshop II – homework layout problem (c).

E.6 Mobile Adaptation – Design Workshop III

Table E.6-1. Design workshop III – task list and feedback

No	Task	Scenario	Feedback
1	Visit library	Try to navigate in the library slide menu	Library slider menu redesign options: 1) Replace the current library slider with a slider plugin that works well on mobile (this will affect all themes). a) Look for a slider where the selected section is strongly highlighted, and all other sections are greyed out. It is currently difficult to understand which library section you are in; this needs to be obvious to the user. b) Consider slider with left and right arrows to get to the previous and following elements (might be easier to use on mobile). 2) If the slider does not give a satisfactory experience on mobile, replace it with a drop-up menu containing all the library headings that would have been available in the slider. This menu would [pop up] from the footer when the library icon is pressed. In the library, there is often extra grey space to the right: remove it. Once a library section has been selected: don't show any text, only wiki-style headings. Do this recursively for sub-headings: a) Remove all tables containing heading links (turn them into wiki-style headings) b) Remove all tabs (turn them into wiki-style headings). Pictures are visually compelling: add the library heading symbol (i.e., the book icon) next to the page header with the name of the selected section.
2	Navigate	Navigate through header and footer menu while visiting library	Remove the menu items from the main menu already shown in the footer icons menu. Notifications on footer IM icon: shown as a little red square with unread IM counter. Footer chat icon: add colour coding and unread counter as a) no other users are online in the chat – grey chat button (just like all others), b) if other users are visible in the chat – green button, c) if other users are actively chatting in the common room – blue button, d) if other users are actively chatting with the current user – blue button and red notification icon with number of unread chat messages. The size of characters in the headers of the chat is not consistent: sub-headers should have smaller char size than headers, (recursively for sub-sub-headers and so on).

E.7 Mobile Adaptation – Design Workshop IV

Table E.7-1. Design workshop IV – task list and feedback

No	Task	Scenario	Feedback
1	Fill in questionnaires	Log in to the mobile version of U-CARE website via your mobile or tablet device and try to fill in all questionnaires (there are 5 of them).	When clicking on option in questionnaire, loading is sometimes too slow. When the question spans more than one line, the second line should be indented below the number (screenshot A).
2	Navigate on the homepage	Try to use navigation panel to navigate on the website.	Remove chat icon if study not using it or replace it with other option, for example, replace chat icon with "questions and answers"; create an icon for it with a question mark and the text "Question" below it. On some Android devices, the bottom nav bar gets covered by the browser bar: this is 'very confusing,' it might take a while before the user realises that the bar is even there.
3	Try to see if you can chat with anyone	You want to chat with a peer. Locate the chat option and try to see if you can chat with a peer if anyone is online.	Chat pop-up (on iPhone 4): cannot agree to "Regular" pop-up. Even so, the user is redirected to the chat page and allowed to use it. Though the pop-up works, it is possible to move on without accepting.
4	Visit library	Try to navigate in the library slide menu.	Library slide menu (carousel) a) Current carousel does not work well (at least on iPhone 4/5/6). It takes about half a second of pressing on an icon before the click is registered and the corresponding library section is shown. b) Greying out of library icons is confusing, use something else instead (for example a frame around the selected item). Also: put selected item in the middle. c) Remove yellow tooltip on library icons (on mobile only) (screenshot B). Wiki-style sections a) The whole tab should be clickable. b) Add up/down arrows (open/close sections). Pagination does not work Cannot see last few lines (CC licence text) at the bottom of the page. Individual library items (inside wiki-style sections) are not evenly spaced (screenshot C). Fix to padding according to UI Policy doc (screenshot D).
5	Make a post on the forum	You want to post a message on the forum. Locate the forum icon, write your message and then post it online.	The whole subject area box needs to be clickable. Fix to padding according to UI Policy doc. When you choose a thread, post should be clickable as well (not the whole box in this case).

			When you choose a thread, “created by […]” should be smaller (screenshot E). Add back buttons to each page and any other pages in the forum. When writing a post, the space between “write post” and the box should be avoided. Forum thread page: remove space between info and box below it (screenshot F). Footer: add number of new posts since last login (little red number, just like IMs and chat). Forum activity (to be discussed further): one suggestion is to write, next to the subject, the number of new posts that have been written since the last time the user logged in.
6	Send an IM to User X	You want to send an IM. Locate the IM option. Click on the IM option, add the person to whom you want to send your IM, write your message and send it.	“Write a new message:” the “:” is on a new line? iPhone 4 and Android Samsung had trouble sending IMs. Confirmation message that an IM has been sent is not translated. Make the whole box clickable in the menu (not just “inbox”). Read message/read the archived message: open popup in full page and add a back button to return. Clicking on the username from inbox brings you to the user’s profile page: fix padding. Selecting usernames in IM. If a correct username is entered, but no option is selected, the IM is not sent. This is confusing: either send the IM or display an error message. When answering an IM, the old message should not be displayed (contains strange HTML) (screenshot H).
7	Ask an expert a question	You want to ask an expert a question (not via instant messaging or forum). Locate this option, write your question and send it.	Take away yellowish box at the top (it brings you to the same page). Fix width of text: make it one column (on both phone and tablet), no space to the sides. “Ask a question” button: put it at the top of the page (on top of “questions and answers”). “Ask a question” view: adjust padding (screenshot G). “Ask an expert” and “question and answer —> ask a question” should both redirect to the same full page. “View question and answers”: fix font size. Make questions wiki-style.
8	Other issues		Enforce pop-up policy everywhere. Make background grey in all info boxes. JS problems on HTC One X with Android 4.2.2 (screenshot I).

