



UPPSALA
UNIVERSITET

Design Principles for Data Export

Action Design Research in U-CARE

Mudassir Imran Mustafa



Uppsala University
Department of Informatics and Media



Design Principles for Data Export: Action Design Research in U-CARE

Mudassir Imran Mustafa

Supervisor: Dr. Jonas Sjöström

Thesis for the Degree of Master of Science in Information Systems
Department of Informatics and Media,
Uppsala University,
UPPSALA, Sweden
27 August, 2012

Abstract

In this thesis, we report the findings of designing data export functionality in Uppsala University Psychosocial Care Program (U-CARE) at Uppsala University. The aim of this thesis was to explore the design space for generic data export functionality in data centric clinical research applications for data analysis. This was attained by the construction and evaluation of a prototype for a data-centric clinical research application. For this purpose Action Design Research (ADR) was conducted, situated in the domain of clinical research. The results consist of a set of design principles expressing key aspects needed to address when designing data export functionality. The artifacts derived from the development and evaluation process each one constitutes an example of how to design for data export functionality of this kind.

Key words: Data Export, Data-centric Application, Clinical Research Application, Clinical Trial Management System, MVC, Action Design Research

Acknowledgements

I thank to Allah (God) Almighty the Creator of knowledge. I express my profound thanks to all my teachers, especially at the Department of Informatics and Media, for their advice and guidance throughout my studies. Special thanks to Anneli¹ for always being helpful, you are a great teacher, to Hafij² being my mentor, and to my supervisor Jonas³ for supervision and providing me opportunity to work on U-CARE platform. I am also thankful to my wife⁴ and my daughter⁵ for being my source of inspiration. At the end I am grateful to my parents and in-laws for taking care of all my worries and encouraged me to concentrate only on my studies.

¹ Dr. Annali Edman (anneli.edman@im.uu.se)

² Mr. Mohammad Hafijur Rahman (hafijur.rahman@pubcare.uu.se)

³ Dr. Jonas Sjöström (jonas.sjostrom@im.uu.se)

⁴ Mrs. Saima Zubair

⁵ Ms. Emaan Mudassir

List of Tables

Table 1: ADR problem formulation stage (adapted from Sein, et al., 2011)	6
Table 2: Design evaluation approaches / methods	10
Table 3: MVC frameworks	16
Table 4: Lab data in a generic schema	21
Table 5: Lab data in specific schema	21
Table 6: Problems and challenges regarding data export in U-CARE platform	30
Table 7: Design principles (version 1)	43
Table 8: Design principles (version 2)	55
Table 9: Design principles (version 3)	60
Table 10: Design Principles	62
Table 11: Summary of the ADR Process	64

List of Figures

Figure 1: ADR steps adopted from Sein, et al. (2011)	6
Figure 2: Research framework (based on Hevner et al., 2004 and Sein et al., 2011)	8
Figure 3: Illustration of the design process	9
Figure 4: IT-dominant BIE	10
Figure 5: N-Tier architecture	13
Figure 6: Web application architecture	14
Figure 7: MVC architecture control flow	16
Figure 8: LINQ to SQL ORM	18
Figure 9: Relational data model	18
Figure 10: ETL for data warehouse	22
Figure 11: ETL process	22
Figure 12: Data integration using wrapper	23
Figure 13: DTO implementation	23
Figure 14: SDO interaction between client and data source	24
Figure 15: U-CARE platform partial ERD	33
Figure 16: Data objects tree structure	37
Figure 17: User interface for data export fields selection	38
Figure 18: Modified user interface for pivoting selection	40
Figure 19: Data export module architecture (alpha)	41
Figure 20: Data export ERD	45
Figure 21: User interface for data export metadata list	46

Figure 22: Metadata tree structure for developer's verification	46
Figure 23: User interface to edit metadata	47
Figure 24: User interface to list data export packages	48
Figure 25: Modified user interface for data export fields selection using packages	48
Figure 26: Data export module architecture (beta v1)	49
Figure 27: Test study and test participants	51
Figure 28: Observation points in the randomization study	51
Figure 29: Quiz at first observation point associated to randomization process	52
Figure 30: Result variables for Inclusion	52
Figure 31: Data exported in excel format without pivoting	53
Figure 32: Data exported in excel format using pivoting	54
Figure 33: Modified user interface for data export fields selection using filters	57
Figure 34: User interface for data export filter	58

Abbreviations

ADR	Action Design Research
AR	Action Research
CSV	Comma-Separated Value or Character-Separated Value
DR	Design Research
DSR	Design Science Research
JCP	Java Community Process
JSR	Java Specification Requests
MVC	Model-View-Controller
RDBMS	Relational Database Management Systems
TDD	Test-Driven Development
XML	Extensible Markup Language

Table of Contents

1	Introduction	1
1.1	Case organization	1
1.2	Problem	2
1.3	Aim	2
1.4	Research Questions	3
1.5	Research Methodology.....	3
1.6	Structure	3
1.7	Audience	3
1.8	Delimitations	4
2	Research Approach and Theoretical Grounding	5
2.1	Action Design Research	5
2.1.1	Problem Formulation	6
2.1.2	Building, Intervention and Evaluation	7
2.1.3	Reflecting and Learning.....	7
2.1.4	Formalization of Learning	7
2.2	Research Framework.....	8
2.3	Design Process	8
2.4	Knowledge Base	11
2.4.1	Clinical Trial	12
2.4.2	Internet-based Self-help	12
2.4.3	Agile Software Development.....	13
2.4.4	N-Tier Architecture.....	13
2.4.5	Data Centric Web Applications	14
2.4.6	Design Patterns	15
2.4.7	Object Relational Mapping (ORM)	17
2.4.8	Relational Data Model	18
2.4.9	Recursion	20
2.4.10	Data Analysis	20
2.4.11	Data Transfer Objects.....	23
2.4.12	Service Data Objects	24
2.4.13	Metadata Mapping.....	25
2.4.14	Data Export Formats	25
3	Problem Formulation	27

3.1	Problem Setting.....	27
3.1.1	U-CARE Platform Development Process.....	27
3.1.2	U-CARE Platform	28
3.1.3	U-CARE Platform (Database) Design	29
3.2	Research Problem	30
3.3	Class of Problems.....	31
4	BIE Cycle I.....	32
4.1	Constructing the Prototype (Alpha)	32
4.1.1	Large and Complex System (Table 6-i).....	32
4.1.2	Generic and Flexible System (Table 6-ii).....	33
4.1.3	Dynamic Application (Table 6-iii).....	34
4.1.4	Non-normalized Database (Table 6Table 6-iv).....	35
4.1.5	Continuous Changing Data Models (Table 6-v)	35
4.1.6	Difficulties in Understanding Data (Table 6Table 6-vi)	35
4.1.7	Exporting (Business) Models - Designing Data Objects	36
4.1.8	Generic and Adaptable User Interface – Designing Reflection Utility	38
4.1.9	Pivoting (Row to Column Conversion) – Upgrading Reflection Utility.....	39
4.1.10	Render Useful Output – Designing Export Utility	40
4.2	Intervention and Evaluation with Practitioners	42
4.3	Identifying Design Principles.....	43
5	BIE Cycle II.....	45
5.1	Constructing the Prototype (Beta v1).....	45
5.1.1	Reflection on Every Export Request – Designing Persistent Metadata Structure.....	45
5.1.2	Reuse of Data Export Packages – Designing Persistent Package Structures	47
5.2	Intervention and Evaluation with End-users	50
5.3	Identifying Design Principles.....	55
6	BIE Cycle III	56
6.1	Constructing the Prototype (Beta v2).....	56
6.1.1	Restricted Access to Data – Configuring U-CARE Platform	56
6.1.2	Anonymity of Data	56
6.1.3	Render Required Data Only - Designing Filter.....	57
6.2	Intervention and Evaluation with End-users	58

6.3	Identifying Design Principles	60
7	Conclusion	62
7.1	Concluding Discussion.....	63
7.2	Future Work	65
8	References	66

1 Introduction

Sir William Thomson (Lord Kelvin) once said *“In physical science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it. I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it”* (Kelvin, 1889). Measurement plays a significant role when researcher attempts to test his theories by carefully designed and controlled experiments. An experiment is a process or study that results in the collection of data. Theory and experiment work together in science, with experiments leading to new theories that in turn suggest further experiments. Data analysis is required to understand the results of experiments which are not known in advance. Data analysis refers to a wide range of techniques to describe, explore, understand, prove or predict, based on sample datasets collected from populations, using some sampling strategy. We want to summarize data in a shorter form to understand some process and/or predict based on this understanding. The treatment of data has been recognized since time immemorial. One of such treatment is data summarization using tables. The tabular data not only led to data analysis but also Babbage’s early (mechanical) versions of computers were designed keeping tabular data processing in mind (Croaken, 2004). Data-centric clinical applications generate enormous data. The data is stored by database management systems (DBMS) like Oracle, Microsoft SQL Server, MySQL, PostgreSQL, etc. These applications have typical functions like create, read, update, and delete (CRUD) data. Some applications have more advance features like data grouping, ordering, and aggregation. Advance data analysis is performed by statistical analysis tools like Microsoft Excel, SPSS, SAS, etc.

In this research our focus was how to export data from a data centric clinical research application in a format so that it is readily available for data analysis tools without any further manipulations at researcher’s end. We proposed and implemented design principles for development of data export functionality in an information system, integrated best practices and design patterns, to create a generic data export tool which emerged from interaction with the organizational context. Our objective was to design a flexible infrastructure to develop data export functionality and make data export available to the researchers in a timely and labour extensive manner. Data integration, data migration etc. always remains top priority of the researchers but data export never being studied in detail. So the current study was focused on data export and proposed design for generic data export functionality.

1.1 Case organization

The current thesis is about the design of data export artifacts shaped during their development and use in Uppsala University Psychosocial Care Program (U-CARE) at Uppsala University. U-CARE programme (www.u-care.uu.se) goal is to promote psychosocial health among patients with somatic illness and their loved ones, at a lower cost to the benefit of individuals and society. For this purpose, research is conducted in close collaboration between research groups within the fields of information systems, clinical psychology, health economics and medicine at Uppsala University. Pilot research studies are conducted within the areas of cardiology, paediatric oncology and adult oncology in close collaboration with clinicians at Uppsala University Hospital. During 2010-2012, research

studies were designed, and a web application (U-CARE platform) for psychosocial care was developed (U-CARE, Uppsala University, 2012). U-CARE platform is a data-centric clinical research application to provide web-based psychosocial care (support, help and treatment) as well as manage clinical trials. Clinical trial is a scientific experiment in clinical research that is conducted to collect data about health interventions (e.g., drugs, diagnostics, devices, therapy protocols). A randomized controlled trial (RCT) is the preferred design for a clinical trial. The key distinguishing feature of the usual RCT is that study subjects are randomly allocated, to receive one or other of the alternative treatment, after assessment of eligibility and recruitment, but before starting the intervention. At the end of an RCT statistical method used to compare the intervention and control conditions' influence on the treatment outcome. For this analysis data is exported from the data collection application to the data analysis applications. U-CARE programme is divided to few work packages and each work package has a number of research studies. The research studies are related to the fields of paediatric oncological, adult oncological and cardiological care. There are even a number of associated studies within other clinical areas. The current on-going RCT are like:

- U-CARE: Treatment of posttraumatic stress among parents of children with cancer with cognitive behavioral therapy over the Internet
- U-CARE: A randomized controlled trial of the effect of a self-help programme via Internet on anxiety and depression among adolescents with cancer
- U-CARE: A randomized controlled trial of the effect of a self-help program via Internet on anxiety and depression among adults with cancer
- U-CARE: A randomized controlled trial of the effect of a self-help program via Internet on anxiety and depression among adults after a myocardial infarct

1.2 Problem

Healthcare technology, practice and knowledge are continuously evolving at a rapid pace. So the applications design and developed during any health care research project need to adopt itself continuously. A data-centric clinical research application, such as U-CARE, takes care of many research studies and clinical trials. Typically, the database is designed to manage data related to research studies. Data gathers by these health care application need to be interoperable between different application. Data interoperability and integration is one of the biggest issues in healthcare sector. There is always need of some sort of data export functionality to transfer data from, either application or directly from DBMS, to data analysis tools. The data-centric clinical research applications have mostly dynamic, flexible and generic design. The data export from these applications takes an incredible amount of time and is fraught with errors/defects. The chapter 3, problem formulation (p-27), provides details inside regarding problems and challenges during data export.

1.3 Aim

The aim of this thesis was to identify the design principles to develop generic data export functionality in data-centric clinical research applications. To fulfil this aim, we designed and introduced IT artifacts (as defined by

Hevner, et al., 2004) like data export prototype as instantiation, design principles as method and data export architectures as Model.

1.4 Research Questions

There are two main research questions:

- RQ1: Principles to guide data export design?
What are the current best practices and principles to *design* generic data export functionality, in a data-centric clinical research application, in line with existing information system design?
- RQ2: What are the implications of RQ1 for software design?
How can we *develop* generic data export functionality, in a data-centric clinical research application, by not only by following best practice and principles but also interacting with organizational context?

1.5 Research Methodology

The main research methodology was Action Design Research (ADR), proposed by Sein, et al. (2011). The research approach is discussed in detail in next chapter.

1.6 Structure

The structure of the thesis is as follows: Chapter-1 consists of several subsections which firstly introduces the topic of data export and give background details. Problem, aim, research questions, anticipated audience, and delimitations conclude the first chapter. Chapter-2 introduces research methodology, research framework, design process and knowledge base. Chapter-3 describes research problems in relation to existing system. Chapter-4, 5 and 6 provide inside the Action Design Research BIE cycles (explained in chapter 2). Chapter 7 includes the conclusions of the thesis work, a discussion of the findings, and recommendations for future work.

1.7 Audience

The proposed data export artifacts aims to add new knowledge to the existing information systems, clinical research, and data management knowledge bases used by researchers, practitioners, academician and/or managers. The artifacts are specifically designed to be used by the following:

- IS researcher who want to design an artifact
- Clinical researcher who want to export data for analysis from clinical trial application
- Software engineer who are responsible for designing data-centric applications
- Programmer who work with design pattern and develop large scale applications.
- Teacher who teaches software engineering, programming, system development and/or database management
- Project manager or clinical research associate who is responsible for management of clinical trials and/or clinical trial data

1.8 Delimitations

This study has several delimitations. It is delimited to the data export only for the purpose of data analysis in statistical tools. It is neither an alternative of data export nor replacement of backup functionality provided by database management systems. Also, data import was not in the scope of the study. Our approach to working with generic database designs is to build a software layer that insulates users from the schema, and enables them to export model objects in which they are interested. While such data export module can aid a certain class of users, some users will still need/want to work with clinical database directly e.g. database administrators and analysts proficient in SQL.

2 Research Approach and Theoretical Grounding

To identify the design principles the Action Design Research (ADR), proposed by Sein, et al. (2011), was the ultimate choice for research methodology. Even though ADR is a new method, its effectiveness is evident in recent studies related to artifacts designing (Lempinen & Tuunainen, 2011; Lempinen, et al., 2012; Fris, et al., 2011; Racz, et al., 2011). We further integrated the Design Science Research (DSR), proposed by Hevner, et al. (2004), to identify design implications. This chapter describes and explains the research approach adopted in this study. Further it focuses on the research framework. At the end a knowledge base related to data export presented that guided the design of data export artifacts.

2.1 Action Design Research

ADR is a new research method that combines action research and design research. ADR contribute to the advancing of theoretical understanding and the solving of current or anticipated problems through building and evaluating ensemble IT artefacts in an organisational setting (Sein, et al., 2011). ADR is a Design Research (DR) method primarily focused on technical aspects and generalised design knowledge of building IT-artefacts leaving the organisational context as a secondary consideration. DR has been criticised for being too theoretical and for the lack of connection to the problems of every-day practices in organisations. ADR resolve this by embracing principles of Action Research (AR). AR emphasises on solving problems in everyday practice and keep the organisational intervention as a primary concern. By combining aspects of action research into its design research foundation, ADR enlarges the focus of research on enterprises, helps to obtain technological rigor and organizational relevance, and bridges the technical and the organisational domain. This is attained by building an innovative artifact to solve a situated organizational problem that has been identified as an instance of a general class of problems. The lessons learned, while addressing a problematic situation, are continuously abstracted to add to the theoretical understanding of the problem class and its solution. This is then used to derive design principles as solutions to the class of problem (Sein, et al., 2011).

ADR consists of four stages: problem formulation; building, intervention and evaluation; reflection and learning; formalisation of learning. The three first stages are conducted iteratively and in parallel. The stages associated with a number of principles to guide the research. Figure 1 shows the ADR stages.

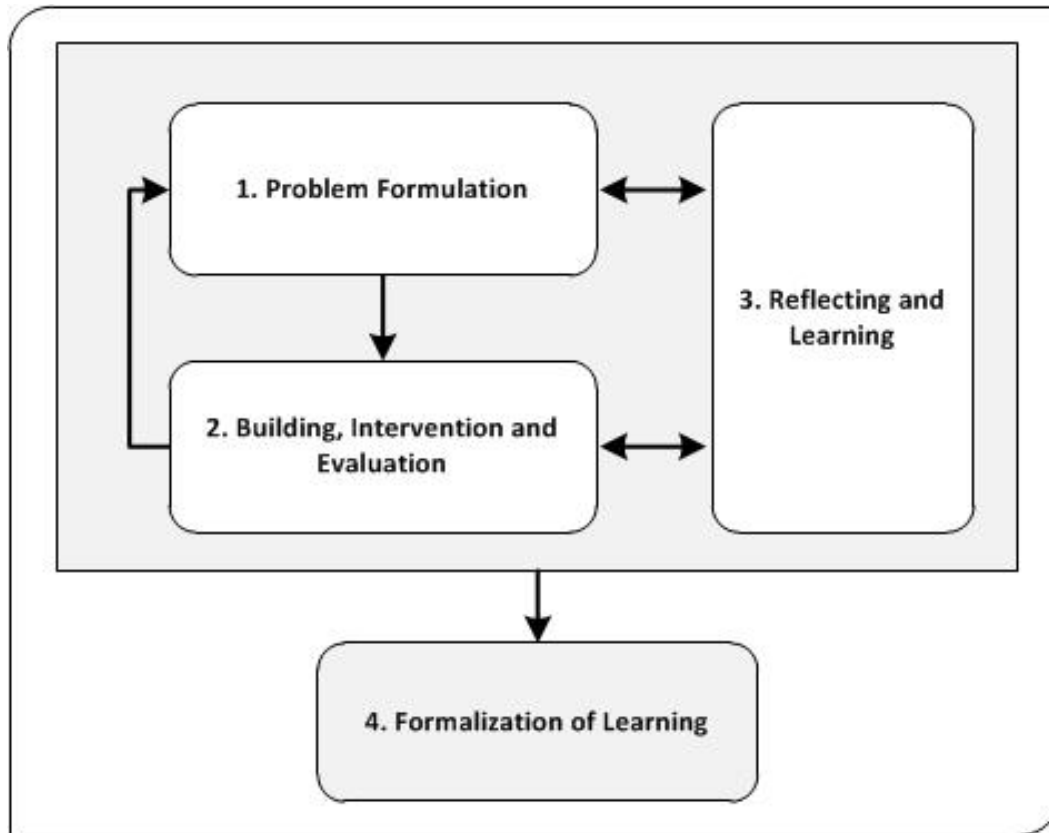


Figure 1: ADR steps adopted from Sein, et al. (2011)

2.1.1 Problem Formulation

In the first stage the research questions are formulated based on a problem; perceived in practice, anticipated by researchers or practitioners, encountered in existing technologies, or identified in prior research. The research problem is regarded as an instance of a class of problems for which the research aims to generate knowledge about. This stage also includes determining the initial scope, deciding the roles and scope for practitioner participation. Guiding principles in this stage are to keep the research inspired by practice and to maintain a theoretically sound base for the artifact. Critical issues in this stage are attaining the long-term commitment of the organization and articulating the identified problem as an instance of a class of problems.

Table 1: ADR problem formulation stage (adapted from Sein, et al., 2011)

Tasks	1) Identify and conceptualize the research opportunity 2) Formulate initial research questions 3) Cast the problem as an instance of a class of problems 4) Identify contributing theoretical bases and prior technology advances 5) Secure long-term organizational commitment 6) Set up roles and responsibilities
--------------	---

Principles	<p>1). <i>practice-inspired research</i></p> <p>The action design researcher should generate knowledge that can be applied to the class of problems that the specific problem exemplifies. As a result, the research activity is problem-inspired.</p> <p>2). <i>theory-ingrained artifact</i></p> <p>The action design researcher actively inscribes theoretical elements in the ensemble artifact, thus manifesting the theory “in a socially recognizable form”</p>
Challenges	Attaining the long-term commitment of the organization and articulating the identified problem as an instance of a class of problems.

2.1.2 Building, Intervention and Evaluation

The second stage of ADR uses the problem exploration and theoretical premises adopted in stage one to carry out a change in the target organization. The preliminary design of the artifact is based on the outcomes of stage one. During BIE (building, intervention and evaluation) stage the artifact is developed (**B**), put into the organizational situation (**I**). As the artifact is used in the organizational context it is assessed and refined (**E**) continuously to meet the needs of the end-users. Each BIE cycle not only evaluates the resulting IT artifact but also overall project and the research effort. Depending on kind of artifact this stage can be either IT-dominant BIE or organization-dominant BIE. The artifact design is the primary source of innovation in IT-dominant BIE whereas organizational intervention in organization-dominant BIE. Guiding principles in this stage are to allow: the artifact and organization shape one another; practitioners and researchers influence each other; evaluation to be continuous and organizational situated (Sein, et al., 2011).

2.1.3 Reflecting and Learning

Reflection and learning deals with the experiences and insights from the BIE stage in respect to problem formulated in the first stage. This stage is continues and parallel to first two stages. This stage emphasizes that the ensemble artifact reflects not only the preliminary design but how it is shaped in organizational context. The guiding principle of this stage is guided emergence of the artifact which means the design of the artifact emerged through the repeated cycles of BIE.

2.1.4 Formalization of Learning

In this final stage, in spite of the situated nature of ADR, the lessons learned in the organisational situation are further developed into general solutions for a class of similar problems. The solutions are formalised learning as design principles to address the class of problems derived from the design research outcomes. In this stage the guiding principle is to generalise the outcome of the organizationally situated intervention and artifact building.

2.2 Research Framework

The domain knowledge of the problem is very important to the success of research efforts. Theoretical grounding in the subject matter and having organizational context to build and evaluate artifacts is crucial toward achieving success. The theoretical basis was drawn primarily from data analysis, data warehousing, data mining, software engineering, information systems, healthcare and clinical science. The multi-disciplinary approach and U-CARE context in data export artifacts design aligns well with the Design Science Research guidelines put forth by Hevner et al. (2004) and stage wise principles suggested in ADR by Sein, et al., (2011). The research followed the research framework (see Figure 2) adopted from and Sein, et al., (2011) to achieve technological rigor and organizational relevance.

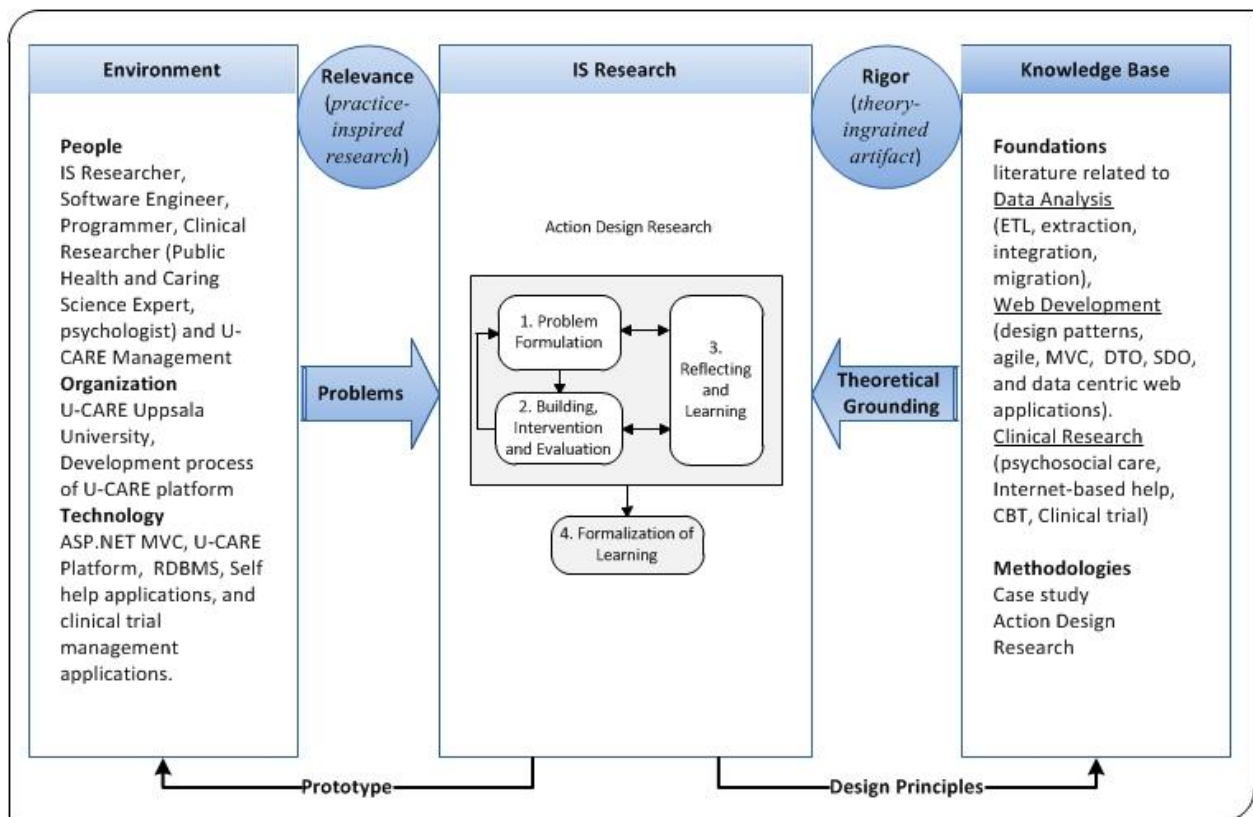


Figure 2: Research framework (based on Hevner et al., 2004 and Sein et al., 2011)

2.3 Design Process

The first stage of ADR is problem formulation which requires understanding existing system and practices. We also began with a thorough discussion together with IS researchers (practitioners) to find and formulate the research problems. Analysing the feedback from practitioners we continued by discussing and sketching suggested design solutions to answer research problems (identified during first stage of ADR) extracted from the domain of clinical research, data analysis and web development. We based this process on knowledge base, environment and information systems research by following research framework. The design work paralleled ADR stages i.e. “BIE”

and “reflecting and learning”. Figure 3 is illustration of how different aspects of the design process have influenced each other. During reflecting and learning stage the design solutions emerged that are in accordance with the principle of guided emergence. The design solutions implemented in prototype and next BIE cycle executed. We also continued to merge my findings with the knowledge base and design solutions to form design principles. In order to answer to the last stage in the ADR process, formalization of learning, we formalized design principles to provide guidance for further research and design work on data export functionality in research projects, outside the domain of clinical research. The design process is repeated three times; the first iteration of building was based on the problem formulation, the second followed up on the feedback received from the evaluation of the first and third followed up on the feedback received from the evaluation of the second.

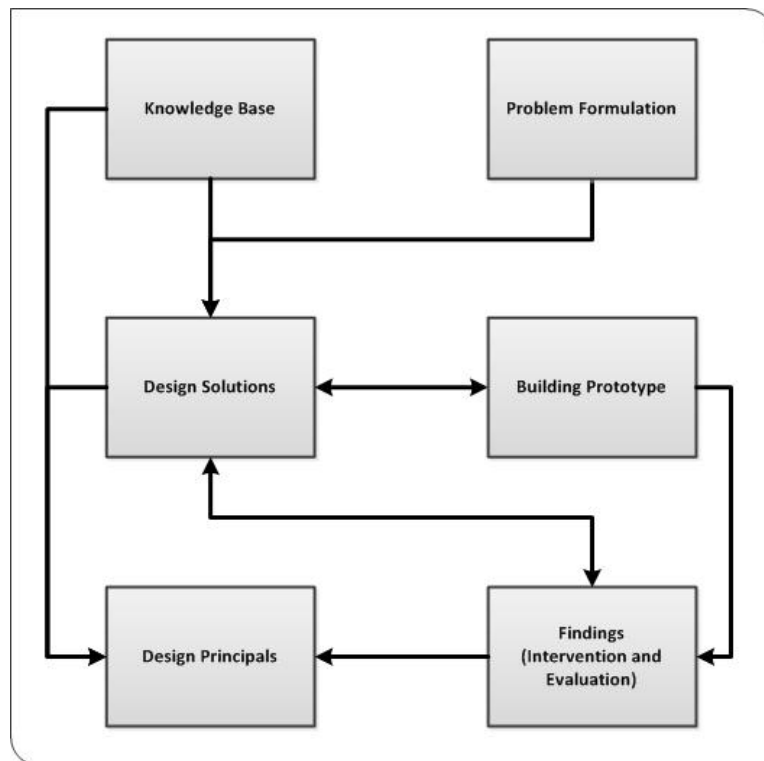


Figure 3: Illustration of the design process

The end users’ understanding is based on things they are already familiar with. We must understand the users to know what they might consider familiar. This understanding may be achieved by involving the users in the design process (Sein, et al., 2011). In ADR this is done by BIE cycles and required long term organizational commitment regarding end users’ active participation in the entire research process. Even though the current study was initiated by IS researchers, the end users in U-CARE were committed to participate in the research (possibly due to being researcher themselves). Figure 4 visualises BIE cycles during design process based on the ADR’s Generic Schema for IT-dominant BIE (Sein, et al., 2011). Furthermore it also pictures different participating users’ roles and responsibilities. The numbered phases represent: 1) the building of the alpha prototype, 2) the evaluation thereof, 3)

the building of the first version of beta prototype, 4) the intervention & evaluation thereof, 5), the building of the second version of beta prototype, 6) the last intervention & evaluation session, and 7) the proposed future building of next version. Following the ADR process the next step would be to take beta version prototype to next stage of development, given that the research studies under U-CARE have reached a stage where it is relevant to export real data. The aim of this thesis was to examine the design principles for data export functionality by building a prototype, and reaching a stage where end users can start exporting data for data analysis was not within the scope of this thesis.

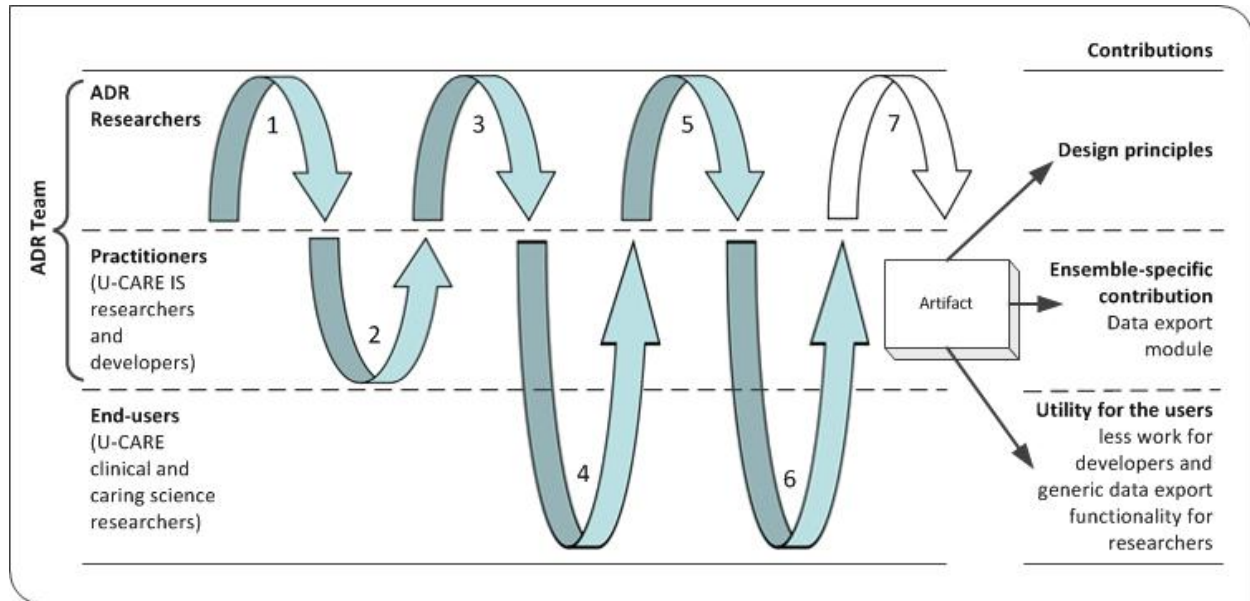


Figure 4: IT-dominant BIE

As the current research methodology required building new artifact utility which leads us to two approaches; the artifacts building approach and artifacts evaluating approach (Järvinen, 2000). The Action Design Research (ADR) contains both building and evaluation in the same process. We also evaluate (justify/evaluate) the data export module by following Hevner's design evaluation methods. There are five approaches or methods for evaluation in Design Science proposed by Hevner et al., (2004) . Table 2 explains the evaluation methods used during different BIE cycles.

Table 2: Design evaluation approaches / methods

Evaluation type	Evaluation method	Description
Observational	Case study	Over all ADR was performed in case organization.
	Field study	Field study is proposed for future implementation. Infact the author is already involved in another research project regarding data export functionality and have plan to implement the same

Evaluation type	Evaluation method	Description
		design for their research application to evaluate data export artifacts use in wide variety of projects.
Analytical	Static analysis	BIE I
	Architectural analysis	BIE I and II
	Optimization	Did not perform yet.
	Dynamic analysis	Did not perform yet.
Experimental	Controlled experiment	BIE III
	Simulation.	Did not perform yet.
Testing	Functional (black box) testing	Following agile principles, testing was performed on daily bases.
	Structural (white box) testing	As unit testing was performed on each model of the U-CARE platform so most of the system was automatically tested. But the data export module classes still require systematic unit testing.
Descriptive	Informed arguments	Did not perform yet.
	Scenarios	BIE II

2.4 Knowledge Base

This section discusses the basic concepts, related to U-CARE platform development, in the most general context for understanding the existing system and formulating problem in U-CARE context (in this research situated/case organization). For this purpose, a variety of literature sources were researched and reviewed; those included Information Systems journals, Computer Science and Software Engineering journals, Medical & Healthcare journals, Psychology journals, on-line publications, and professional resources like websites, books, forums, technology manuals, knowledge bases. Searching was done by trial-and-error using combinations of keywords to get the best match of articles covering the problem. Furthermore search is extended using search engines like Google (www.google.com) and Primo Central (<http://www.ub.uu.se/>) using a similar trial-and-error strategy. The

knowledge base is intended as a primer for the non-technical reader, explaining many key terms and concepts that will be used throughout this thesis, so that he/she may fully understand the significance of the research project.

2.4.1 Clinical Trial

Clinical trial is a powerful experimental technique in (patient-oriented) clinical research for assessing the effectiveness of an intervention. During clinical trial in any research study researchers assigns human participants or groups of humans to one or more health-related interventions to evaluate the effects on health outcomes. Clinical trial requires an administrative structure that includes a study director (sometimes referred to as Principal Investigator of the study) and committees that have responsibility for planning and for operational decisions during the course of the trial. The persons responsible for study activities are referred to as “investigators”. Those enrolled and followed in a trial are referred to as “participants”. Activities in a clinical trial are divided into phases. The first phase after funding has been attained is dedicated to preparing the required study documents, that are, the study protocol; the manual of operations; and the necessary study forms, including the informed consent form; establishing the study organization; and developing the data management system. The second phase is the period of patient recruitment and collection of baseline data. The third phase is for follow-up data collection. The last phase includes close-out of study, data analysis, and preparation of publications (Knatterud, 2002).

Clinical trial operations represent one of the most resource-demanding areas within medical science and often consist of tedious, manual processes for collecting, aggregating and visualizing information from a variety of data sources. A clinical trials management system (CTMS) offers efficiencies and cost savings for clinical trial operations by providing a single, centralized system to coordinate the operational and administrative activities across multiple studies. Web based CTMS provides more efficient and innovative data capturing.

2.4.2 Internet-based Self-help

Internet-based self-help programs in general an interactive website consist of modules to be completed by a participant independently. Modules may include reading materials, images, audios, videos, quizzes, and the assignment of homework. Website contents are presented on a step-by-step basis (Andersson, et al., 2005). Advantages of Internet-based self-help programs can be seen not only in the broader reach, but also in increasing convenience for the participants, the opportunity to provide information in an interactive and timely manner (Kelders, et al., 2012). Research has demonstrated that Internet-based self-help websites can be helpful for reducing symptoms for a variety of disorders, including anxiety disorder (Berger, et al., 2009; Titov, et al., 2009; Andersson, et al., 2006) depression (Perini, et al., 2009; Andersson, et al., 2005) and pain management (Rini, et al., 2012). Furthermore, several randomized controlled trials comparing face-to-face treatment with Internet-based self-help treatments have revealed that self-help websites produce results comparable to face-to-face treatments (Barak & Grohol, 2011).

2.4.3 Agile Software Development

Globalized and unsettled business environments joined with rapid advancements in technology put new strains on software developers. User requirements are often hard to define or visualise and can seldom be assumed to be stable throughout a project. As a consequence, a software development methodology referred to as agile has emerged (Cockburn, 2001). Agile methods operate on the “just enough” approach and are tailored to “embrace change”. Some of the key points of Agile Manifesto (Beck, et al., 2001) are:

- ✓ Customer satisfaction by rapid, continuous delivery of useful software
- ✓ Working software is delivered frequently and is the principal measure of progress
- ✓ Late changes in requirements are welcome
- ✓ Close (daily) cooperation between business people and developers
- ✓ Face-to-face conversation is the best form of communication
- ✓ Projects are built around motivated individuals, who should be trusted
- ✓ Continuous attention to technical excellence and good design
- ✓ Simplicity
- ✓ Self-organizing teams
- ✓ Regular adaptation to changing circumstances

2.4.4 N-Tier Architecture

N-tier architecture provides a model for software developers to create a reusable and flexible application. N-tier means any number of levels arranged above another, each serving distinct and separate tasks. By breaking up an application into tiers, developers only have to modify or add a specific layer if they decide to change technologies or scale up, rather than have to rewrite the entire application over.