Screenshot A

Figure E.7-1. Design workshop IV – screenshot a.

Screenshot B

Figure E.7-2. Design workshop IV – screenshot b.

Screenshot C

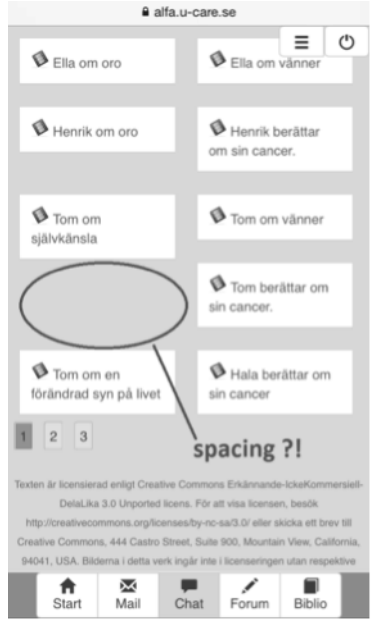


Figure E.7-3. Design workshop IV – screenshot c.

Screenshot D

Figure E.7-4. Design workshop IV – screenshot d.

Screenshot E

Figure E.7-5. Design workshop IV – screenshot e.

Screenshot F

Figure E.7-6. Design workshop IV – screenshot f.

Screenshot G

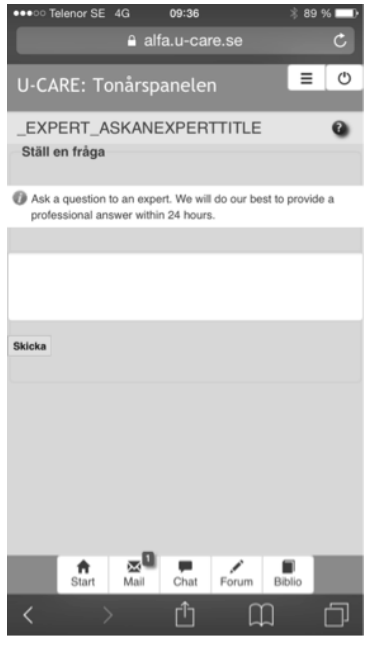


Figure E.7-7. Design workshop IV – screenshot g.

Screenshot H

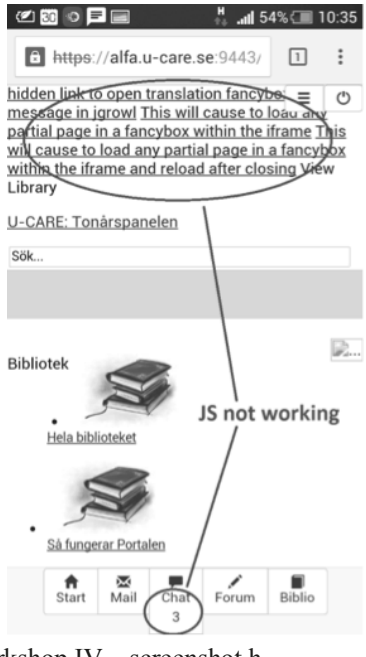


Figure E.7-8. Design workshop IV – screenshot h.

Screenshot I

Figure E.7-9. Design workshop IV – screenshot i.

E.8 Mobile Adaptation – Iteration V Feedback

Table E.8-1. Iteration V feedback

No	Sections	Feedback
1	Navigation	Top menu: missing last horizontal white line. iPad: first button (profile) too much left padding. U-CARE AdultCan Study: Shortcut to the library -> library items (in boxes). The separator line before "See all content" looks like an "i." It is not clear. Need to update credentials on the first login.
2	Chat	Need to make it full page. The pop-up does not extend vertically to the bottom of the page (Samsung). "Visa inte igen" adds space between checkbox and text. The left padding is wrong. Scrolling horizontally with the pop-up should not be allowed: the text disappears! If you do not click on "godkänn," you still have access to chat for a few secs, then you are redirected to start page. Suggestion: JSON error in regular pop-up is asking the user to click on "godkänn." Cannot log out from chat page.

		“Öppen chatt” title: Samsung/iPhone: not displayed at all! Don't know if I am in public or private; if in private I don't know with whom I am chatting. iPad: too much left padding. When the chat space is filled with messages, there is no left bar to indicate that you can scroll. When starting to chat, a date is displayed at the top: use format YYYY-MM-DD. When the user receives a private chat, a little bubble appears next to his/her name in the “inloggade” list. This is not visible enough since on mobile the list is not visible all the time! “Gå till den öppna chatten/Inloggade deltagare”: font size too big. Chat: Log in as user x, go to chat, make visible, click on “x”: opens private chat with self. This should not be allowed. Emoticons do not work: they come up as “?”. On iPad, the whole page “periodically” blinks (every five secs or so). Sometimes the sent chat messages are displayed twice, and sometimes whole chunks of conversations are displayed twice. When accessing chat, by default visible true; still, you are not visible to others. User Y on iPhone 6 wrote to [user] Z in private, but text was shown in open chat!
3	Library	Padding: Below slider, fix left padding of title, text and wiki-style sections. “Hela biblioteket” has different padding than the other sections Too much space between slider dots and sections title. It is easy to click on the slider element when trying to click on the arrow. Suggestion 1: move arrows below the slider, position dynamically based on the position of the dots. Suggestion 2: otherwise just remove them. vuxna -> “Föreläsningar om cancerdiag”-> intro section -> introduction: Too much empty vertical space inside the wiki-style section (other wiki-style sections in the introduction have a little space in the text). In the popup that appears: left padding of the title is off with respect to the text. Size of text in pop-up is too big compared with text outside and title. In the popup that appears, “Skriv ut”, does not work. Slider: (Samsung only) If we slide (with finger) to some element not currently displayed, we need to click on the item twice to select it. When there are too few elements the elements, do not appear centred (e.g., Hjärta library on iPad). The text below the slider items: Text overlaps with that of neighbouring elements; needs to be nicely spaced and centred in relation to the element (e.g., vuxna -> “Föreläsningar om cancerdiag”) (iPhone5/6 only). Similarly: text below the item in the slider is not centred in relation to the blue box. Page buttons (e.g., Vuxna library -> “hela biblioteket” -> bottom of the page). The left padding of the page buttons is not the same as the rest of elements on the page.