Presentation GUI	HTML, windows froms, etc. physically on client's machine		
Presentation Login Tier	The Web (Server-Sided) web forms, ASP.NET, C#, VB.NET, VBScript, Jscript, etc.	Client Interface (Client-Sided) windows froms, a custom application, etc.	
		Distributed Logic (Client-Sides) need to connect to proxy layer	
		Proxy Tier (Server-Sided) .Net Romoating, etc.	
Business Tier	Business objects and rules, Data manipulation and transformation into information		
Data Access Tier	Interface with the database, Handles all data input/output, made to scale, usually stateless		
Data Tier	Storage, Query & Storage optimization, Performance (Indexing, etc.)		

Figure 5: N-Tier architecture

The most widespread use of n-tier architecture is the three-tier architecture. In three-tier architecture clients remain focused on presenting information and receiving input from users. This is known as the presentation tier. Data, meanwhile, is hosted on one or more data servers in the data tier. Application tier (business logic, logic tier, data access tier, or middle tier) is the part of the software application that deals with the performance of business-related tasks. Code in the application tier operates on data that attempts to model entities in the problem domain - invoices, customers, orders, and the like.

2.4.5 Data Centric Web Applications

These days, web applications are almost ubiquitous. The web has become a platform not only for information delivery, but also for healthcare systems, social networks, mobile services, and distributed learning environments. Data centric web applications which make large amounts of data available on the Web. These applications main task is to enable the browsing of complex data collections. To develop such application web developers have to consider the design dimensions such as: (data layer) the schemas of the application data on the server and their mapping to pre-existing data sources; (application layer) the business logic of the application; (presentation layer) the hypertexts for navigating within them, thus generating the web interface and styles of presentation being offered to the user on the client (Casteleyn, et al., 2009).

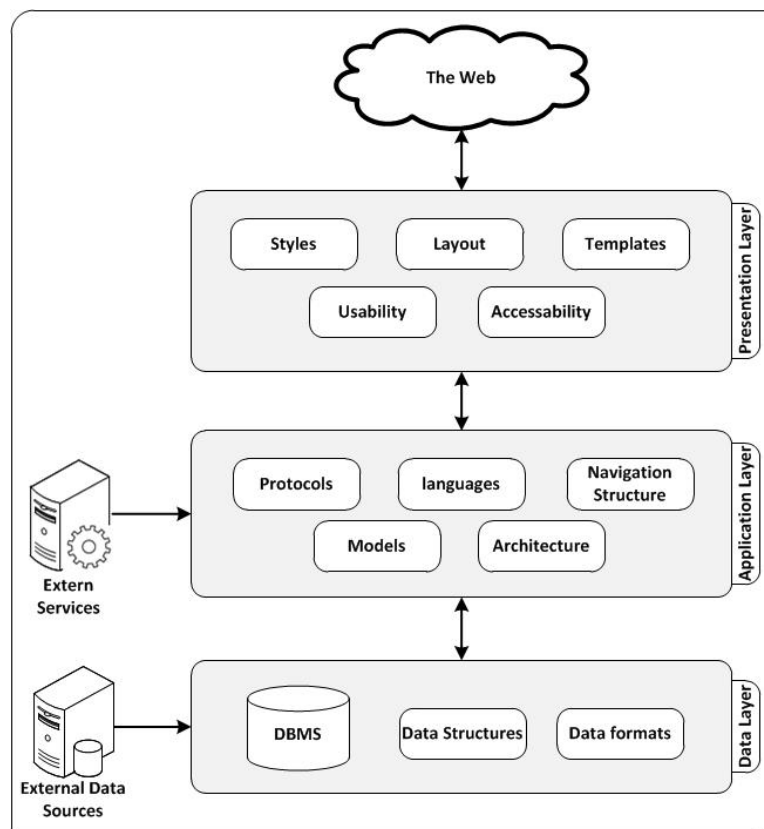


Figure 6: Web application architecture

Casteleyn, et al., (2009) described three well-known conceptual web application development methods (specifically data-centric web applications), i.e., WebML-web modelling language (Ceri, et al., 2002), MSDM -web site design method (Troyer & Leune, 1998), and OOHDM – (Schwabe & Rossi, 1998).

2.4.6 Design Patterns

Object orientation is the dominant paradigm in software development. Designing object-oriented software is tough, and designing reusable object-oriented software is even tougher. We have to find relevant objects, factor them into classes, define class interfaces and inheritance hierarchies, and create essential relationships among them. Design should be specific to the current problem we have but also general enough to address future problems and requirements. We also want to avoid redesign, or at least minimize it. It is hard if not impossible to develop a reusable and flexible design "right" during first attempt. Designer usually try to reuse a design several times, modifying it each time before it reached to optimal solution. This leads us to find a design solution for the same/similar problem(s) everytime, and we end up "re-inventing the wheel" every now and then. Design patterns can be used to assist us in solving related problems. Gamma, et al. (1995) defined the design pattern by quoting Christopher Alexander as, *"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice"*. Design pattern has four essential elements as pattern name, problem, solution, and consequences. Design patterns improve the documentation and maintenance of existing systems by providing an explicit specification of class and object interactions and their underlying intent. A framework is a set of co-operating classes that make up a reusable design for a specific class of software. A typical framework contains a number of design patterns. Frameworks are becoming increasingly common and important. They are the way that object-oriented systems achieve the most reuse. Most of the design and code in the application is come from or be influenced by the frameworks it uses. Larger object-oriented applications are consisting of layers of frameworks that cooperate with each other (Gamma, et al., 1995). There are several design pattern related to this study like Active Record, Domain Model, Adapter, Reflection, Observer, Model-View-Controller (MVC) etc. but only one is explain in details as it is necessary to understand the overall system and more specifically data export functionality.

2.4.6.1 Model–View–Controller Pattern

Model-View-Controller (MVC) is a design pattern, where the data are represented by the *model* and the *view* by the visual component. The *controller* is the communication between the *model* and *view* objects (Gamma, et al., 1995). MVC is adapted as architecture for World Wide Web applications architecture. It separates each layer (*model*, *view*, and *controller*) for different purpose allowing independent development, maintenance and testing. The *model* is responsible for acting on certain logical conditions or rules by managing data from the application domain. It takes some information, possibly stores information, gives some new information and changes its current state according to the directions given by the *controller*. The *view* displays the *model*'s state in an appropriate format so that the users can easily understand. MVC control flow for a web application generally works as shown in Figure 7.

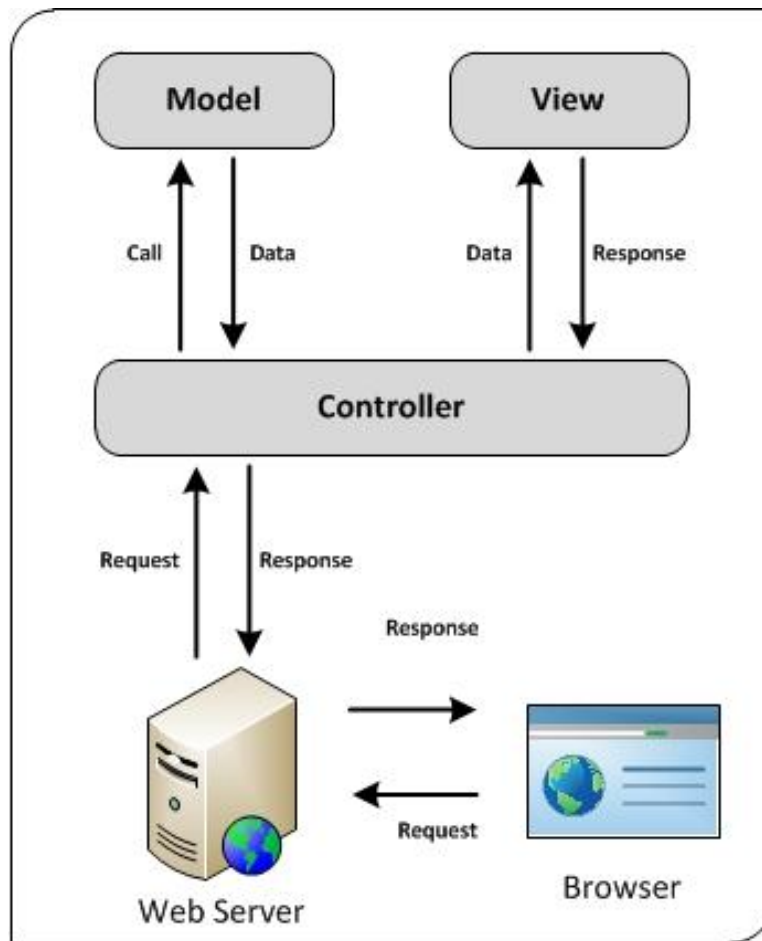


Figure 7: MVC architecture control flow

The user interacts with the user interface using browser e.g. presses a login button. A controller handles the input from the user interface (e.g. user name and password). The controller notifies the model of the user action, possibly resulting in a change in the model's state (e.g. user model change to login user). A view uses the model (indirectly) to generate an appropriate user interface (e.g. the view produces a screen for login user). The view gets its own data from the model and model has no direct knowledge of the view. The user interface waits for further user interactions, which begins the cycle once again. By decoupling models and views, MVC helps to reduce the complexity in architectural design, and to increase flexibility and reuse. Several commercial and non-commercial application frameworks have been created to enforce the design. Frameworks those are broadly used in these days for developing web applications can be seen in Table 3.

Table 3: MVC frameworks

Programming language	Framework
ASP.NET (C#, VB.NET)	ASP.NET MVC (http://www.asp.net/mvc)

Programming language	Framework
PHP	Zend (http://framework.zend.com/), CodeIgniter (http://codeigniter.com/), CakePHP (http://cakephp.org/)
Java	Struts (http://struts.apache.org/)
Ruby	Ruby on Rails (http://rubyonrails.org/)

ASP.NET MVC is a web development framework from Microsoft that combines the effectiveness and tidiness of MVC architecture, the most up-to-date ideas and the techniques from agile development, and the best parts of the existing ASP.NET platform.

2.4.7 Object Relational Mapping (ORM)

All applications developed today manipulate data in one way or another and generally this data is stored in a relational database. Applications developed in an object-oriented programming language like C#, Java, or PHP need some form of object/relational mapping as object model and relational model are two different paradigms. The object model deals with classes, instances of those classes and relationships between those instances. The relational model instead deals with tables, rows and the relationships between those tables. Therefore mapping between the two paradigms is not trivial but an Object Relational Mapping tool (ORM) can ease this process by providing the means to automatically map from the object model into the relational model (and back again). For example Visual Studio's Object Relational Designer, part of the LINQ to SQL ORM tool, generates the classes for data model automatically for the .NET developers. Together, the LINQ to SQL run-time infrastructure and design-time tools significantly reduce the workload for the database application developer. Visual Studio brings some facilities that allow developers to map their database tables into objects in a very easy way. Developers have to add a LINQ to SQL mapping file to the Visual Studio Project. The file is a DBML (Database Mapping Language) file. Then drag and drop the tables that will be mapped by using server explorer. LINQ to SQL relies on an object-to-relational model set of classes that map to our database tables. We can access these classes through the data context subclass also created in the dbml file. Each object we create is mapped to a row in a database table. The object has one property for each table's column of the same type and with the same constraints. The Active Record pattern recommends that each object makes itself responsible for its own persistence but LINQ to SQL delegate persistence to an integrated O/RM infrastructure that acts as the real data access layer.

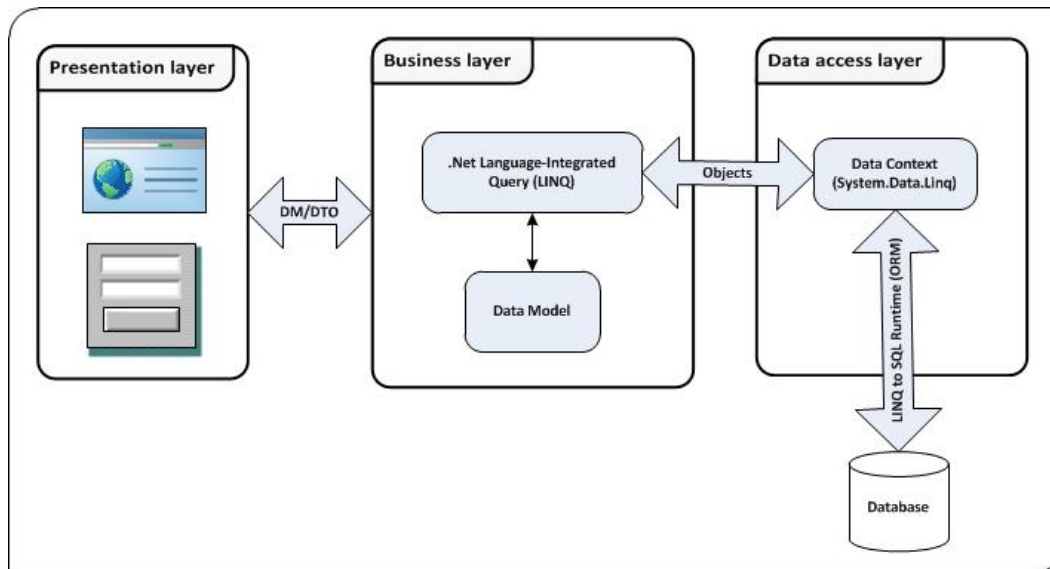


Figure 8: LINQ to SQL ORM

2.4.8 Relational Data Model

Codd (1970) proposed the relational model for database management is a database model based on first-order predicate logic. There has been a number of commercial and open source software applications, known as relational database management systems (RDBMS), based on Codd's ideas, including Oracle Database, Microsoft SQL Server, PostgreSQL, MySQL, DB2, and many others. SQL data definition and query language is used by most of these RDMS. The relation is the basic element in a relational data model. Relation is a two-dimensional table. Column in the table are attribute (i.e. field or data item). Each column in the table has a unique name within that table. A tuple (i.e. record) is a row in the table. The consistency of a relational database is enforced, not by rules built into the applications that use it, but rather by constraints, declared as part of the logical schema and enforced by the DBMS for all applications.

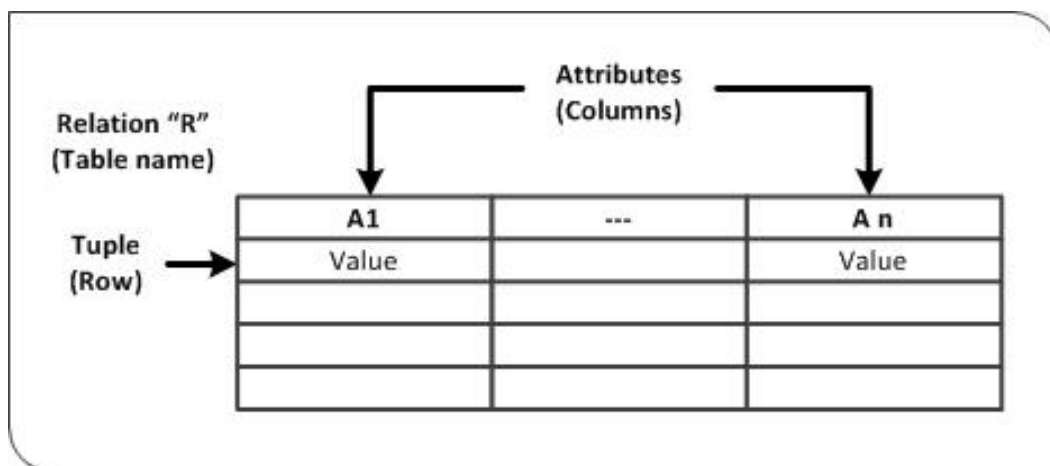


Figure 9: Relational data model

2.4.8.1 Constraint

Constraint can be described as rule defined in RDBMS e.g. candidate key, primary key, foreign key and default. A simple key contains a single attribute where as a composite key contains more than one attributes. A *candidate* key is an attribute (or set of attributes) that uniquely identifies a row. A *primary* key is the candidate key which is unique identifier. Every relation must contain a primary key to enforce entity integrity (not allowing multiple rows to have the same identity within a table). A *foreign* key is a field within a database record that points to a key (or group of fields forming a primary key) of another record in a different table. In this arrangement, a foreign key in one table will typically refer to the primary key of another table. This enables references which can be made to link information together. This is a major component of database normalization. Foreign keys are used to maintain the data integrity (known as referential integrity, requiring the existence of a related row in another table) which results a minor performance overhead because before inserting data into the child table, the process has to check whether the corresponding parent key is there in the parent table or not. The *default* constraint is used to insert a default value (character, string or number) into a column if no other value is specified while adding new record(s). There are other constraints like check, not-null and unique as well.

2.4.8.2 Logical Deletion

Logical deleting means setting a flag stating that the record is deleted rather actually or physically deleting the record. Advantages of logical deletion are to keep the history (good for auditing, troubleshooting, and reporting) and we don't have to worry about cascading a delete through various other tables in the database that reference the row we are deleting. Disadvantage is that we have to take the flag into account while coding any methods in business/application layer. Another disadvantage is performance implications of keeping all that data.

2.4.8.3 Surrogate Key

Most tables have some kind of natural key, where a set of columns provide a good candidate for defining the unique attributes of each row. Data changes in an organisation's business processes so using the natural key vulnerable to changes. Ideally we want our primary keys to be immutable, i.e. not subject to change over the lifetime of the application. To counter this problem data designers prefer to add an extra key into the design, i.e. surrogate key, ensuring that business logic does not appear in the primary key. The surrogate key does not occur naturally in the data - it is an artificial construct designed to provide each row with a unique value - generated by the database management system. A MS SQL Server or Sybase 'identity column', a PostgreSQL 'serial', an Oracle 'Sequence' or a column defined with 'Auto increment' in MySQL is often used to provide this surrogate key. Surrogate keys can also become useful where the natural key is inconvenient or difficult to work with, i.e. there are too many columns or the columns include large amount of text. Using an identity column over a composite key complicates the data design. In order to guarantee data integrity we have to either add a constraint onto the columns or code in the application.

2.4.9 Recursion

Recursion is a powerful general-purpose programming practise and is the key to many critically important computational applications. It provides basic support for information processing like combinatorial search and sorting methods. Nearly all programming languages support recursion by allowing a function to call itself within the program. A recursive function definition has one or more base cases, meaning input(s) for which the function produces a result without recurring, and one or more recursive cases, meaning input(s) for which the program calls itself. The job of the recursive cases can be seen as breaking down complex inputs into simpler ones. In a well-designed recursive function, with each recursive call, the input problem must be simplified in such a way that eventually the base case must be reached.

2.4.10 Data Analysis

Current technological advancement permits the storage and access of large amounts of data at virtually no cost. The main problem in current data-centric applications is how to use the collected raw data. The true value is not in storing the data, but rather in our ability to extract useful information and to find interesting results, through the use of statistical analysis and inference, to support or reject hypotheses and conclusions made by researchers (Fayyad, et al., 1996). Clinical research is a highly interdisciplinary field that employs a variety of quantitative and qualitative methods. The researchers choose different software applications to fulfil study's analytical needs. Many statistical software brands such as SAS, Stata, and SPSS, suite of general and specialized statistical software applications, market by leading vendors like SAS Institute, Inc. (Cary, North Carolina), StataCorp LP (College Station, Texas), and SPSS Inc. (a subsidiary of IBM, based in Chicago, Illinois). Spreadsheet applications such as Microsoft Excel (Microsoft Corporation, Redmond, Washington) are also commonly used for statistical analysis.

Before any attempt can be made to perform the data analysis and extraction of useful knowledge, we need to understand how data should be extracted. Therefore, my focus is not on describing the methods that can be used for data analysis, but rather on discussing the process that supports the data export in useful format that leads to data analysis. In data analysis applications, the preparation of data for analysis constitutes considerable percentage of the total effort. This is because most data analysis tools require that data to be organized into a single table with specific, named columns. Typically, clinical data are distributed across several tables, and have generic columns with coded values.

The principal difficulty of generic database designs is the use of coded values instead of specific column names. This results in a schema that is conceptually difficult for users to manipulate and which performs very poorly for queries involving multiple participant records. A classic example of a generic schema is shown in Table 4, in which lab data are identified by codes A, B and C.

Table 4: Lab data in a generic schema

R-No	Date	Test	Value
1	02-10-2012	A	125
1	02-10-2012	B	5.2
1	02-10-2012	C	206
1	05-15-2012	A	135
1	05-15-2012	B	7.2
1	05-15-2012	C	175
2	03-06-2012	A	130
2	03-06-2012	B	6.0
2	03-06-2012	C	190

In contrast, a specific schema employs distinct columns names for each attribute of interest. Table 5 shows the same data as in Table 4, but arranged in a table in which the value of each lab test can be identified by its column name. This schema is for easier to use in data analysis applications.

Table 5: Lab data in specific schema

R-No	Date	A	B	C
1	02-10-2012	125	5.2	206
1	05-15-2012	135	7.2	175
2	03-06-2012	130	6.0	190

The rapid adoption of databases and growing data-reliance forced led to the need to share or to merge existing repositories. As a solution a lot of companies build data warehousing systems. The data warehouse (DW) system extracts, transforms, and loads data from heterogeneous sources into a single common queriable schema. DW is used for reporting and analysis. Extract -Transform-Load (ETL) is a process for the extraction of data from several sources, cleansing the data and customized insertion of the data into a data warehouse (see Figure 10). The ETL tools improve and facilitate data warehousing.

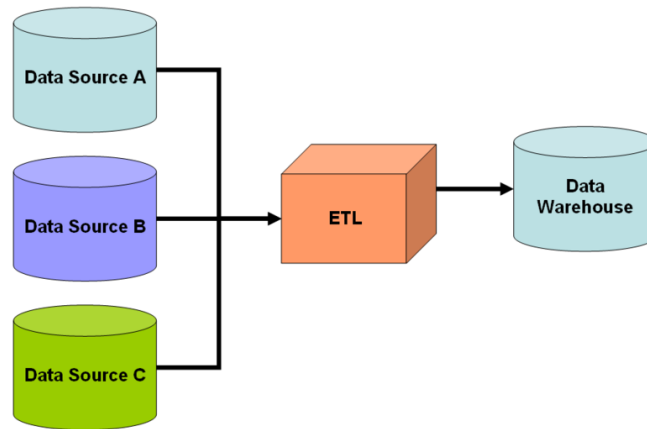


Figure 10: ETL for data warehouse⁶

The first task is data extraction from internal or external sources. The second task is transforming data into a structure which is required to continue the operation like sorting data, cleansing, checking quality etc. The third task is loading into a data warehouse (see Figure 11).



Figure 11: ETL process

Data integration includes combining data from different sources and providing a unified view of data to users. In a variety of situations data integration becomes significant e.g. when two similar companies need to merge their databases or researchers need to combine results from different data repositories (see Figure 12). Data export is similar to data extraction and integration for data warehouse. The only difference is that in data warehouse data is accumulated over years and later used for complex types of data analysis.

⁶ Source: <http://en.wikipedia.org/wiki/File:Datawarehouse.png>

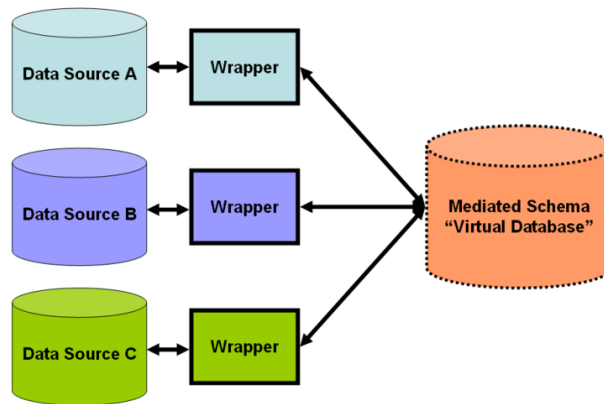


Figure 12: Data integration using wrapper⁷

2.4.11 Data Transfer Objects

Data Transfer Objects (DTOs) are ad hoc container objects used to pass data from the presentation layer to the service layer, and back. A DTO is nothing more than a container class that exposes properties but no methods. A DTO is helpful whenever we need to group values in ad hoc structures for passing data around. DTOs help to further decouple presentation from the business layer and the domain model. The adoption of DTO adds a good deal of flexibility to the business layer and subsequently to the design of the entire application (Fowler, et al., 2002).

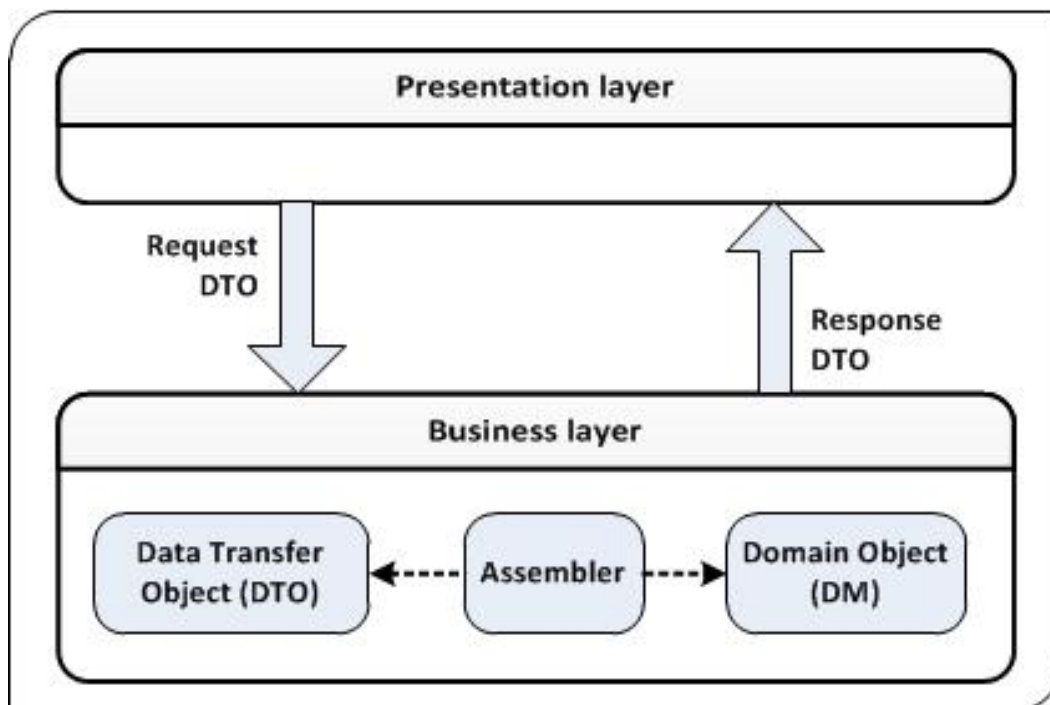


Figure 13: DTO implementation

⁷ source: <http://en.wikipedia.org/wiki/File:Dataintegration.png>

When DTOs are employed, we also need a DTO adapter layer (an assembler) to adapt one or more entity objects to a different interface as required by the use case. In doing so, we actually implement the "Adapter" pattern. A single DTO usually contains more than just a single server object. It aggregates data from all the server objects that the presentation layer likely to want data from. The fields in a DTO are simple classes like strings and dates, or other DTOs. Structure between data transfer objects is simple hierarchy graph structure. We need to populate DTO with data which derived from more than one domain object. DTO cannot extract the data from the domain objects as it has no behaviour. This is reasonable, because keeping the DTO unaware of the domain objects enables us to reuse the DTO in different contexts. Similarly, we do not want the domain objects to know about the DTO because that may mean that changing the DTO would require changing code in the domain logic, which would lead to a maintenance nightmare. The best solution is to use the Assembler pattern, which creates DTOs from business objects and vice versa. Assembler is a specialized instance of the Mapper pattern.

2.4.12 Service Data Objects

Service Data Objects (SDO) is designed to simplify and unify the way in which applications handle data from heterogeneous data sources, including relational databases, XML data sources, Web services, and enterprise information systems. SDO is based on the idea of disconnected data graphs. A data graph is a collection of tree-structured or graph-structured data objects, with a root data object and traversal systems (depth/breadth-first) that allow client programs to traverse the elements. The data graph facilitates communication between structural tiers and various service-providing entities. Disconnected mean a client retrieves a data graph from a data source, modifies the data graph, and can then apply the data graph changes back to the data source. SDOs required a data access service (DAS). The task of a DAS is to retrieve and write data to and from a data source (IBM & BEA, 2006). The following diagram illustrates a typical client interaction.

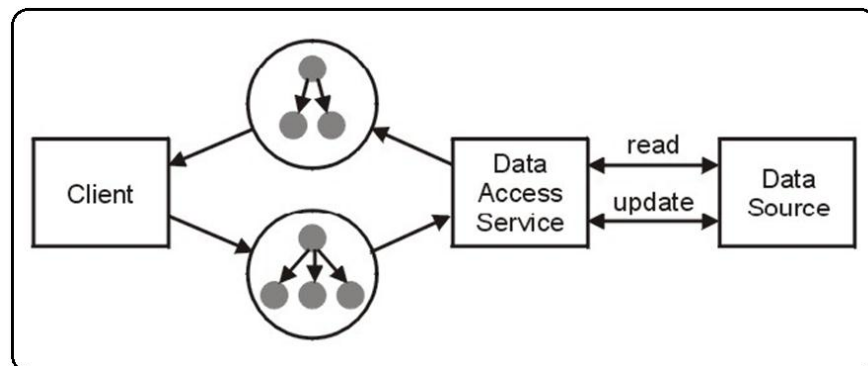


Figure 14: SDO interaction between client and data source

SDOs support simple meta-data that describes types, their attributes and their relationships. This makes it possible to develop generic applications (for instance, a data explorer/browser interface) for interacting with heterogeneous data sources (JCP, 2007).

2.4.13 Metadata Mapping

Metadata mapping is related to object-relational mapping and describes how fields in the database correspond to fields in in-memory objects. This is already explained in section 2.1.6 (Object relational mapping). This section is related to metadata mapping of an objects container like data transfer object (DTO), service data object (SDO), etc. There are two main methods for metadata mapping; code generation and reflective programming. With code generation we develop a program whose input is the metadata and whose output is the source code of classes that do the mapping. These classes are entirely generated prior to compilation and deployed with the server code. It is a less dynamic approach as any changes to the container require recompiling and redeploying. A reflective program takes an object and treats methods (and fields) as data. With a reflective approach, changes can even be done during runtime by just rereading the metadata.

2.4.14 Data Export Formats

Exporting data is generally a multi-step process. First, the metadata needs to be pulled from the database (table name and table's columns name) and is rendered for user selection. The next step is collection data from database according to the user selection in a temporary storage. The last step, involves converting the data into desired format like CSV, XML or Excel. The most common task is to write a matrix or data frame to file as a rectangular grid of numbers, possibly with row and column labels.

2.4.14.1 CSV

There are a number of issues that need to be considered in writing out a data frame to a text file. It is preferred to have a header row which contains the column names. A common field separator to use in the file is a comma, as that is unlikely to appear in any of the fields in English-speaking countries. Such files are known as CSV (comma separated values) files. In some locales the comma is used as the decimal point and there CSV files use the semicolon as the field separator. By default missing values are output as empty string but can set to NA. By default strings are quoted. Some programs, for example Mondrian, do not accept quoted strings (which are the default). Some care is needed if the strings contain embedded quotes. Text files do not contain metadata on their encodings, so for non-ASCII data the file needs to be targeted to the application intended to read it. All of these functions can write to a connection which allows an encoding to be specified for the file. The advantage of CSV files comes from its age and its widespread compatibility. We can open a CSV file in almost any application, even on a text editor.

2.4.14.2 XML

When reading data from text files, it is the responsibility of the user to know and to specify the conventions used to create that file, e.g. the comment character, whether a header line is present, the value separator, representation for missing values, etc. A markup language which can be used to describe not only content but also the structure of the content can make a file self-describing, so that one need not provide these details to the software reading the data. The eXtensible Markup Language (XML) can be used to provide such structure, not only for standard datasets but also more complex data structures. XML is becoming extremely popular and is emerging as a standard for general

data markup and exchange. It is being used by different communities to describe geographical data such as maps, graphical displays, and mathematics and so on. XML also provides a way to specify the file's encoding.

2.4.14.3 *Excel*

Excel is a spread sheet application which holds information in the workbook. A workbook consists of worksheets. A worksheet contained both content and formatting (number masking, colouring, conditional formatting, etc.). Values are arranged in a tabular form. Excel is way more superior compared to CSV as it is able to do a lot more to the tabular data.

3 Problem Formulation

ADR is driven by proving utility and efficacy of the solution as opposed to forming hypothesis and proving facts or truths. Innovative thinking along with creativity is largely motivation behind ADR researchers. The novelty of the solution is similarly influential to them. The first stage of ADR is problem formulation which requires understanding existing system and practices. Furthermore, the existing practice is a valuable starting point to explore design of an artifact that is not yet there. We begin with a thorough discussion together with IS researchers to find and formulate the research problem. The next step consisted of researching and testing data export functionality in different tools to get a better understanding for the environment, data export methods, technologies and tools that existed. We tried several ways for exporting data and tools (both stand-alone desktop applications and, mainly, web browser applications). This chapter start with problem setting in U-CARE case organization and describe U-CARE platform in detail. Then it explains the research problem and at the end design process for data export artifacts design.

3.1 Problem Setting

Design science s about understanding, explaining, and improving information systems. Understanding leads to knowledge for foreseeing how various aspects of a phenomenon performs. Design uses this knowledge plus innovation to create new enhanced artifacts that go beyond what was available beforehand. The case study domain used in this thesis is clinical research or more specifically Internet based self-help program for psychosocial care (support, help and psychological treatment using CBT). This field has provided context for our work as well as provided the organizational feedback needed to conduct Action Design Research. The case study and organizational participation have provided vital data and feedback during the process.

3.1.1 U-CARE Platform Development Process

The development of U-CARE platform was conducted in close collaboration between Information Systems (IS) researchers and other stakeholders in the research context. IS researchers also acted as system analysts and software developers. The development process for the IS researchers was part of design-oriented research, meaning that both theory and practice informed the development. The theoretical basis was drawn primarily from psychosocial care, software engineering, interaction design, and information systems (Sjöström, et al., 2011). The development followed a selection of agile development principles (Beck, et al., 2001) - such as short iterations and test-driven development (TDD) - to be more flexible and better suited to handle changing requirements during the development process (Abrahamsson, et al., 2009). Development sprints lasted for 2–3 weeks and each sprit followed by a sprint review. The ‘customers’, various stakeholders (mainly researchers from other disciplines), were exposed to the most current version of the platform during the sprint review. These review meetings provided feedback and directed the continued development efforts. The feedback was documented, email correspondence has been kept and field notes were taken. Continuous assessment of the platform allows gradually refinement of ideas and design principles. The iterative and stakeholder-centric (Sjöström, 2010) approach is used with an emphasis on value creation. U-CARE platform, as a result, means to contribute to the primary goals of the U-CARE programme.

3.1.2 U-CARE Platform

U-CARE platform is large, complex system with large and complex user requirements and data sets. It is a self-help programme with rich contents (text, pdf, narrative, image, movie and audio), interactive communications (chat, forum and diary) and feedback (quiz, homework and questionnaire). To be successful, being an agile software development project, it need flexible development tools and environments able to cope with the required pace of change. On the other hand object orientation is, the dominant paradigm in software development, a means of bundling functions and data structures in to models that resemble real world objects. Object oriented programming is also good for large projects that require manageability. High system maintainability is the key to project success which lies in anticipating new requirements and changes to existing requirements, and in designing the system so that it can evolve accordingly. To design the system so that it's robust to changes, designer must consider how the system might need to change over its lifetime. Design patterns help us avoid this by ensuring that a system can change in specific ways. Design patterns make it easier to reuse successful designs and architectures. Further they help a designer get a design "right" faster.

U-CARE platform is web-based application developed from ground up using Microsoft technologies. U-CARE platform is set up in a typical three-tier architecture using Microsoft IIS as the application server, Microsoft SQL Server as the backend database server and a browser client to access application. Data is encrypted and transferred via secure connection. The advantage of the solution is its portability. The middleware provides the system with necessary dynamics to transfer information to the databases and allow user interaction. U-CARE platform is dynamic application. Dynamic refer to the process of generating user interface, like web-based questionnaire, from a database of questions, possible responses, and controls at the time of connection by the client browser using an application running on the web server. This method is in contrast to the static method of pre-designing the questionnaire as an HTML web form and placing it on the server so that it can be loaded and displayed through the client browser at the time of connection. Web-based questionnaire using a dynamic approach proved to be a very efficient and effective data collection system (Cooper, et al., 2006). Relational model for metadata used to simplify the process of maintaining consistency during metadata creation and editing. U-CARE metadata-driven design evolves as needs grow and it is not hard to add new capabilities to the application.

ASP.NET MVC framework is used to develop application using C# programming language. The ASP.NET MVC framework dictates the architecture of U-CARE platform. ASP.NET MVC is part of ASP.NET framework. The Application interacts with database using run-time infrastructure LINQ to SQL for managing relational data as objects. Developer use Microsoft Visual Studio 2010 as development environment and Subversion as their source code control system. U-CARE platform adopt Active Record pattern to get LINQ-to-SQL object model using Visual Studio's Object Relational Designer to generate the classes for data model automatically. Together, the LINQ to SQL run-time infrastructure and design-time tools not only significantly reduce the workload for the U-CARE application developers but also support rapid development as required by agile software development method.