		When clicking on a page button, we lose the slider selection marker (blue square on selected slider element): the user does not know where s/he is in the library. "Hela biblioteket" (Vuxna) -> (iPhone) when item name is too long, the text wraps on two lines and the padding of the second line is wrong. "Extramaterial till modulerna" (Hjärta) -> "Introduktion:" not enough vertical space between text and wiki-style sections. It is confusing that the content associated with the first slider element is displayed from the beginning! (CR-9 had not understood that clicking on the slider would update the content below!). Suggestion: Show content only after the click. Slider dots: They are not horizontally centred. They only change if you touch the arrows or slide with fingers, not if you touch the slider elements. Wiki-style tabs: change colour, for example, use the background colour of the page title ("Bibliotek"). General: why do we have a "Hela biblioteket"? Does it just contain the same content as the rest of the sections? Duplication of info? PDF: when accessing a PDF, we are redirected to a page without a back/cancel button. The only way to go back is to use the browser's back button, but that redirects us to a different page than the one [where] we started.
4.	PDF	Can't find a way to open PDF, it only downloads. In CR-9's opinion, this is too confusing (on Samsung). PDF in library (hjärta): "Hela biblioteket" -> 3rd button -> "Så går behandlingen till" Network error when opening with Adobe Acrobat Reader.
5	CBT	Read completed homework, question headers: fix left padding when on more than one line. Read feedback: too much space before feedback, also fix left padding. Composite item (Hjärta last item of stage-2 of intro module): fix to the pad of submitted homework. Composite item: centre video and adjusts size. Composite item: does not have an icon. Also "Klar" does not appear in parentheses.
6	Login page	Upon tilting the device, any text that has been entered is removed. Check to see if it applies once logged in. [The login page is special in this respect: for technical reasons (the carousel is visible/hidden depending on the width of the screen) the page is reloaded upon tilting. Does not apply to the rest of the portal.] iPad: padding of radio box; too close to the text. iPhone: text alignment is bad. "Har du problem att logga in" text is too big for all devices. iPhone: in pop-up, too much scrolling, and scrolling is cumbersome (sometimes page moves instead of box content). Eliminate all pop-ups, including the one, that asking for the new password, on the login page. Brute force: text should be: "Har du glömt lösenord" – "Du kan tidigast logga in" should come first.
7	Edit profile	Samsung/iPad: "Ändra Profilbild" text is misaligned. Text to describe what is needed in a password is missing sometimes. Was there the first time. Then upon logging in again, it disappeared.

		Can't change the password without JSON untranslated errors popping up. The user is not informed why the password is invalid (e.g., minimum length, one capital letter required, and one number required). Change profile picture: Picture is flipped at a wrong angle upon uploading. Edit about me: JSON Message for text change is not translated. Edit settings: iPad/Samsung: “Inställningar” text is badly aligned with regard to radio boxes, the semicolon is missing. “Spara ändringar” button too close to everything. No back buttons. Make whole text clickable (not just radio box). Change Mobile and Email: Pop-up no good. Kill all pop-ups in profile/settings. Bad mobile number format gives untranslated JSON error. Access UU Links: There is no pop-up (if there needs to be one). The user should be warned about navigating to another link. Access About us: Names are displayed twice. Bad alignment of items (not evenly aligned). Samsung: Text is squeezed. Bad formatting.
8	IM	Make whole box (cell) selectable instead of just text within it. No possibility to delete sent IMs. “Svara” link in the message does not work. IMs ViewInbox From inbox, clicking on sender’s name works, but there is no back button to navigate back to the inbox. Manually selecting or deselecting specific inbox items does not always work (similar to tap swipe in the library). Clicking on Forum message title takes you to a page without translated text. IMs Send: Write a new message: Padding on bottom text content needs to be aligned with other text boxes. Write a new message: Top text box is of a different type than the other two. When sending a message to someone not on the list, it does not go through, and the user is not notified. Translation incomplete when sending a message successfully. Username (subject) – colon is missing. IMs ViewSent: When clicking on own profile, redirected to the homepage instead of to profile. When clicking on a user with no profile, directed to a page with no translation and missing back button. Missing translation when clicking on email content via subject. TYPE column in sent messages is capitalised and not in Swedish.
9	Calendar	Upon clicking on a day box, pop-up has underscores before the dates. “Skapad” should be “Starta” instead. We saw the year 1970 for test user account XYZ. Translation is missing when expiry date precedes creation date.