Interfaces to LINQ entities make U-CARE platform more maintainable by removing direct dependency on LINQ entities and also separate it from data access layer (DAL) layer.

3.1.3 U-CARE Platform (Database) Design

U-CARE platform takes care of almost everything related to data on the application layer though business logic. The application layer enforces entity, domain and referential integrity ensuring the quality of the data in the database that means data entered into the database is accurate, valid, and consistent. The data validation rules are satisfied before permitting a change to the database. In this way performance is improved because there is no double validation happens on the database model especially for the big data centric application like U-CARE. The other benefits in doing this is efficient error handling because the application know what and where the error was.

The complexity and the rapid evolution and expansion of the domain of clinical research make development and maintenance of clinical research application difficult. In a conventional relational database system, the physical design of the database must be changed whenever new data types are introduced or existing types are modified. For this reason, U-CARE database is flexible that allow modifications and addition of new types of data without having to substantial change in the physical database schema. The U-CARE platform implements a highly-detailed logical database schema in a completely-generic physical schema that stores the wide variety of clinical data in a small and constant number of tables. This made system a generic (and configurable) platform applicable in new situations in without further design efforts other than user configurations.

As development followed agile principles, changes are continuously adopted in the system and system refactoring is force the change in the models as well as in databases at least in some cases or in part of the system design. New fields are added to database and default value is set to zero (0), null (\emptyset) or empty string (''). In this way the system take care of the new filed and set it with proper value as required by business logic but the old data already entered remain as default value. Whenever changes are made the business logic takes care of the old data and new data. U-CARE platform designed to work with logical deletion that means whenever user want to delete a record it there is a filed "isDeleted" is set to false. To adopt changes the system uses the surrogate key instead primary key (and/or composite primary key). The logical schema defines the foreign keys but physical schema doesn't define them in the database. If we look at the schema we can't see foreign key fields mapped to primary key fields in other tables. It's all just enforced by application business login in the code. Data consistency and integrity handled by the application.

By mentioning above points our goal is not to perform a critical evaluation of the quality of the platform but rather to explain the system to understand problem scenarios presented in next section. The pros and cons of developing application in such way are not in the scope of this thesis but above detailed inside of U-CARE platform and knowledge base relevant to current problem was presented in previous chapter for the understanding of the readers.

3.2 Research Problem

IS Researchers in U-CARE initiated the current research by foreseeing the need of data export functionality as ultimately the clinical researchers required to export data, that is accumulated over time through U-CARE platform, to data analysis tools for research purposes. Most features of U-CARE platform are designed using flexible database structure to store data *in* and *about* studies (i.e. data as well as metadata) which provide the generic and flexible application but for data export it become a challenge as well. To extract data in this environment takes an incredible amount of time and is fraught with errors/defects. The practitioners used the SQL queries or DMBS data export functionality to export data directly from database for data analysis whereas the clinical researchers have no or little knowledge of SQL and DBMS. Developers required an efficient and generic solution to export data, for data analysis in statistical tools, with less effort. The literature review suggests that data extraction, data migration, data warehousing, data mining and data analysis fields are well explored as well as solutions and challenges for related problems are presented but data export is still much unexplored which create a unique research opportunity and importance of current research. So the fundamental challenge was to find the easy, effective and efficient (generic) way to export data that satisfy all stakeholders involved in the project. Furthermore, from the research perspective, we wanted to consider such functionality's design principles that would be applicable to a class of similar problems. Therefore, my research questions were:

➤ **RQ1: Principles to guide data export design?**

What are the current best practices and principles to *design* generic data export functionality, in a data-centric clinical research application, in line with existing information system design?

➤ **RQ2: What are the implications of RQ1 for software design?**

How can we *develop* generic data export functionality, in a data-centric clinical research application, by not only by following best practice and principles but also interacting with organizational context?

The study was commenced by interviewing IS researchers, practitioners and clinical researchers during a first two month to identify problems (and/or challenges) regarding data export functionality in U-CARE platform. Furthermore the existing system was studied in details. Table 6 below summarizes identified problems. However, not all problems can be foreseen through the initial problem formulation. The remaining problems and challenges were identified through BIE cycles (presented in subsequent chapters) during the development and use of the system.

Table 6: Problems and challenges regarding data export in U-CARE platform

Problem / Challenges	Description
i. Large and complex system	U-CARE platform is large, complex system with large and complex user requirements and data sets. As each work package, even individual study, have unique requirements.

Problem / Challenges	Description
ii. Generic and flexible system	Most features of the platform are designed using flexible database structure to store data <i>in</i> and <i>about</i> studies (i.e. data as well as metadata). The platform is designed in such a way that it adopts itself dynamically as well as provides generic solutions for designing questionnaire etc. This provides generic and flexible application but to extract data in this environment takes an incredible amount of time and is fraught with errors/defects.
iii. Dynamic application	It is a dynamic application, build for change and adaptable to business process context. Application logic and data is handled by models, and stored in database.
iv. Non-normalized database	There is no foreign key defined in the physical database design instead it is handled by models. Surrogate keys are used instead composite keys and logical deletion (without cascading) instead physical deletion.
v. Continuous changing data models	Changes are continuously adopted in the system and system refactoring force the change in the business models, data models and database schema. New fields are added to database and default value is set to zero (0), null (∅) or empty string (''). In this way the system take care of the new filed and set it with proper value as required by business logic but the old data already entered remain as default value. Whenever changes are made the business logic takes care of the old data and new data.
vi. Difficulties in understanding data	Over the time there is huge data is stored in the database and without using knowledge stored in business logic it is hard to understand the data.

3.3 Class of Problems

Healthcare technology, practice and knowledge are continuously evolving at a rapid pace. So the applications design and developed during any health care research project need to adopt itself continuously. A data-centric clinical research application, such as U-CARE, takes care of many research studies and clinical trials. Typically, the database is designed to manage data related to research studies. Data gathers by these health care application need to be interoperable between different application. Data interoperability and integration is one of the biggest issues in healthcare sector. There is always need of some sort of data export functionality to transfer data from, either application or directly from DBMS, to data analysis tools. The data-centric clinical research applications have mostly dynamic, flexible and generic design. The data export from these applications takes an incredible amount of time and is fraught with errors/defects. The problems mentioned in previous section about U-CARE platform can be taken as an instance class of problems faced during data export in any clinical research application.

4 BIE Cycle I

Practitioners with first-hand experience of the problem domain, organizational setting, existing system and projected use of data export functionality have opportunities to influence the design throughout this stage. Prototype construction was started by exploring the strengths and the weaknesses of the technologies and procedures upon which the current system had developed so far. During first cycle of IT-dominant BIE initial design of alpha version artifact is lightweight intervention in a limited organizational context. Based on the findings from the first stage (formulating the problem) we identified design solutions (written as bold and italic) for constructing first prototype using knowledgebase with input from the researchers and practitioners. Design solutions pose challenges while developing the prototype which brings us at the start of design solution identification cycle.

4.1 Constructing the Prototype (Alpha)

4.1.1 Large and Complex System (Table 6-i)

There is complexity in requirements, as well as within the U-CARE platform itself (). The system has work packages and each work package has related studies. Each study consists of study group and study members. Each study group have participants. Study participants and study members have a user account to login to the system. Work packages, study, study group and study members are part of study design area. Each study has some observation points. Each observation point has some items and/or interventions. An intervention has one or more items grouped as modules, steps and folders. An item can be different types, simple types (Image, Video, Rest, Text, Pdf, Audio, and Narratives) or complex types (Quiz, Homework, and EQ-5D). ***It was decided to build the new prototype step by step, starting with exporting partial data related to study (only items type quiz).*** Figure 15 shows partial ERD of the U-CARE platform database. Quiz is a sort of questionnaire. Each quiz has a set of questions. Each question has an answer choice(s) of either single value (input using radio, select, text or text area) or multiple values (input using checkbox, text or text area). Even each question has different style to present the choices like for radio type choices (vertical, horizontal, complex, ladder, select, scale), for checkbox type choices (vertical, horizontal), and for question answer type choices (text area, text, percentage). A quiz item may have a number of expressions consists of questions IDs and athematic symbols. Expression result is calculated based on the answers of the questions those are part of the expression. Observation points, items (quiz), questions, expressions and question choices are part of data collection design. The next area, data collection results, is the results of these items which includes user's question answers and expression results. Further the question results depend on the user who answers the quiz item. It is possible to have more than one set of answers for a quiz due to multiple entries by different users (participant himself, health staff or relatives etc.). The quiz allows different participants to receive a different set of questions based on skip condition. The participant answered one question in a specific way and therefore skips to a designated question further down the quiz. It is also possible for participants not to answer any question if they don't want to. This lead to missing values either due to skip condition or participants do not complete a quiz.

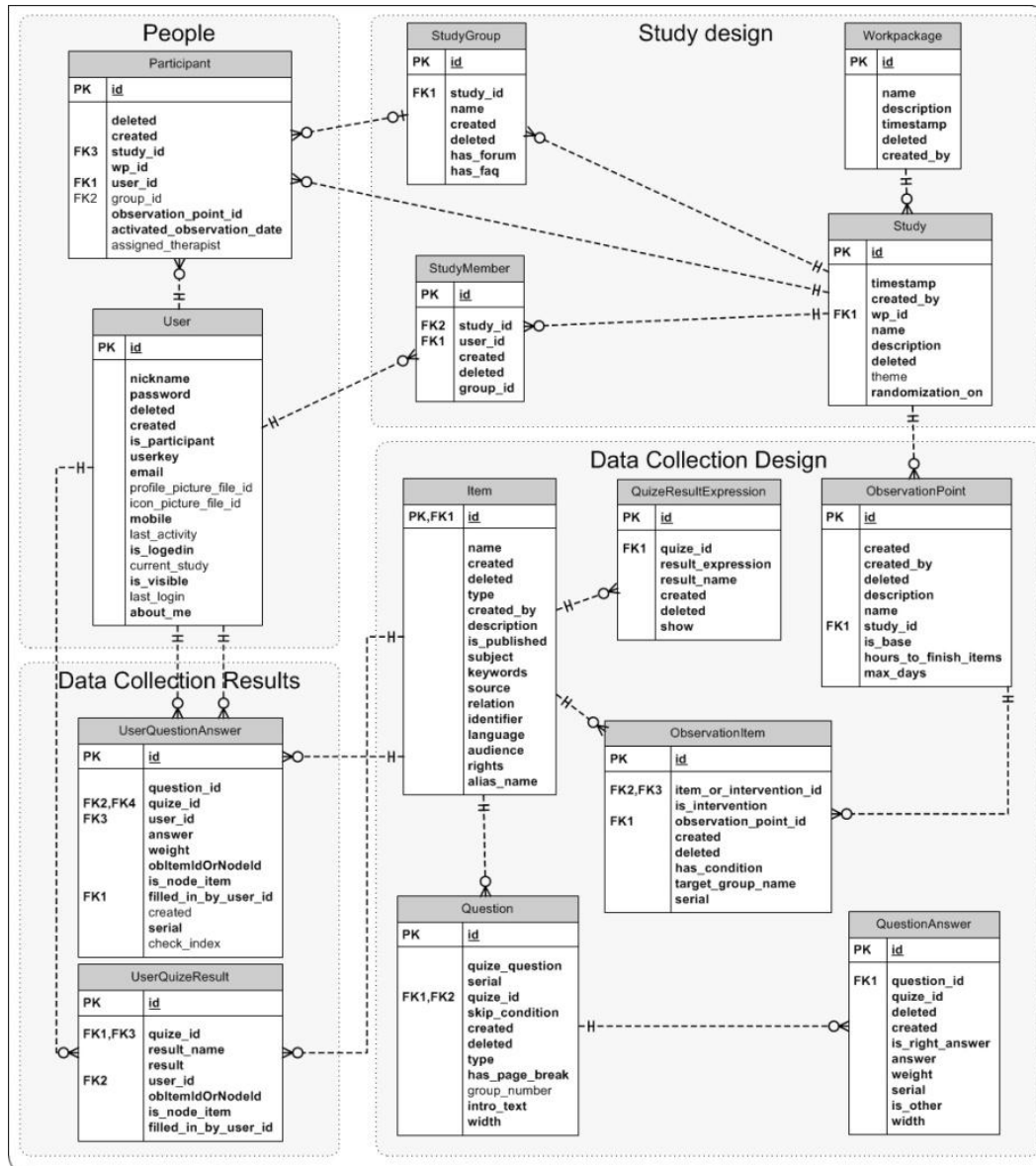


Figure 15: U-CARE platform partial ERD

4.1.2 Generic and Flexible System (Table 6-ii)

Most features of U-CARE platform are designed using flexible database structure to store data in and about studies (i.e. data as well as metadata). The platform is designed in such a way that it adopts itself dynamically as well as provides generic solutions for designing items like quiz. Static and specific (non-generic) design is used when total no of questions given are fix; the users won't add any more question later on once the questionnaire is design and modelled in the physical database. Furthermore each questionnaire has its own table in the database and new questionnaire addition required creation of new table which results application modification regarding GUI and business logic. Static solution is better in case we know total no of questionnaires and the no of question in each questionnaire throughout application's entire lifetime. Application development is easy and fast but modification are

difficult and slow. Data entry required updating single row for each question's answer by a single user else otherwise results are stored temporarily and inserted once user finally submit the questionnaire.

Generic and flexible system design is used when total no of questions given are not know. Questions can be added to questionnaire (in our case it is called quiz) at any later stage but before allowing the user to answer questions. The generic design is difficult but in this way same application can be used for any number of questionnaires with any number of questions. Furthermore it delivers this functionality without needing to continually add new tables to the database schema. It is also better with respect to performance because it provided insertion for each question's answer as compared to updation in static design. In spite usefulness the generic and flexible design pose challenges for data export and extracting data in this environment takes an incredible amount of time and we also have to convert the rows into columns for data analysis. U-CARE platform allow researchers to design the questionnaire themselves by creating a quiz and then adding questions in it without any involvement of developers' end.

Initial design solution was to export data directly from database. We tried data export wizard, SQL (Structured Query Language) statement and stored procedure to export data. Major challenge was converting hierarchy of tables into one linear table and rows of each table into columns as most analytical programs (such as statistics packages) require the data in the one column per parameter format. The end-users (clinical researchers) need answers of all questions of a quiz, all quizzes in an observational point, and all observational points in a study in one row per participant. The SQL statements required to operate on conventional data are different from those needed to operate on generic and dynamic data to return equivalent results. Stored procedure solved this problem to some extend but complicates the task of query data significantly. Furthermore business logic implementation shifted from application layer to data layer and it also not fulfills the design aspects like extensibility, fault-tolerance, maintainability, modularity, re-usability, and robustness. ***An improved design solution was to build system with LINQ to SQL Meta Model API using Data Context.*** Meta model allows us to get list of tables, their columns and a lot more information present in our data context. We developed a data export module within the U-CARE platform. This module used the Meta model API to list tables and their columns at the front end (presentation layer) to generate a generic SQL statement for data export. But the design fails as there is lack foreign key relationship information in U-CARE data context. Adding this relationship information again and again for each data export was not optimal solution. ***The final design solution was to develop the data export module using models (, the business objects).*** The design solution was based on and aligns with a number of related research studies by Bennett & Bayrak (2011), Morris, et al. (2008), Bordbar, et al. (2005) and Brandt, et al. (2002).

4.1.3 Dynamic Application (Table 6-iii)

U-CARE platform is a dynamic application, build for change and adaptable to clinical research process. Application logic and data is handled by models, and stored in database. Dynamic is taken in the sense that user can add any no of observation points in a study, quizzes in an observation point and questions in a quiz. The data export module has to handle the study and its contents (study groups, study members, observation points, quizzes, questions etc.) that

are added by the researchers dynamically. *It was decided that researchers should be able to configure the data export themselves.*

4.1.4 Non-normalized Database (Table 6Table 6-iv)

There is no foreign key defined in the physical database design instead it is handled by models. Surrogate keys are used instead composite keys and logical deletion (without cascading) instead physical deletion. Surrogate keys and logical deletion is also handles by the models. *The non-normalized database further reinforces our decision to use models to develop the data export module.*

4.1.5 Continuous Changing Data Models (Table 6-v)

The development followed the Agile development principles. In the Agile development process, applications are built with the goal - get it done quickly, don't worry about doing it right. The relationship between Agile and architecture appears mutually exclusive at least in some cases or in part of the design. Fulfilling the requirements of a particular iteration is paramount. Changes are continuously adopted in the system and system refactoring force the change in the business models, data models and database schema. The database is refactored to support the new needs of researchers and/or to improve application design without changing its semantics. New fields are added to database and default value is set to zero (0), null (∅) or empty string (''). In this way the system take care of the new filed and set it with proper value as required by business logic but the old data already entered remain as default value. Whenever changes are made the business logic takes care of the old data and new data. In other words the business models remain always up-to-date and able to handle logic related to data. *The continuous changing data models also strengthen our decision to use models to develop the data export module.*

The Agile software development primarily focuses on satisfying the customer by delivering working software quickly with minimum features and then improvising on it based on the feedback. The delivery timelines are short and the new code is built on the previous one. To ensure that delivered product meets the end user's requirements it is important that adequate testing is done and all scenarios are tested. The testing style and approach in Agile is very different from traditional bureaucratic methods. The developers do the unit testing to ensure that the software component is functioning correctly. The development team tests if the newly added functionality works correctly and that the previously released functionality still works as expected. The core functionalities of U-CARE platform are well tested. *Working with models reduces the need of testing as models' functionality is already well test.*

4.1.6 Difficulties in Understanding Data (Table 6Table 6-vi)

Over the time there is huge data is stored in the database and without using knowledge stored in business logic it is hard to understand data. The logical data model is stable as compared to physical data model which is much more dynamic. Logical data model represents independence of our physical database, thus permitting mobility and portability to new database or data analysis tools. Integrating data for analysis is a big challenge while joining from single column, to single table, to cross-table. *Understanding knowledge embedded in business models is much easier than from database which is well understood by developers only.*

4.1.7 Exporting (Business) Models - Designing Data Objects

Development using MVC architecture required a set of model classes that manage the relationship between data and the way it is presented to the user. The separation of functionality introduced by this architecture gives developers greater flexibility. To export data related to single model require creating a single function in the model class that export the contents of the model. But when we need to export more than one model, or we have to integrate the data from more than one model, we use data transfer object (DTO) between business tier and presentation tier. DTO is also used to transfer data between two service layers (see 2.4.11 for details). On the other hand service data objects (SDO) is also used to transfer data between service layers (see 2.4.12 for details). In U-CARE platform there is a logical hierarchy (work packages => studies (study group, study members) => observation points => items => quizzes => questions => answer choice(s).....) of models. We required a container to hold this hierarchy and then process it for data export. ***After working with different possibilities it is decided to develop our own data structure, by integrating DTO and SDO concepts, a collection of object as “Data Objects”.***

The main purpose of the data objects (DO) is to get data from models of U-CARE platform and provide to data export module. The DO uses its own set of classes to hold the models. For example in U-CARE platform there is a model “Workpackage” and accessible by “IWorkpackage” interface. DO has “EWorkPackage” class that compose of an object IWorkpackage and List object (a strongly typed list/array of objects, a collection in .NET framework) EStudy. The same process repeat itself down the hierarchy to build a tree structures same way as it is a graph in SDO. The DO classes act as wrapper class around in the U-CARE interfaces and the data export classes themselves. There are two advantages using wrapper classes in this way to contain the models. One, it decouples the data export form U-CARE platform so changes within the models don’t affect the data export. Further changes in the hierarchy of data export wrapper classes will not affect the models. And second, no need to test these wrapper classes as they are composed of already tested existing models. Use of data object not only efficiently handle data but also provides an interface between the U-CARE application and data export module, an efficient application of Adopter and Façade patterns. Figure 16 depicts the DO tree structure. Data transfer object uses the assembler to build/ populate the DTO. If the DTO metadata is not known to other layer reflection is used for deserialization. An assembler can keep the domain model and the data transfer objects independent of each other. We used the DO’s constructors as assembler to initialize the DO’s data members. The overall data export module architecture and interaction between U-CARE models and the DO can be seen in Figure 19.

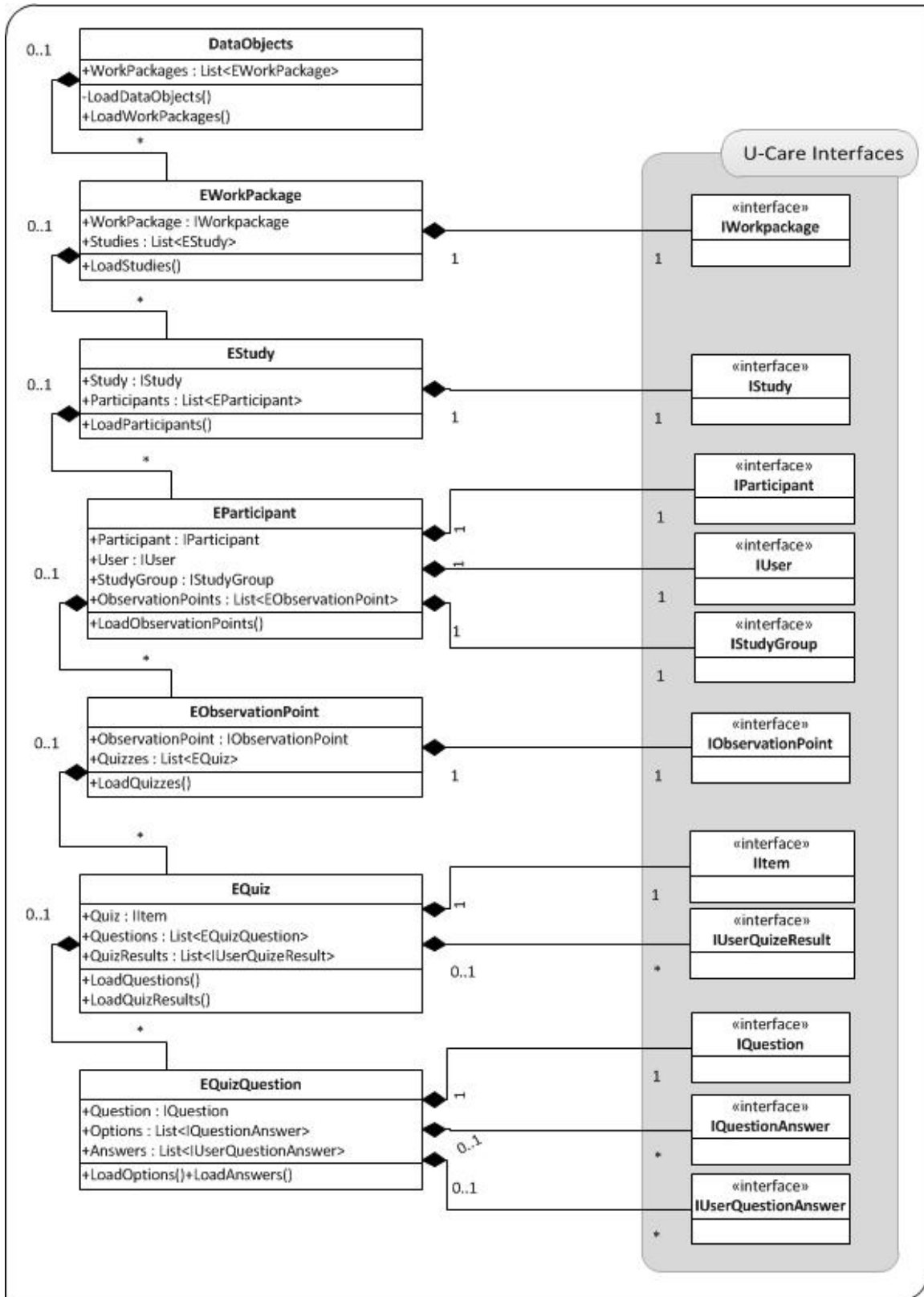


Figure 16: Data objects tree structure

4.1.8 Generic and Adaptable User Interface – Designing Reflection Utility

The data objects (DO), explained in above section, are adaptable to any set of models and can be design for different hierarchy of models. This mean the DO are not affected when the there is any change in the underneath structure the model like addition or deletion of a filed/property from associated table/class and the composition of the DO tree structure can be modified as well to meet the researchers' requirements. This ability made the design of the DO a generic solution. *It was decided that adaptability must extend to the data export module's user interface as well, requiring the interface adapt itself as soon as the data objects changed.* The reflection is used to develop generic and adoptable interface. *It was one of the requirements that user should be able to select models and their properties he want to export.* We developed a utility that uses reflection to take the DO class/object and provide the metadata to build the user interface for user selection. Figure 17 depicts the user interface for data export configuration.

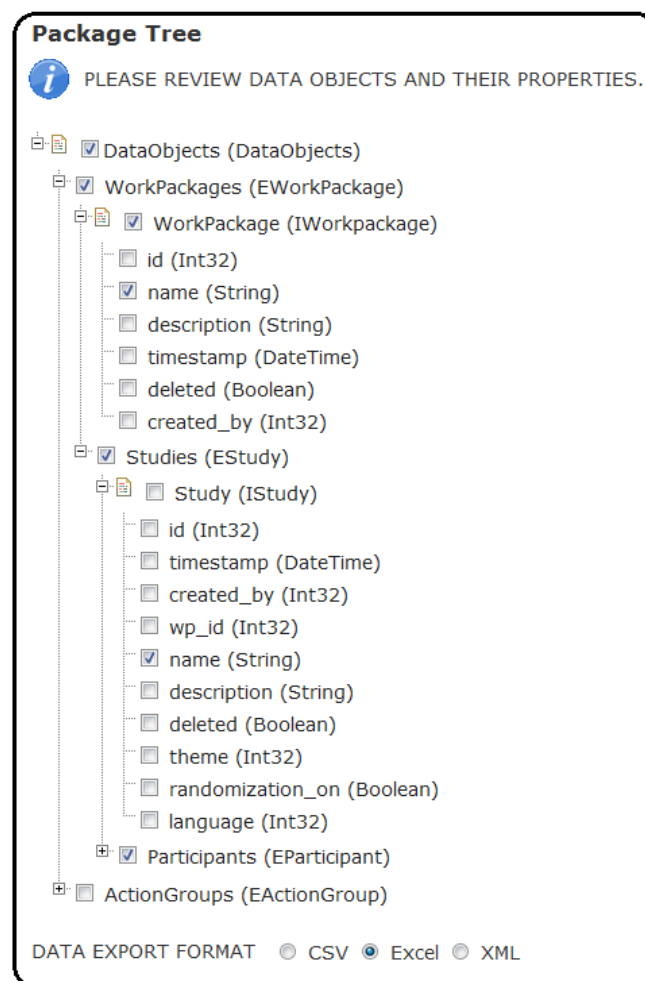


Figure 17: User interface for data export fields selection

The reflection utility gets all the information on a given DO and performs a recursive traversal of the metadata until every attribute retrieved is simple (atomic). The metadata is stored in the specially build data structure

MetaTreeNode. The metadata about the data object is presented in tree structure build dynamically using custom control and recursion. The idea is taken from U-CARE platform where the intervention is design in a tree structure format by researchers. The interface provides efficient way for user selection for hierarchal list of objects. Furthermore it expands and collapse according to user selection which provides better and compact view. The user selection is stored in the PackageTreeNode data structure which is an extension of MetaTreeNode. MetaTreeNode keeps all metadata regarding the data objects while PackageTreeNode only keep the user selected filed for data export. Each node of the tree can be expanded or collapsed by clicking on it, enabling the researcher to zoom in or out of the specific regions of interest. The ordering of items within a higher-level grouping can be inspected and altered and items can be added to (or removed from) specific places in the tree by changing the data object class and required recompile and publish the project. The order within the fields of model depends on the order of attributes in the database table. Figure 19 shows the ReflectionUtility, MetaTreeNode and PackageTreeNode and their link to data objects.

4.1.9 Pivoting (Row to Column Conversion) – Upgrading Reflection Utility

Researchers need to select data from the study and then packaged into a format suitable to their analysis software. They typically need a two dimensional data matrix and each row is a vector (x_1, \dots, x_n) . This process is also known as “flattening”. As U-CARE platform is dynamic and generic some of the model need conversion from rows to columns. This requires working on the data of the objects and converting a list of objects into one row for a particular participant. This process is known as “pivoting”. For example if we have N questions in a quiz then there are N rows in the database table and we have to convert N rows to one row with N columns (one column per question). ***For pivoting it is decided to upgrade reflection utility to work with data objects’ data within objects instead of data objects’ class.*** The modified reflection utility gets all the information on a given DO and performs a recursive traversal of the metadata as well as recursive traversal of the data (where it requires) until every attribute retrieved was simple (atomic).

The reflection process has two types of methods. One deals with metadata and the second deals with the data of the data objects. The first step is using reflection utility to get metadata and generate the GUI for user selection. The next step is to use saved user selection and data objects’ data to build a DataTable (.Net type to structure data into row and columns). This data table is further used to export data in different formats. Data table is also constructing in to steps. In first step columns of data table are defined and in second step data in inserted rows wise into each column. By default all rows are kept as rows but if checked (Rows2Columns) the module convert rows to columns. It provides user the opportunity to choose what he wants to put in rows and what into columns. The pivoting is applied only when there is one-to-many relation between parent and child nodes in the data objects tree structure (see Figure 18).

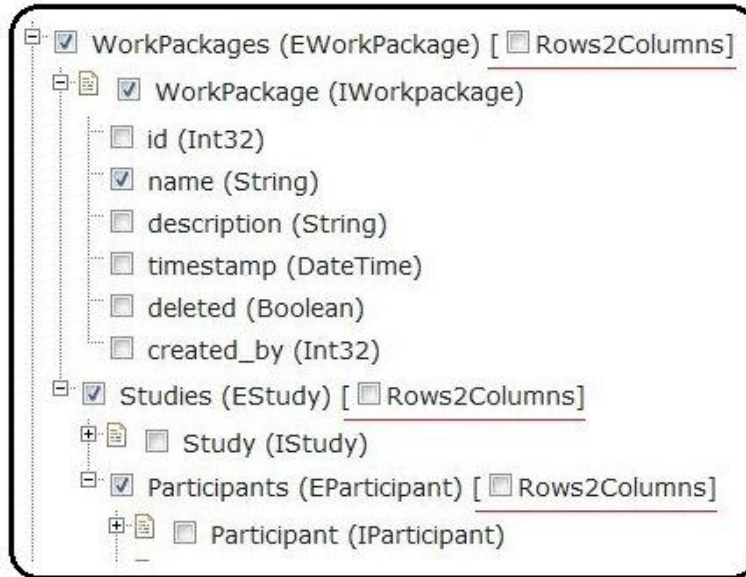


Figure 18: Modified user interface for pivoting selection

4.1.10 Render Useful Output – Designing Export Utility

The data analysis tools required a specific file format to import data. There are lot of open source and proprietary software applications file formats. Comma Separated Values (CSV) is a well-known and platform /applications independent format. Beside CSV XML and Spread sheet format is popular. ***It was decided to provide data export in CSV, XML and Excel format.*** Reflection utility discussed in previous section returns us the data in DataTable. Now we only need to convert it to different formats like CSV, XML, and Spread sheet. To accomplish this we develop an export utility. Export utility not only convert into specific format but also sort and clean data. Cleaning data is required to preserve the data format. For example in CSV we remove or encode the every “,” that is in the data. Figure 19 shows the ExportUtility relation to other components.

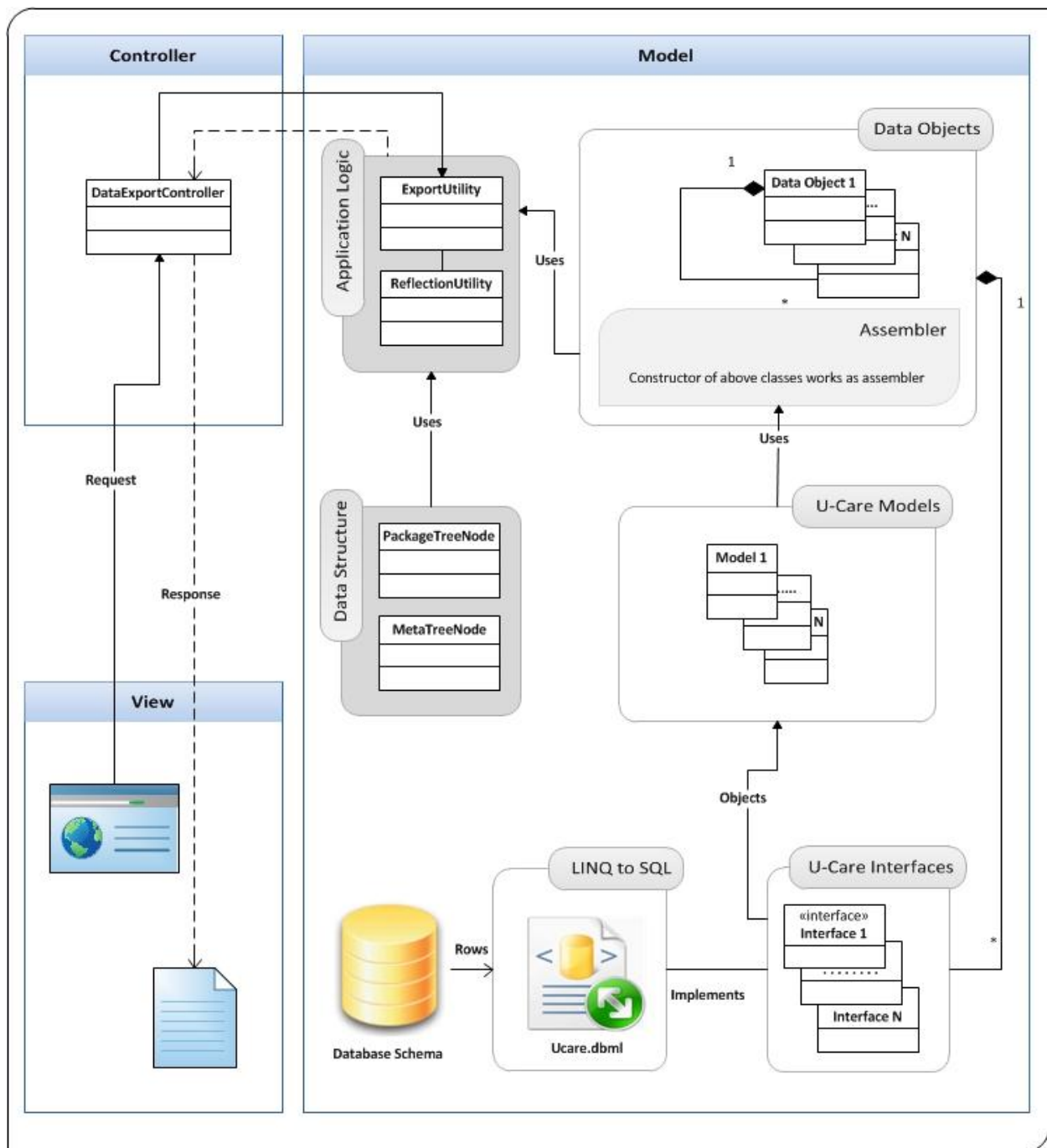


Figure 19: Data export module architecture (alpha)

4.2 Intervention and Evaluation with Practitioners

Following the ADR method we conducted continuous intervention and evaluation sessions during the prototype design process. In addition we followed the Agile principles: continuous delivery of useful software, frequently delivery working software, face-to-face conversation, continuous attention to technical excellence and good design, simplicity. The ADR BIE and Agile principles are well aligned with each other. During creating the first (alpha) version of prototype we continuously work on-site with the practitioners (the U-CARE platform development team). The intervention and evaluation sessions were performed as a part of daily meetings between researchers, practitioners, and domain experts of clinical research. Practitioners were involved in using, discussing and providing feedback on the prototype and its application in data export contexts. During the interventions decision related to prototype's design are taken. Most of the design decisions are mentioned in previous section. In this section our main focus is on formative evaluation of prototype. Venable, et al. (2012) while describing purpose of evaluation mentioned evaluation of alpha version in ADR is formative evaluation, "in which an artifact still under development is evaluated to determine areas for improvement and refinement". The prototype is designed, developed and executed in the case organizational environment which is itself an (case study) observational evaluation method (Hevner, et al., 2004, p. 86).

Right from the start the practitioners want the data export module should be generic so that they can fulfil the requirements of the end-users (researchers) with no or minimum development. The other requirement was the module should adopt and ensure the design aspects⁸ of U-CARE platform i.e. compatibility, extensibility, fault-tolerance, maintainability, modularity, reliability, re-usability, robustness, security, usability, etc. Furthermore they also want that the system should be reliable in such a way that minimum test should require to maintain the module. Throughout the intervention and evaluation the alpha version fit well with the expectations of the practitioners. During the construction of prototype whenever the developers modified the U-CARE platform models, e.g. add/delete/modify any filed in the model's associate database table, the data export module adopt itself including user interface by running reflection utility. Furthermore there is no need of testing the data export module as data objects wrapper classes are just composition these models and the modified models are tested by the developers at their end. In this way developers time saved and can be utilized in the development of the core functionalities of the system. This process was tested again and again during alpha prototype and also benefited the data export developers to ensure the reliability of the system. The testing is part of evaluation process (Hevner, et al., 2004, p. 86). Both functional (black box)⁹ and structural (white box)¹⁰ testing performed throughout the construction of prototype.

⁸ Definitions of the terms are defined in appendix A.