		On iPad: input pop-up calendar dates for start and finish does not always work.
10	Diary	AddPrivateEntry: Test User 4 has no access to the logbook. "Rubrik" textbox not aligned. Translation is missing when expiry date precedes creation date.
11	Forum	Regular: upon clicking on the x button, you are still allowed to use forum without being redirected. Samsung: in topic window (not thread) with two threads or more, padding is inconsistent between items, borders also missing. Post page: page numbering not aligned with everything else. Post page: there is a spacing difference between "Skrivet av"/"Citera rapportera" in posts in the thread. Should always be two lines on small devices. When citing someone, you still have square quotes in original message. When starting to write a new message in a thread, it sometimes happens (randomly) that user cannot click on any items above which have links.
11	FAQ	OK
12	General	iPad/iPhone: when writing in a text area (e.g., a new post in a thread, or an IM), the software keyboard which is displayed does not disappear if the user touches other areas of the screen. CBT, composite folder: does not have an icon. Brute force: text should be: "Har du glömt lösenord? – Du kan tidigast logga in" should come first. When sending a message to someone not on the list, it does not go through, and the user is not notified. IMs ViewSent: when clicking on own nickname, the user is redirected to his/her start page instead of the personal profile (clicking on other users' nicknames in the IM pages redirects to their profile). Edit Profile: the bullet points do not look good in the context of the page. Calendar. The RA-1/Dev-4 report "we saw the year 1970 for [test user x]." Investigate. Calendar bug (on phone/iPad/PC): Steps to reproduce: open calendar and try to add a note in a date in the past; in the pop-up that appears, click on the input field for "end date"; a tiny calendar is supposed to appear, but it does not; then close the pop-up and try adding another note (at any date): the "end date" tiny calendar will not appear. Refresh the page and try to add a note for a date in the present/future. The tiny calendar appears as expected. A thin white box appears on the pages: "Write new diary entry" (mobilelegacy [theme]), profile (U2013 [theme]), calendar (mobilelegacy [theme]). It is not in the view; it is created on the fly by one of our underlying frameworks (I suspect Bootstrap).

E.9 Mobile Adaptation – Design Workshop VI

The following was the instruction for the workshop participants:

a. On your phone/tablet, navigate to https://beta.u-care.se:xxxx.
b. Log in: "mr1_[your_name]" (e.g., "mr1_abcd"), password: "xxxxxxx".
c. Carry out each task (note: the description was written before the mobile conversion),
    i. Note down any feedback (related to mobile conversion only!),
    ii. Cross out the task and move to the next one.
d. When done: submit all feedback.

Table E.9-1. Design workshop VI – task list and feedback

No	Task	Scenario	Feedback
1	Reset/forgotten password	Navigate to the login page and click on "Har du problem att logga in?", then "Jag vet inte vad jag har för lösenord": input participant email.	
2	Login	Navigate to the login page, input participant email and password and click on "Logga in."	
3	Update credentials (first login only)	Navigate to the login page, input participant credentials and click on "Logga in." Upon the first login, the participant is redirected to a page where s/he is asked to fill in a form to update his/her username and password.	
4	Answer baseline questionnaires	The participant login for the first time and is presented with a pop-up suggesting completion of the baseline questionnaires. The participant confirms the pop-up, then completes and submits each questionnaire.	(Dev-7): - "Cancel" button has the same effect as "Save and continue later" button. Suggestion remove "Save and continue later." - Save and continue later" is slow. Details: "save and continue took a long time, then pressed the browser's back button, and then came back but now could not save and continue or submit without interrupting. I pressed cancel and then opened it again. My answers were from before were gone, and when I filled in the answers again, I received a message about object instance for each response. Only the cancel button was left. I logged out and logged in again; then the buttons came back."

			- No confirm pop-up for “Save and continue later.” Should we add one? - I pressed cancel due to background issues, but got an error trying to load the page. I had been inactive for a long time. It forced me to log in again. When I got in, some answers had not been saved. - I can press start to leave the questionnaires as well. I do not get anything then. - Questions 11 & 16 on Background issues are indexed. - Background issues: The fields for alcoholic beverages are not in line with each other. - There is a “__” after the intro text in each questionnaire. - It took a very long time to load after submitting the last BADS-SF (BASELINE HEART). (Dev-5): - Questionnaire submission is slow (baseline Vuxna). - Almost all users in Vuxna got stuck on questionnaires (“mr1_user_x”, “Mr1_user_y”, “Mr1_user_z”). - Example 1: “MADRS” remained greyed out for several minutes, and the wheel was not present. When I closed the confirm submission pop-up and navigated back to start, the questionnaire had been submitted. - Example 2: “PCL-C” (last questionnaire) remained greyed out for several minutes. I left the phone, and when I got back, the questionnaire had been submitted. (CR-4): - Questionnaire submission (baseline Vuxna). - The page “hanged” [froze] when I tried to move the marker to an exact place at VAS-skalan. I had to force Chrome to close and log in again. It was very hard/impossible to give an exact answer. The EORTC breast cancer module was slow, but I could complete it.
5	Logout	From any location in the portal login page, the participant clicks on the “Logga ut” button.	
6	Change password	From any location in the portal, click on the “Personliga inställningar” icon, then on the “Byt lösenord”	

		link; the participant fills in the form and submits.	
7	Change profile picture	From any location in the Portal, click on the "Personliga inställningar" icon, then on "Ändra profilbild." Choose a file from the local file system and submit the form.	
8	Edit about me	From any location in the portal, click on the "Personliga inställningar" icon, then on "Om mig". Fill in the text box and click on "Spara."	- (Dev-7) I would like the text to be visible on the profile page. I would like to receive the confirmation on the profile page, instead of in the window. - (RA-2 and CR-3) After submitting, the pop-up does not close automatically - (CR-3) The box where you write is not white. - (CR-3) The message is in English: "JSONMSG_CHANGES_WERE_SUCCESSFUL".
9	Edit settings	From any location in the portal, click on the "Personliga inställningar" icon, then on "Inställningar"; the participant is redirected to a page with options: all combinations of options are valid.	- (CR-3) Works, but the box will not close when you click on "Save Changes." - (CR-3) I chose "Do not show my profile picture," but it appears in "Personal settings."
10	Change mobile and email	From any location in the portal, click on the "Personliga inställningar" icon, then on "Byt telefonnummer eller e-postadress"; fill in email or mobile information and submit.	- (Dev-7 and CR-3) Text on the length of the mobile number does not work: JSonerr_Mobile_NUMBER[…] Could perhaps be made eaiser to understand? - (CR-3) After submitting, the pop-up does not close automatically. - (CR-3) The box where you write is not white.
11	Access UU link	Click on the "Uppsala University" icon.	- (RA-2) Names could be made into links to user profiles.
12	Access 'about us'	From any location in the portal, click on the "Om oss" menu link.	
13	IM view inbox	Click on the "IM" icon, then in the IM menu click on "Inkorg"; the table with incoming IMs is presented. For each IM: clicking on the sender's name will redirect the participant to the sender's profile; clicking on the IM subject will redirect to the IM itself. The links "Välj alla brev på denna sida" and "Avmarkera alla brev" select and deselect all IMs and the	- (CR-2) The IM notification counter (footer icons on mobilelegacy [theme], side icons on U2013 [theme]) is incremented when a new IM is received, but never reset to 0 when the page is reloaded.

		dropdown menu next to them marks them as read or moves them to "Archived" folder.	
14	IM send (recipient cannot be therapist)	Click on the "IM" icon, then in the IM menu click on "Skriv ett nytt meddelande"; the participant is then asked to provide a recipient, a subject and body for the IM. The participant sends the IM by clicking on "Skicka meddelande."	- (Dev-7) I could only send to me and CR-7. I tried to send to RA-2, but it did not work. I received no message that it was not working. However, I did not get any field for RA-2. I think it may be that the others have not completed the baseline. I am left on the same page after I had sent the message.
15	IM view sent	Click on the "IM" icon, then in the IM menu click on "Skickade meddelanden"; the table with Sent IMs is presented. For each IM: clicking on the Recipient's name will redirect the participant to the Recipient's profile; clicking on the IM subject will redirect to the IM itself.	
16	IM view archived	Click on the "IM" icon, then in the IM menu click on "Arkiverade meddelanden"; the table with archived IMs is presented. For each IM: clicking on the sender's name will redirect the participant to the sender's profile; clicking on the IM subject will redirect to the IM itself. The links "Välj alla brev på denna sida" and "Avmarkera alla brev" select and deselect all IMs and the dropdown menu next to them marks them as read or moves them to inbox folder.	- (CR-4) There is test in English "The user does not have any profile."
17	Calendar add note	Click on the "calendar" icon; when the calendar page is presented, click on any of the squares corresponding to a day of the month; a pop-up appears where the participant can input the title, beginning/end date and the message corresponding to the calendar note s/he wants to	- (Dev-7) A bit difficult to enter the dates by hand instead of scrolling.

		add. Submit by clicking on "Lägg till."	
18	Navigate library	Click on the "Library" link/icon, then select one of the items in the carousel: sections, divided into subsections, containing library items.	- (Dev-7) The dots under the carousel are not used. - (Dev-7) I cannot open the PDF files. It says that the network is not available. - (CR-4) I got a message that I had no network connection when I tried to open a PDF in "Krisreaktioner" in the library. - (CR-3) There's something strange going on here. When I select "Interviews with […]" then all subcategories and items from "Extras" are possible to view and select. When I click on arrows, the selection will not move on the icons, and the content will not change either.
19	Ask an expert (FAQ)	Click on the link "Frågor & Svar", then "Visa Frågor & Svar" and finally click on the button "Fråga en expert". The participant fills in the form and submits.	- (Dev-7) Why no heading on the message to the expert?
20	Read FAQs	Click on the link "Frågor & Svar", then "Visa Frågor & Svar" and finally click on one of the question topics.	
21	Diary – add private entry	Click on the link "Loggbok", then "Skriv i loggbok"; the participant fills in the form with the title, message and date of the diary entry. The option "Private" is selected under "Vem får se?".	
22	Diary – add public entry	Click on the link "Loggbok", then "Skriv i loggbok"; the participant fills in the form with the title, message and date of the diary entry. The option "Andra deltagare" is selected under "Vem får se?".	
23	Diary – read own entries	Click on the link "Loggbok", then "Läs loggboken."	
24	Diary – read others entries	click on the link "Loggbok", then "Andras tankar."	
25	Forum navigate	When the participant clicks on the link "Forum", s/he is redirected to a page with a list of Forum topics; a button called "Regler" at the top of the page opens a pop-up with terms and con-	- (Dev-7) I was in forum before I got to this point, and I am not sure if the rules came up automatically the first time. I pressed the "Rules" button and then opened the window to approve the rules. I do not know if I have approved the rules, but I can write in the forum. I did

		ditions: the pop-up automatically opens the first time a participant accesses the Forum and s/he is required to click on "Jag godkänner" in order to continue. Below the "Regler" button, there is a list of Forum topics. Clicking on a topic name opens a page with a list of related threads. There are two links for actions associated with each thread ("Rapportera" and "Prenumerera"). Clicking on a thread name opens a page with a list of related posts. There are two links for actions associated with each post ("Citera" and "Rapportera"). At the bottom of the post page there is an input text field which allows the participant to contribute to a discussion. At the top of the thread and post pages, there is a button to create a new thread/post ("Skapa ny diskussion"/ "Skriv ett inlägg") and links to go back to previous pages in the forum. Each thread and post is accompanied by a picture of the participant who created it: clicking on the pictures redirects to the participant profile.	not get any confirmation that I had approved the rules. I opened the window with rules and pressed accept. I opened it again and could still accept. Suggestion: I think we should take away the "godkänn" button as it does nothing. - (CR-4) "Regler" did not pop up. - (CR-2) When a participant logs in for the first time, s/he should not get any forum notifications ("xxx posts since you last logged in").
26	Forum contribute to thread	Click on the link "Forum" (if needed, accept the conditions in the "Regler" pop-up), click on a topic, click on a thread, click on "Skriv ett inlägg" – the participant is presented with a text input field where the post content can be entered and submitted via the "Skicka" button.	
27	CBT activate new module	If there are less than two active CBT modules, the participant page will include a button called "Aktivera modul"; the participant clicks on it and selects	