⁹ Execution of artifact interfaces to discover failures and identify defects (Hevner, et al., 2004, p. 86)

¹⁰ Performance of coverage testing of some metric in the artifact implementation (Hevner, et al., 2004, p. 86)

The addition of new models was very rare and when it was required mostly due to refactoring purposes only. The addition of new models required change in the data object classes and changes were applied rapidly with minimum development. During the evaluation developers suggested performance improvements like user selection should be saved and during subsequent data export the same selection of fields or small variation can be exported efficiently with less clicks. The other suggestion was the metadata should be generated only if the underneath models or data object tree structure is changed. Developers also suggest storing data export data structures in database instead in the session. The main outcome of the evaluation was that *reflection should not be used every time data export module loads and user selection should be saved in database as a package so it can be reuse at later stage.*

4.3 Identifying Design Principles

The emerging artifact, as well as the theories ingrained in it, are continuously instantiated and repeatedly tested through organizational intervention and subjected to participating members' assumptions, expectations, and knowledge. This highly participatory process builds organizational commitment and guides the eventual design of the ensemble artifact. In order to answer to the last stage in the ADR process, formalisation of learning, we continued to merge my findings at the end of every BIE cycle with the knowledge base in order to create design principles. We generalized the findings in the light of the research framework which resulted in these principles, meant to provide guidance for further research and design work on data export modules, outside the clinical research domain. Considering the feedback from our sessions we continued by discussing and drawing suggested design solutions to answer challenges and problems. We based this process on the experiences gathered through prototype construction, intervention and evaluation sessions. The design principles are based on suggested design solutions during the ADR stage 'reflecting and learning' and in accordance with the principle of guided emergence. Table 7 describes a set of design principles that were derived from our findings, during BIE first cycle, to address the problems put forward in this thesis. The Design Principles are presented by title, description and the findings they relate to and were derived from.

Table 7: Design principles (version 1)

Design Principle	Description
i. User-friendly	Data export should be easy to use.
ii. Simple	Data export features should be designed for end users (researchers) and should not require much technical knowledge.
iii. Mutable	Data export should adopt itself to the changes in the research application/tool due to change in end-user's requirements.
iv. Render useful output	Data export should render useful output that is importable by data analysis / statistical application.

Design Principle	Description
v. Require least resources	Data export should require less development/implementation resources. In other words developer should not have to spend much time developing data export functionality. They should have more focus on developing application for core business functionalities.
vi. Generic	Data export should not require development on every data export request. Strictly speaking system should handle dynamically nearly all possible present and future data export requirements.

5 BIE Cycle II

During second cycle of BIE ADR team build a more mature artifact (beta version) based on the initial interactions and takes into wider organizational settings. This phase allows for a comprehensive intervention that includes evaluating the artifact in the use setting. The objective of this extensive evaluation is the on-going refinement of the artifact as it is shaped and reshaped by the use context. This stage of intervention may result in the parting of the researchers or trigger a new BIE cycle (Sein, et al., 2011). We divided our beta BIE cycle into two small BIE cycles, BIE II (beta v1) and BIE III (beta v2). Prototype (beta v1) was lightweight intervention in a limited organizational context whereas the prototype (beta v2) was a comprehensive intervention in a wider organizational context. Prototype (beta v1) construction was started by exploring design solution for the problem identified during intervention and evaluation of first cycle.

5.1 Constructing the Prototype (Beta v1)

After the first prototype, the intervention and evaluation thereof we continued with further development. Based on the attained feedback from the meeting with the practitioners we continued the development with sketching, designing and implementing additional features. The feedback was about the performance and graphical user interface. The second prototype was an evolution with enhanced features with respect to performance.

5.1.1 Reflection on Every Export Request – Designing Persistent Metadata Structure

In first prototype the data objects' metadata mapping information is generated through reflection on each data export request and held in the session. This leads to a performance issue as well as misuse of resources. ***So it is decided to develop the persistent metadata structures.*** We designed a database tables to hold the data objects' metadata information (see ExportMetaData and ExportMetaTree tables in Figure 20). By following MVC architecture model for these tables also created. MetaTreeNode data structure and MetaDataModel work together to handle all request related to data objects' metadata. Furthermore a new user interface created for the developers so that whenever they change in the existing models of U-CARE platform they can generate the metadata with new version using reflection. In this way reflection for metadata generation is only happen as and when it is required.

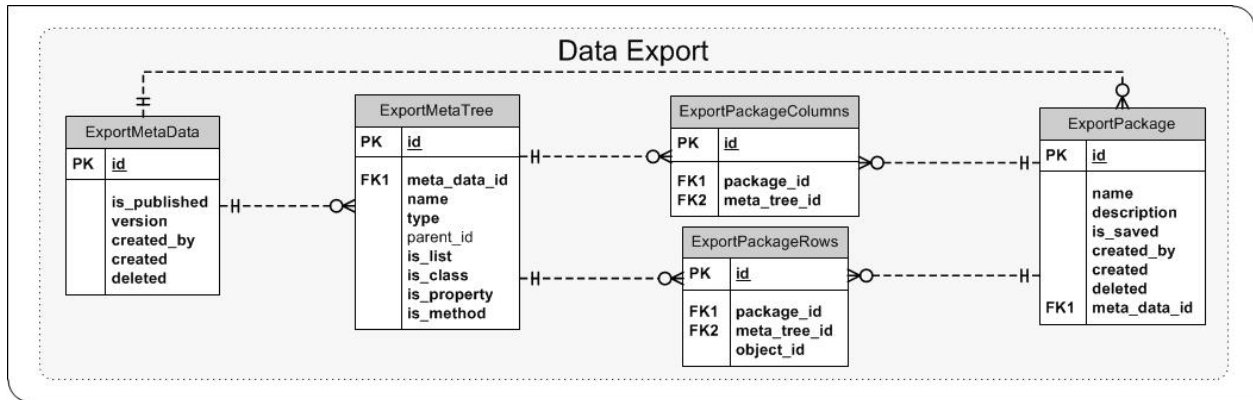
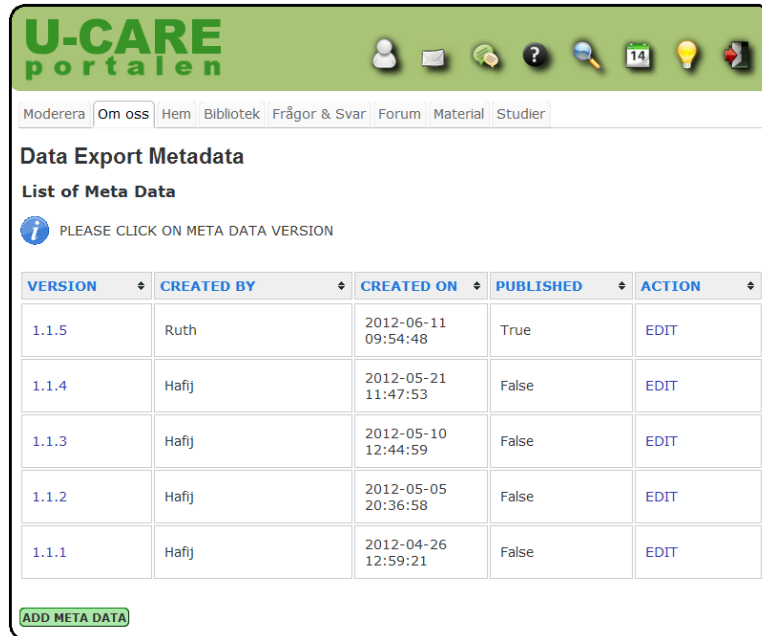


Figure 20: Data export ERD

Figure 21 shows how developer can see different versions of metadata generated as well as active one. Developers can new metadata with is not active by default. After verification of the new data objects tree structure (see Figure 22) they can activate the metadata for subsequent request of data export (see Figure 23).



U-CARE portalen

Moderera Om oss Hem Bibliotek Frågor & Svar Forum Material Studier

Data Export Metadata

List of Meta Data

PLEASE CLICK ON META DATA VERSION

VERSION	CREATED BY	CREATED ON	PUBLISHED	ACTION
1.1.5	Ruth	2012-06-11 09:54:48	True	EDIT
1.1.4	Hafij	2012-05-21 11:47:53	False	EDIT
1.1.3	Hafij	2012-05-10 12:44:59	False	EDIT
1.1.2	Hafij	2012-05-05 20:36:58	False	EDIT
1.1.1	Hafij	2012-04-26 12:59:21	False	EDIT

[ADD META DATA](#)

Figure 21: User interface for data export metadata list

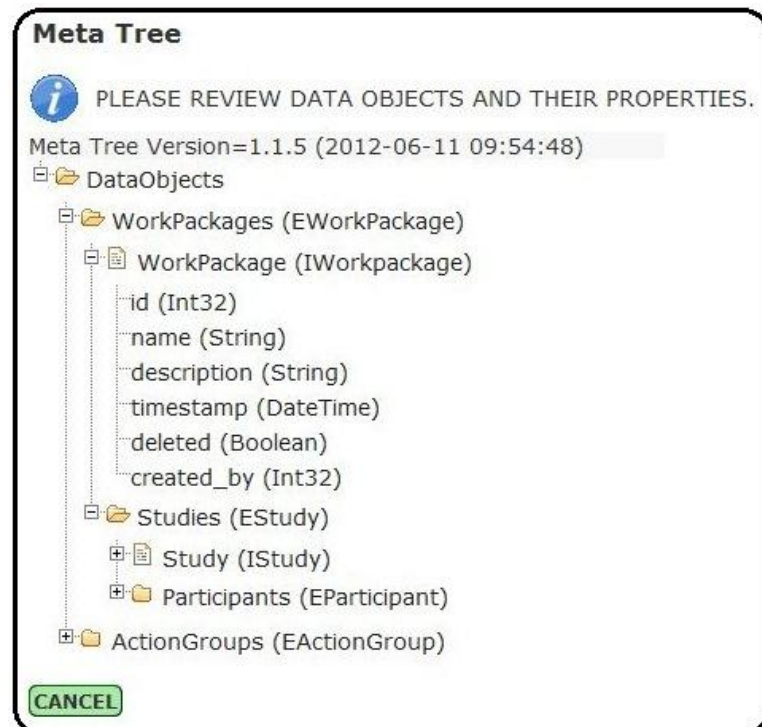


Figure 22: Metadata tree structure for developer's verification

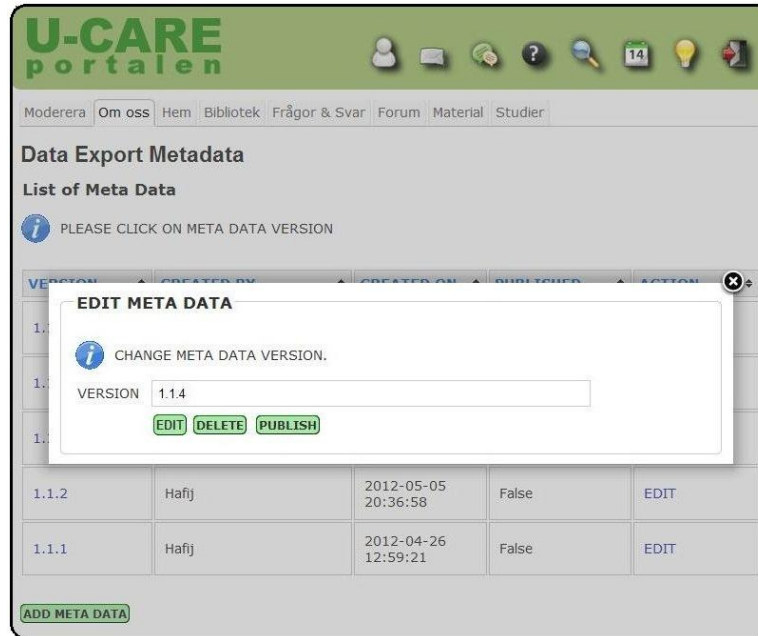


Figure 23: User interface to edit metadata

The data objects' metadata information held in the database improved the performance while exporting data. As metadata changes less often but once it is changed we parse it through reflection and then keep it in database. This improves the performance during or subsequent access.

5.1.2 Reuse of Data Export Packages – Designing Persistent Package Structures

In first prototype the user selection is saved in the PackageTreeNode and saved and kept in the session. Once data is exported into any one of export format the user selection is destroyed. If user wants to export same set of selection once again he has to perform the task once again and go through whole tree structure. Other more likely senior is that user wants to modify his previous selection to export different set of selection. During the intervention it was suggested to have possibilities to see the previous selection in both cases that are mentioned earlier. ***It is decided to develop the persistent package structures.*** We designed a database tables to hold the data export package information (see ExportPackage and ExportPackageColumns tables in Figure 20). By following MVC architecture models for these tables also created. PackageTreeNode data structure and PackageModel work together to handle all request related to data export package. Furthermore a new user interface created for the users so that they can create packages for each data export (see Figure 24). Once package is created the user selection is saved. After exporting data the user can export same selected data form saved package again at any time until the metadata related to package is not expired. If user wants to change set of selection he can modify and save same package with minimum clicks. The user interface is also modified with the use of AJAX (Asynchronous JavaScript and XML) and jQuery (www.jquery.com) for dynamic forms and handling of events. User selection is saved incidentally by Ajax to database. User can also work without creating package first. In this case there is a temporary packages created. After exporting data user can save the package if he wants.

List of Packages

PLEASE CLICK ON PACKAGE NAME

NAME	DESCRIPTION	CREATED BY	CREATED ON	SAVED	ACTION
testting erk	package	Ruth	06-12-2012	True	EDIT
my package	sdgsdgsdg	Ruth	06-11-2012	True	EDIT
1.1.5	New Package	Ruth	06-11-2012	True	EDIT

CANCEL
ADD PACKAGE
NEXT

Figure 24: User interface to list data export packages

The user interface for data export fields selection is modified and now it provide information about the package saved data as well as meta tree details (see Figure 25). Figure 26 depicts overall data export architecture of data export after applying modification.

Package Tree: my package

(2012-06-11 10:26:54)

PLEASE REVIEW DATA OBJECTS AND THEIR PROPERTIES.

Meta Tree - V1.1.5 (2012-06-11 09:54:48)

- ☒ DataObjects (DataObjects)
 - ☒ WorkPackages (EWorkPackage) [☐ Rows2Columns]
 - ☒ WorkPackage (IWorkpackage)
 - ☐ id (Int32)
 - ☒ name (String)
 - ☐ description (String)
 - ☐ timestamp (DateTime)
 - ☐ deleted (Boolean)
 - ☐ created_by (Int32)
 - ☒ Studies (EStudy) [☐ Rows2Columns]
 - ☐ Study (IStudy)
 - ☒ Participants (EParticipant) [☐ Rows2Columns]
 - ☐ Participant (IParticipant)
 - ☒ User (IUser)
 - ☐ StudyGroup (IStudyGroup)
 - ☒ ObservationPoints (EObservationPoint) [☐ Rows2Columns]
 - ☐ ActionGroups (EActionGroup)

DATA EXPORT FORMAT ☐ CSV ☒ Excel ☐ XML

Rows2Column: Folder means it is list of values. These list of values or rows can be converted to columns if checked

CANCEL
EXPORT

Figure 25: Modified user interface for data export fields selection using packages

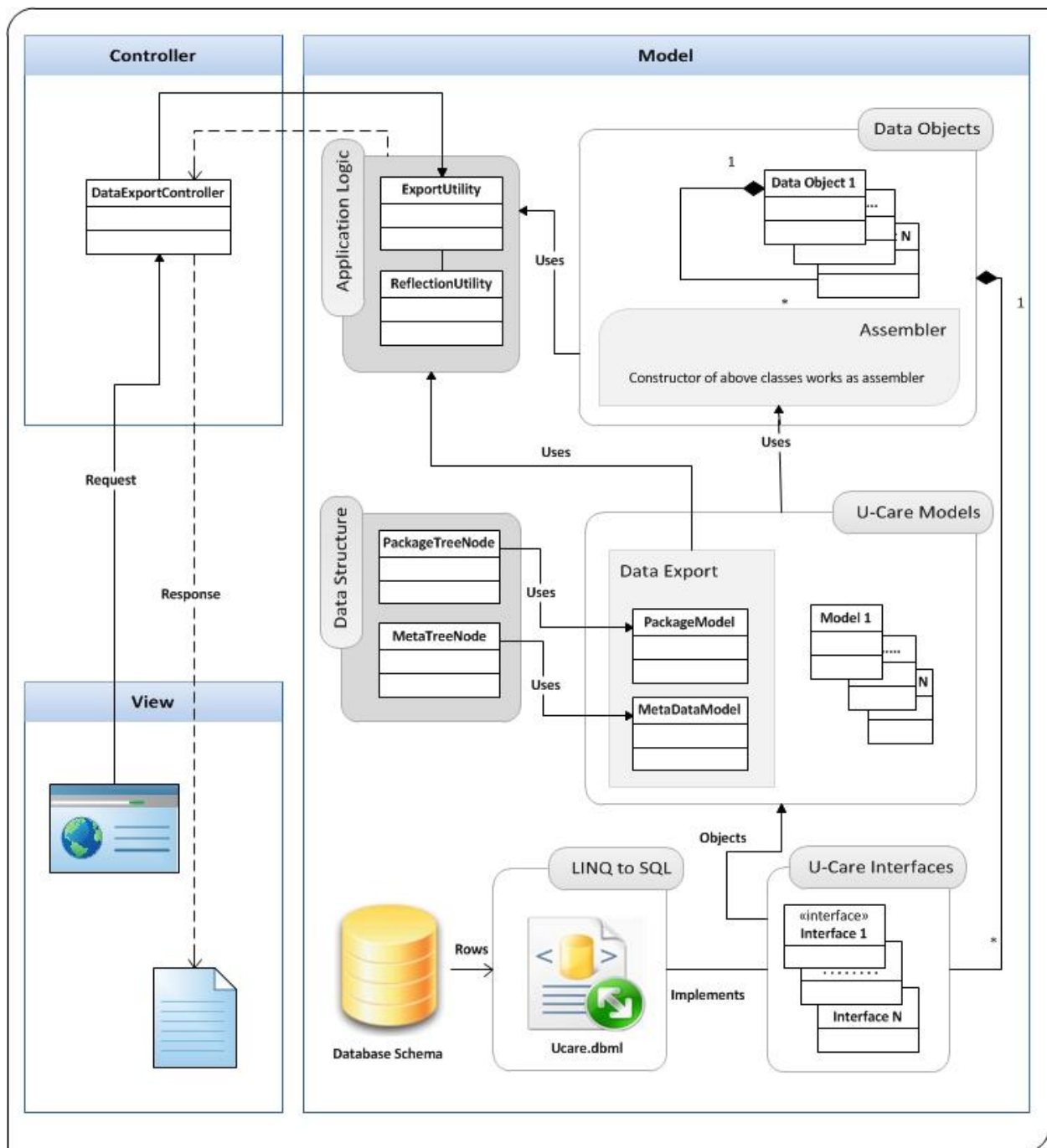


Figure 26: Data export module architecture (beta v1)

5.2 Intervention and Evaluation with End-users

The second prototype was almost complete with all requirements put forward by the practitioners as well as by the end users so far. The alpha version was only accessible to the practitioners and was available on test/local server. After completing the beta version prototype was integrated with U-CARE platform and published on production server. But the access of the data export module was given to developers only. At that time there was no research study was going on at production server. Mostly data was available with test users and for testing the platform modules and functionalities. In second cycle of BIE we conducted one intervention and one evaluation session. Intervention session was conducted during the sprint meeting of U-CARE platform where researchers (end-users) of all related domains were present including management. It is also worth mentioning that meeting participants were not only belonging to different professional groups but also related to different research studies (those are part of U-CARE work packages). Data export module demo was presented and different design aspects explained using a test study data. The module development effort was highly appreciated by everyone. Different questions were asked about access and use of the module. The feedback and comments were written. It was decided that therapist/psychologist should not have access to data export module at any stage of the study and only users registered as researcher in the system should be allowed to access data export module. Moreover researcher should be allowed to export data related only to his own study. Another most important decision was regarding *when* researcher should be allowed to export data from a study. It was proposed that study status should be set to finished/completed and data export module should export a study that is already completed. Another topic discussed during demo was regarding participant privacy. It was requested to make sure data de-identification¹¹ before exporting for data analysis. The main outcome of the intervention was that ***data export module should have restricted access regarding who is allowed to export data, which data and when. Furthermore exported data must be anonymous and de-identified.***

According to Hevner, et al. (2004) “The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well executed evaluation methods”. In previous BIE cycle formative evaluation¹² was performed. During this cycle scenarios based descriptive evaluation was performed. During scenarios based descriptive evaluation detailed scenarios around the artifact constructed to demonstrate its utility (Hevner, et al., 2004, p. 86). The evaluation of prototype was based on the requirements established by organization’s environment. In my case the environment includes the technical infrastructure like U-CARE platform. The prototype was evaluated in terms of functionality, accuracy, reliability, usability, consistency, performance, completeness, and other relevant quality attributes. During this cycle my focus was the integration of the prototype with U-CARE platform under organization settings which is itself an evaluation.