		a module from the list that pops up.	
28	CBT view item	The participant clicks on a CBT item (e.g., a PDF file, video/audio content).	- (Dev-7) I cannot open the PDF. - (CR-3) Two PDF files cannot be opened (Basic text1 and 2).
29	CBT complete homework	The participant clicks on a CBT homework, and s/he is redirected to a page containing the homework; the participant can fill in the cells and submit.	- (Dev-7) Cancel does not work as it says. Everything is saved.
30	CBT read feedback	The participant clicks on a CBT homework that s/he has already submitted and on which the responsible psychologist has given feedback.	
31	View completed item	The participant clicks on a CBT homework, and s/he is redirected to a page containing a static version of the homework.	

[Note: the user needs to have completed baseline to proceed – Task "IM view inbox" onward]

E.10 Mobile Adaptation – Design Workshop VII

The instructions, tasks and scenario for this workshop were the same as given for previous workshop VI. Hence, the scenario column is skipped in the table below and tasks are listed only when there was feedback provided for them.

Table E.10-1. Design workshop VII – task list and feedback

No	Task	Feedback
1	Reset/forgotten password	- (RA-1) iPhone 5: the text rows are not aligned for the three options. Very big white box.
2	Login	- (RA-1) I insert the username (and password) and turn the phone from vertical to horizontal or vice versa, the page is reloaded and the inserted information is deleted. - (CR-2) No space between the login button and the box for input of SMS code. This page is not adjusted for mobile in vertical positioning. The page looks ugly on iPhone 6.
3	Answer base questionnaires	- (Dev-7) Questions 11 & 16 in 'Bakgrundsfrågor' are indexed compared with the other. Also, the answer options are not in line with each other. There is an "_" after the intro text of each questionnaire. It took a very long time to load after submitting the last questionnaire (BADS-SF in baseline U-CARE Heart study). - (Dev-5) Questionnaire submission is slow (baseline U-CARE AdultCan study). The spinning wheel is not displayed, it is outside of the screen (it can be seen by turning the phone to landscape). - (CR-4) Questionnaire submission (baseline Vuxna). The page "froze" when I tried to move the marker to an exact place at VAS-skalan. I had

		to force Chrome to close and log in again. It was very hard or impossible to give an exact answer. The EORTC breast cancer module was slow, but I could complete it. - (RA-1) Text is not aligned where there are multiple choice answers on two rows. EQ5D: The window is too big, it doesn't fit on the screen. - (CR-2) LLI, ESSI and ELSS questionnaires are quite difficult/slow to check/tick the boxes/circles.
4	Edit about me	- (RA-1) When saving, the top "About Me" and "Save" button "don't fit" in the pop-up box. When saved, the textbox it is not responsive, it goes beyond the "borders." This goes for the system in general but especially noticeable when you have a smaller screen. For example, I was writing in the box and accidently clicked outside on the greyed area and then the pop-up closes and you lose what you have written. - (CR-3) The message is in English "_JSONMSG_CHANGES_WERE_SUCCESSFUL".
5	Change mobile and email	- (RA-1) Box doesn't fit in pop-up, this applies both before and after saving. - (Dev-7 and CR-3) Warning text about the length of the mobile number is not working "_jsonerr_Mobile_NUMBER[…]." Could perhaps be made easier to understand?
6	IM view inbox	- (RA-1) I had a red indication on the mailbox, but when opening the mail menu, it didn't show. When a notification is shown, I would think it would say "Inbox (1)" for example. - (CR-2) The IM notification counter (footer icon) is incremented when a new IM is received, but never reset to 0 when the page is reloaded.
7	IM send (recipient cannot be therapist)	- (RA-1) When an IM is sent, the page is updated so the text box is emptied, and the green line saying it has been sent is quite high up and not visible if not scrolling (on iPhone 5). A bit confusing at first glance, whether it has been sent or not. - (Dev-7) I could only send to me and CR-7. I tried to send to RA-2, but it did not work. I received no message that it was not working. But I did not get any field for RA-2. I think it may be that the others have not completed the baseline. I'm on the same page after I have sent the message.
8	IM view sent	- (RA-1): Looking at a sent IM it says "skickat till" (after clicking on sent messages) but then it only shows the date and the hour sent, not the recipient. The window when opening a sent IM is not "fixed," it moves around.
9	IM view archived	- (RA-1): I chose to archive the IM and it got archived, but it didn't get removed from the inbox. I tried again with two IMs and it worked. Tried removing them one by one, had more difficulties. Had to go back and forth and then it worked. - (CR-4) There is English "The user doesn't have any profile."
10	Calendar add note	- (Dev-7) A bit difficult to enter the dates by hand instead of scrolling.
11	Navigate library	- (RA-1) Video pop-up is not responsive. When opening a PDF there is no close button, you have to go "back." Doing that, you come to the starting page of the library and don't know where you were or which the next file is. It would be good if one was moved to the place where one left. - (Dev-7) The dots under the carousel are not used. I cannot open the pdf files. It says that the network is not available.

		- (CR-4) I got a message that I had no network connection when I tried to open a pdf in "Krisreaktioner" in the library. - (CR-3) There's something going on strange here. When I select "Interviews with," then all subcategories and items from "Extra material" is possible to view and select. When I click the arrows, the selection will not move across the icons and the content will not change either.
12	Read FAQs	- (RA-1): I immediately get redirected to a topic and not to the list of topics.
13	Diary – add private entry	- (RA-1) The window is not responsive. The "Save" button is not aligned with the textbox. There is something weird with the row at the bottom.
14	Diary – add public entry	- (RA-1) Bullets from bullet points are not visible in the diary once it has been saved.
15	Forum navigate	- (RA-1) When the participant clicks on the link "Forum", s/he is redirected to a page with a list of forum topics; a button called "Regler" at the top of the page opens a pop-up with terms and conditions: The pop-up automatically opens the first time that a participant accesses the forum and s/he is required to click on "I agree" in order to continue. It did not pop up automatically. Both threads and posts are accompanied by a picture of the participant who created it. Clicking on the pictures redirects to the participant profile. It is supposed to work like this. I clicked on my own picture and I was redirected to the starting page. - (CR-4) "Regler" didn't pop up. - (CR-2) When a participant logs in for the first time s/he should not get any forum notifications ("xxx posts since you last logged in").
16	CBT view item	- (RA-1) The starting view of the video is not good. - (Dev-7) I cannot open the pdfs. - (CR-3) Two PDF files cannot be opened (basic text1 and 2).
17	Other	- (RA-1) When clicking on the "?" sign, a small yellow help pop-up appears to the right, it is not visible if you don't swipe the window to the left (at the same time the window becomes unresponsive). When clicking one more time a white box with information pops up and the text is not adjusted for the pop-up!

E.11 Mobile Adaptation – Desktop Adaptation

The desktop adaptation extended the existing mobile theme for research participants to have the same UI on the desktop as well (see Figure E.11-1). This project was assigned to a single developer, and the design process was based on informal discussions with the stakeholders during the adaptation. A few redundant views, which had previously been duplicated for mobile and desktop, were removed. After going live with the desktop adaptation, the system remained remarkably stable, and no errors were reported. One reason was that the changes were very small and mostly global. Another reason was the increased test coverage during the mobile adaptation, which allowed for through testing before going live.

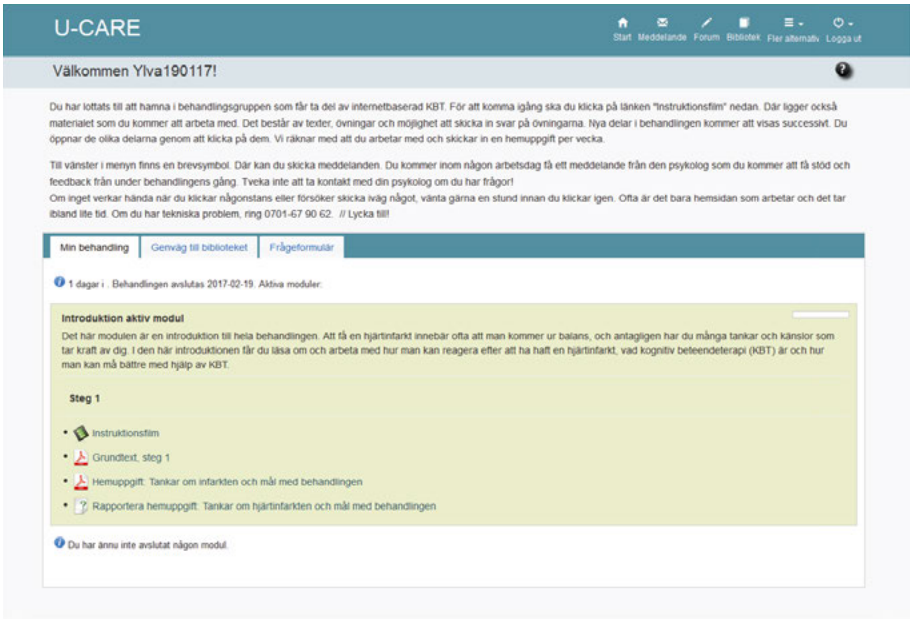


Figure E.11-1. Desktop adaptation for research participant.

E.12 Mobile Adaptation – Study-Specific App

In 2016, the Study PUSSEL (in the U-CARE project ParentsCan) wanted to use the U-CARE Portal to examine attitudes and preferences toward participating in an internet-administrated psychological intervention. For more information about the study, see Woodford *et al.* (2018). However, the study had requirements which were not met by the U-CARE software system. To meet these requirements, the decision was made to design and develop a mobile-adapted stand-alone application for the study, not using the existing U-CARE software system. The required functionality of the app was to first collect online consent from the participants and then show the participants a page with a short film and a questionnaire. At the time, neither online consent nor films on the questionnaire page were designed in the U-CARE software system.

The PUSSEL app was developed by a single developer, while clinical researchers suggested the UI design. The product owner/team leader allowed the developer to spend extra time in learning and designing the app from scratch. It was beneficial for the developer as s/he learned MVC design pattern, C#, CSS, Bootstrap, jQuery, Visual Studio 2015, and translation mechanisms. Although the project took longer than initially planned, the developer learned more about the technology stack used in the U-CARE software system, in the process. It is possible that the functionality could have been achieved in a shorter time if it had been implemented in the existing U-CARE

software system. However, the developer reported a number of learning benefits from developing the application such as: i) Becoming more productive in the long run due to the experience gained in the development of this one-time use app. The learning later went into the development of similar functionalities in the U-CARE software system; ii) An increased understanding of the full development cycle from development to deployment; iii) Learning how to set up a Windows server 2012 and install a secure socket layer (SSL) certificate; iv) Learning that s/he vastly underestimated the time needed to learn and set up a new Windows server 2012.

There were several advantages to stand-alone and one-time use application development, such as i) Quick turnaround on last-minute changes from stakeholders; ii) The source code was shorter and simple because the functionally was extremely limited; iii) Adhering to the accountability principle was achieved with very simple XML-based logging concept. Instead of columns and rows with a fixed structure, an XML node was created for every user action; iv) Since the app was deployed on a different server, the developer could do beta testing on the production environment, start and stop the web server, and restructure the database, with no negative impact on the existing U-CARE software system.

There were a few disadvantages to stand-alone and one-time use application development by a single developer, such as i) It led to critical knowledge being confined to one individual (a.k.a. a knowledge silo); ii) The total time spent was much greater than if the development team had updated the existing system; iii) The application was for one-time use only, since it was specifically tailored to a very narrow use case; iv) While working on this app, the developer was unable to work on the existing U-CARE software system.

A valuable lesson learned by the development team was that they could design a new simple software system for new research studies while keeping the existing U-CARE software system running with existing research studies and only consider cosmetic updates. Also, a new system could be run on the latest technological stack, whereas the existing technological stack could remain as-is for the existing U-CARE software system.

Appendix F: Design Principles Reformulation

F.1 Design Principles – Design Product

Table F.1-1 presents a walk-through of two versions, i.e., simplified (deleted text is stricken out while changes or additions are shown in bold) and reformulated (changes are underlined), of every design principle in the description. The product-related design principles were reformulated based on the hands-on design experiences from development of multiple features in U-CARE software system and retrospective mapping of design principles with multiple features.

Table F.1-1. Design principles for sustaining the usefulness of eHealth research software

Design principle	**Specification**
The principle of simplicity	*Simplified:* Provide easy-to-use ~~data export~~ functionali**ties** in order for ~~[clinical]~~ researchers to ~~export data~~ **do research,** preferably ~~by a single click~~ via a simple UI, given that such functionali**ties** should not require in-depth technical knowledge and should not overwhelm the researcher with details. *Reformulated:* Provide the eHealth research software with easy-to-use functionalities in order for researchers to use it in their [eHealth] research, preferably via a simple UI, given that such functionalities should not require in-depth technical knowledge and should not overwhelm the researcher with details.
The principle of modularity	*Simplified:* ~~Data export~~ functionali**ties** ~~should be~~ divided into modules in order for software developers to maintain and reuse, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules. *Reformulated:* Provide the eHealth research software's functionalities in modules to enable for maintenance and reuse by software developers, given that each module is simple, cohesive, and loosely coupled, such that a change to one module has minimal impact on other modules.
The principle of malleability	*Simplified:* a) ~~Customise: Data export~~ functionali**ties** ~~should be~~ customisable in order for [clinical] researchers to tailor [their own] research ~~data and descriptive metadata export and to import data to data analysis applications and statistical applications, given that such data export output should be in standardised or de facto formats, such as CSV or XML or tailored for spreadsheets or common statistical packages, in a way that is useful for downstream applications.~~ b) ~~Filter: Data export functionality should allow data filtering in order for~~ [experienced ~~clinical~~] researchers to customise ~~data export~~ according to their preferences and needs, given that such functionality should guide

	the researcher ~~to filter exportable data and allow the researcher to save and reuse their data exports as templates~~. ~~c) Schedule: Data export functionality should allow scheduling data export requests in order to get data after specified intervals [based on study design] or when data is available [in cases where the volume of data would increase data export processing time].~~ *Reformulated:* Provide the eHealth research software with customisable functionalities in order for [experienced] researchers to tailor them according to their **[potential]** needs, preferences, **or usage context,** given that such functionalities guide the researcher **during the customisation.**
The principle of accountability	*Simplified:* a) Privacy: ~~Data export~~ functionality ~~should~~ anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: ~~Data export~~ functionality that enables the ~~clinical~~ researcher (i.e., study owner or principal investigator) to restrict ~~data~~ access in order to enforce governance policies, ~~data extraction~~ and ethical guidelines, given that such ~~data~~ access restrictions can be researcher-specific (based on access privileges), ~~time-specific (i.e., at multiple intervals with the same/refreshed/additional datasets, or one-off after the study completion or termination)~~ and data-specific (i.e., partial, full, or selected datasets). c) Auditability: ~~Data export~~ functionality ~~should~~ log ~~all~~ activities related to ~~data export~~ **research** in order for study owner to fulfil audit and regulatory requirements, given that such logs store ~~all data export~~ events [when (timestamp), who (user identity – role), ~~how (encrypted/plain text), why (purpose specification and use)~~ and what (~~data~~ specification)] to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics. *Reformulated:* a) Privacy: Provide the eHealth research software with functionality that anonymise data in order to ensure research participants' privacy, given that such anonymised data do not contain identifiable data or that ID fields are encrypted, and datetime field(s) are removed or offset. b) Security: Provide the eHealth research software with functionality that enables the researcher (i.e., study owner or principal investigator) to restrict system and feature access in order to enforce governance policies and ethical guidelines, given that such access restrictions can be researcher-specific (based on sufficient access privileges), and data-specific (i.e., partial, full, or selected datasets). c) Auditability: Provide the eHealth research software with functionality to log activities related to research in order for study owner to fulfil audit and regulatory requirements, given that such logs store [accountability related] events [when (timestamp), who (user identity – role) and what (specification)] to facilitate follow-up by the study owner and enable audit organisations to confirm compliance with legislation and ethics.