¹¹ Stripping personally identifying information such as names, home addresses, social security numbers, etc. from the data.

¹² an artifact still under development is evaluated to determine areas for improvement and refinement

During this cycle the U-CARE platform was tested regarding randomization process¹³ by IS researchers. The test study “Randomization” was created. Test participants were created and assign to test study (see Figure 27). In the study observation points were created and quizzes (questionnaire) were allocated to observation points (see Figure 28 and Figure 29 respectively). Inclusion variables were created for first quiz that was related to randomization process (see Figure 30). The randomization process starts when any participant first login to the system and answer the quiz associated to randomization. Based on the quiz answers result variables are populated. The participant is assigned a treatment group based on the results. This test case was an efficient, effective and comprehensive scenario for data export module evaluation.

<input type="checkbox"/>	NICK NAME	Roll	STUDY GROUP	INTERVENTION NAME	RESPONSIBLE THERAPIST
<input type="checkbox"/>	Ucare9_post	control	control		RuthT
<input type="checkbox"/>	Ucare8_post	stepone	treatment		RuthT
<input type="checkbox"/>	Ucare7_post	reference	reference		RuthT
<input type="checkbox"/>	Ucare6_post	control	control		RuthT
<input type="checkbox"/>	Ucare5_post	reference	reference		RuthT
<input type="checkbox"/>	Ucare4_post	control	control		RuthT
<input type="checkbox"/>	Ucare3_post	stepone	treatment		RuthT
<input type="checkbox"/>	Ucare2_post	stepone	treatment		RuthT
<input type="checkbox"/>	Ucare1_post	reference	reference		RuthT
<input type="checkbox"/>	Ucare_post	stepone	treatment		RuthT
<input type="checkbox"/>	Jonas_techie	control	control		
<input type="checkbox"/>	Hafijtest	stepone	treatment		
<input type="checkbox"/>	Ucare9	reference	reference		RuthT
<input type="checkbox"/>	Ucare8_email	stepone	treatment		RuthT
<input type="checkbox"/>	Ucare7	reference	reference		RuthT

SELECT ALL IN THIS PAGE | DESELECT ENTIRE ACTION

Figure 27: Test study and test participants

Observationspunkter						
Namn	Observera	CREATE	HOURSE TO FINISH ALL ITEMS	DAYS FROM THE FIRST OP	MAX DAYS	ACTION
Third obs pt	This is the third obs - it should be at the 10th days of the study, that is 3 days after the 2nd obs	2012-05-14 14:26:13	24	10	5	ASSIGN INSTRUMENTS
Second observation	Second observation point - day 7 into the study. This means the 2nd obs will come 2 days after the 1st obs.	2012-05-14 14:23:19	24	7	5	ASSIGN INSTRUMENTS
First real obs point	This is meant to be the first observation point - 5 days into the study	2012-05-14 14:21:45	24	5	5	ASSIGN INSTRUMENTS
Inclusion? or First?	Shouldn't this be the inclusion conditions or something like that and not an observation point???	2012-05-14 14:14:16	24	0	5	ASSIGN INSTRUMENTS

Lägg till observationspunkt

Figure 28: Observation points in the randomization study

¹³ Randomisation is the process of assigning clinical trial participants to treatment groups

HADS (Hospital Anxiety and Depression Scale)

Läs varje påstående och markera det svar som kommer närmast hur du känt dig den senaste veckan. Fundera inte alltför länge. Det första svar som dyker upp är antagligen riktigare än ett svar som du funderat på länge.

[Tillbaka](#)

PAGE 1

1. Jag har känt mig spänd eller "uppskruvad"

☐ För det mesta
☐ Ofta
☐ Då och då
☐ Inte alls

2. Jag har uppskattat samma saker som förut

☐ Precis lika mycket
☐ Inte riktigt lika mycket
☐ Bara lite
☐ Knappast alls

14. Jag har kunnat njuta av en bra bok, eller av ett bra radio- eller TV-program

☐ Ofta
☐ Ibland
☐ Inte ofta
☐ Mycket sällan

15. Har något särskilt hänt den senaste veckan som kan ha påverkat dina svar?

☐ Ja
☐ Nej

[Spara och skicka senare](#) [FINISH](#) [Avbryt](#)

Figure 29: Quiz at first observation point associated to randomization process

INCLUSION						
ITEM	RESULT	EXPRESSION	MIN	MAX	ACTION	
HADS (WP2)	Ångestindex	return @q1@ + @q3@ + @q5@ + @q7@ + @q9@ + @q11@ + @q13@;			CHANGE	
HADS (WP2)	Depressionsindex	return @q2@ + @q4@ + @q6@ + @q8@ + @q10@ + @q12@ + @q14@;			CHANGE	
HADS (WP2)	Totalindex	return @Ångestindex@ + @Depressionsindex@;			CHANGE	
HADS (WP2)	Ångest (class)	if(@Ångestindex@ < 8) return "ingen"; else if(@Ångestindex@ >= 8 && @Ångestindex@ < 11) return "mild"; else return "svårt";			CHANGE	
HADS (WP2)	Depression (class)	if(@Depressionsindex@ < 8) return "ingen"; else if(@Depressionsindex@ >= 8 && @Depressionsindex@ < 11) return "mild"; else return "svårt";			CHANGE	
HADS (WP2)	alert	if(@Depressionsindex@ >= 15) return true; return false;			CHANGE	
HADS (WP2)	inclusion	if(@Ångestindex@ > 7 @Depressionsindex@ > 7) return true; return false;	true	true	CHANGE	

Figure 30: Result variables for Inclusion

The randomization study data was exported using the prototype and verified against the U-CARE system as well as database. The quiz results and answers were counter checked. The IS researcher were satisfied with the results and format of the data export. ***It was noted that the data export not only provide data export functionality but also an efficient reporting tool*** as data can be exported using different parameters, layout and allow selective pivoting. Furthermore data export was also provides data visualization to some extent. The Figure 31 shows the data exported without using pivoting. Only fields selected by user were exported. At first the questions, answers, and results of each quiz in first observation point were listed, then second and so on until data related to one participant was completed. Keeping the same sequence second, third and so on all participant data listed row wise. Figure 32 (left) shows the data exporting using pivoting on the questions, answer choices and results. We can see the data related to a particular participant is in one row. All questions are listed and then for each question all answer choices list as well as the selected choice by participant to answer the question. At the end of row all results are listed in one row. This format not only provides data for analysis but also metadata about the whole study. In this way our exported data made export file independent of the application using it. At any later stage without login to the system researcher can see what was the quiz, no of questions, answers choices, user selection, and results? Results can be directly be used in analysis or the individual question answer can be used for further analysis. Figure 32 (right) shows also shows data exporting using pivoting. Only question serial no, answer weight and results were selected. During the evaluation session feedback and comments were received. The data export module generic interface and ability to select fields to export was much appreciated. ***It was suggested that data export module should provide ability to filter the rows.*** Which means user can select what records/rows he wants to export in the same way as user can select the fields/columns for export e.g. a specific study, a particular observation point and or a quiz etc.

1-WORKPACKAGE-NAME	2-STUDY-NAME	3-QUESTION-SERIAL	4-ANSWERS-WEIGHT	13-QUIZRESULTS-RESULT_NAME	14-QUIZRESULTS-RESULT
WP2	Randomization	1	2		
WP2	Randomization	2	1		
WP2	Randomization	3	2		
WP2	Randomization	4	1		
WP2	Randomization	5	2		
WP2	Randomization	6	1		
WP2	Randomization	7	2		
WP2	Randomization	8	1		
WP2	Randomization	9	2		
WP2	Randomization	10	1		
WP2	Randomization	11	2		
WP2	Randomization	12	1		
WP2	Randomization	13	2		
WP2	Randomization	14	1		
WP2	Randomization	15	1		
WP2	Randomization	17	0		
WP2	Randomization			Ängestindex	14
WP2	Randomization			Depressionsindex	7
WP2	Randomization			Totalindex	21
WP2	Randomization			Ängest (class)	svårt
WP2	Randomization			Depression (class)	ingen
WP2	Randomization			alert	False
WP2	Randomization			inclusion	True
WP2	Randomization	1	1		
WP2	Randomization	2	2		
WP2	Randomization	3	1		
WP2	Randomization	4	2		
WP2	Randomization	5	1		
WP2	Randomization	6	2		
*****	*****	*****	*****	*****	*****
*****	*****	*****	*****	*****	*****
*****	*****	*****	*****	*****	*****
*****	*****	*****	*****	*****	*****

Figure 31: Data exported in excel format without pivoting

5-QUIZ-NAME	6-QUESTION-QUIZE_QUESTION	7-QUESTION-SERIAL	8-OPTIONS-ANSWER	9-OPTIONS-WEIGHT	10-OPTIONS-SERIAL	20-ANSWERS-WEIGHT
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	2
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	1
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	0
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	3
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	0
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	1
HADS (WP2)	Jag har känt mig spänd eller "uppskruvad"	1	För det mesta	3	1	1
....

1-WORKPACKAGE-NAME	2-USER-NICKNAME	3-STUDYGROUP-NAME	4-OBSERVATIONPOINT-NAME	5-QUIZ-NAME	6-QUESTION-SERIAL	7-ANSWERS-WEIGHT	36-QUIZRESULTS-RESULT_NAME	37-QUIZRESULTS-RESULT
WP2	Ucare9_post	control	Inclusion? or First?	HADS (WP2)	1	2	Ängestindex	14
WP2	Ucare8_post	treatment	Inclusion? or First?	HADS (WP2)	1	1	Ängestindex	7
WP2	Ucare7_post	reference	Inclusion? or First?	HADS (WP2)	1	0	Ängestindex	4
WP2	Ucare6_post	control	Inclusion? or First?	HADS (WP2)	1	3	Ängestindex	17
WP2	Ucare5_post	reference	Inclusion? or First?	HADS (WP2)	1	0	Ängestindex	0
WP2	Ucare4_post	control	Inclusion? or First?	HADS (WP2)	1	1	Ängestindex	10
WP2	Ucare3_post	treatment	Inclusion? or First?	HADS (WP2)	1	1	Ängestindex	9
WP2	Ucare2_post	treatment	Inclusion? or First?	HADS (WP2)	1	3	Ängestindex	21
WP2	Ucare1_post	reference	Inclusion? or First?	HADS (WP2)	1	1	Ängestindex	7
WP2	Ucare_post	treatment	Inclusion? or First?	HADS (WP2)	1	2	Ängestindex	14
WP2	Jonas_techie	control	Inclusion? or First?	HADS (WP2)	1	3	Ängestindex	21
....

Figure 32: Data exported in excel format using pivoting

5.3 Identifying Design Principles

During second cycle of BIE we merge my findings with the knowledge base in order to enrich already identified design principles during previous cycle. Considering the feedback from intervention and evaluation sessions we continued by discussing with ADR team and drawing suggested design solutions to answer challenges and problems. Table 8 describes a set of design principles that were derived from our findings, during BIE first cycle. The sentences with bold letters reflect modification and/or addition to existing principles.

Table 8: Design principles (version 2)

Design Principle	Description
i. User-friendly	Data export should be easy to use and data should be exported with minimum clicks.
ii. Simple	Data export features should be designed for end users (researchers) and should not require much technical knowledge.
iii. Mutable	Data export should adopt itself to the changes, as and when they occurred , in the research application/tool due to change in end-user's requirements.
iv. Render useful output	Data export should render useful output that is importable by data analysis / statistical application.
v. Require least resources	Data export should require less development/implementation resources. In other words developer should not have to spend much time developing data export functionality. They should have more focus on developing application for core business functionalities.
vi. Generic	Data export should require minimum or no development on every data export request. Strictly speaking system should handle dynamically nearly all possible present and future data export requirements. In some cases system should be require development with minimum ripple effects.
vii. Restrict data access	Data export module should have restricted access regarding who is allowed to export data, which data and when. This means users (or researchers) need data from the (clinical trial) databases for analysis, but typically should only need (or are authorized to) a certain subset of the data at specific stage (like study completion).
viii. Preserve data 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 should be impossible.

6 BIE Cycle III

The last BIE cycle in ADR is summative evaluation¹⁴ of a beta version (Venable, et al., 2012). The third BIE cycle was the continuation of second BIE cycle, a comprehensive intervention in a wider organizational context and summative evaluation. Prototype (beta v2) construction was started by exploring design solution for the problem identified during intervention and evaluation of second cycle.

6.1 Constructing the Prototype (Beta v2)

After the second prototype, the intervention and evaluation thereof we continued with further development based on the attained feedback from the meeting with the researchers. The feedback was about the data access restriction and data filtration. The third prototype was an evolution with additional features like row filtration and data access configuration.

6.1.1 Restricted Access to Data – Configuring U-CARE Platform

The second prototype (beta v1) provides data regardless of user or his role and user can export study data at any stage. During intervention it was decided that only users with ‘researcher’ role should have access the data export module. U-CARE platform has *“dynamic setup of actions, authentication, authorization, and menus. Action and menu setup is connected to dynamic roles. The actions and menus are specifically configured for each study, which further promotes the use of the platform in different research and treatment situations”* (Sjöström, et al., 2011). Due to availability of such module it was very easy to implement end-users requirement to provide data export module access to a specific role. The configuration request for data export module was added to to-do-list of platform developers. To provide access to user’s own study only, data objects assembler function was updated to use another function of study model which list the studies related to login user only. The last requirement was to export data after a specific time like when study status is completed. This required to modify the U-CARE platform study model class and was added to to-do-list of platform developers at top priority. As soon as authorization configured and model classes modified in the U-CARE platform the data export module will adopt it automatically without need any further development.

6.1.2 Anonymity of Data

U-CARE platform keeps the privacy of study participants by keeping two separate databases. One database keeps the participants’ identifiable data while other database keeps all study related data. Both databases are linked

¹⁴ A summative evaluation is holistic in that we evaluate the artifact in every aspect instead of focusing only on one area. Artifacts are evaluated against criteria of value or utility – does it work? The main focus is on the outcome – how well the artifact performs with respect to its stated purpose.

through a participant's nick name which is used only for U-CARE system. Data export module has only access to the study database has no explicit identifiers. Re-identification¹⁵ is nearly impossible using exported data.

6.1.3 Render Required Data Only - Designing Filter

The second prototype (beta v1) lists all data at every node of the tree structure of data objects e.g. 'WorkPackages' node lists all active work packages or 'Studies' node lists all active studies with in specific work package. To provide row selection filters are designed. The user interface was modified to provide a link "FILTER" nears every list node (see Figure 33). When user clicks on the filter link system dynamically generates the user interface for that node (see user click on WorkPackages node at Figure 34) and user can select a specific record to export. The dynamic user interface is generated by passing current populated DO to reflection utility. This provides row wise filtrations.

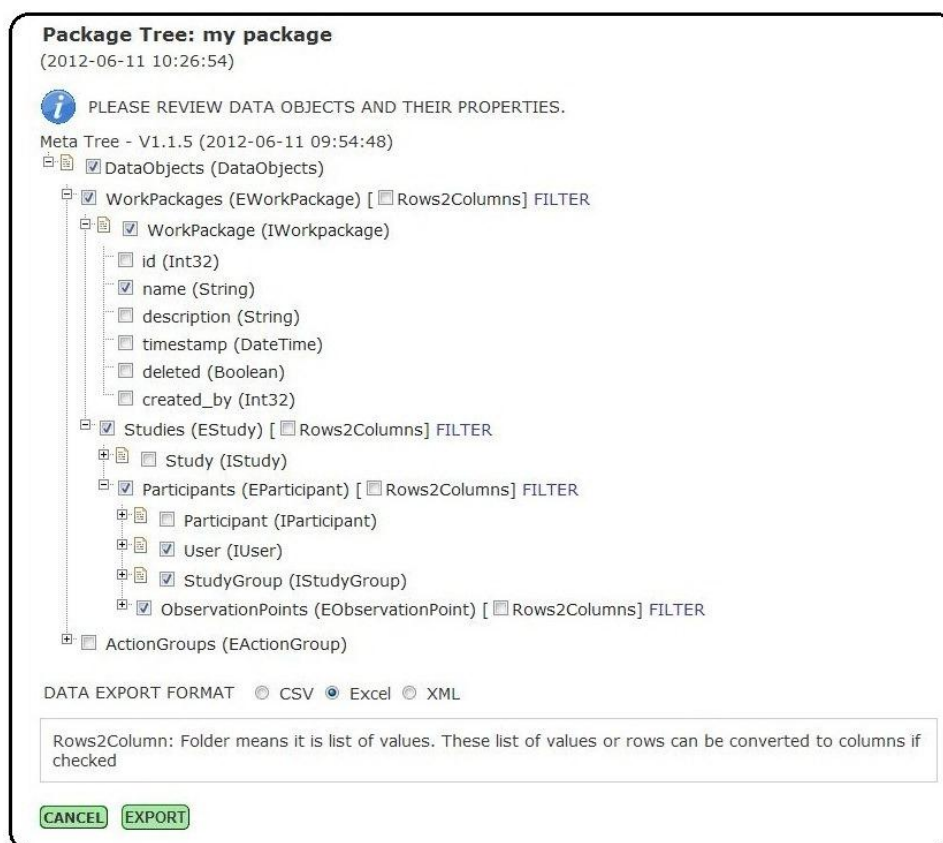


Figure 33: Modified user interface for data export fields selection using filters

¹⁵ Re-identification is the discovery, or determination, of the identity of the individuals who are the participants/subjects of a study through data linkage techniques.

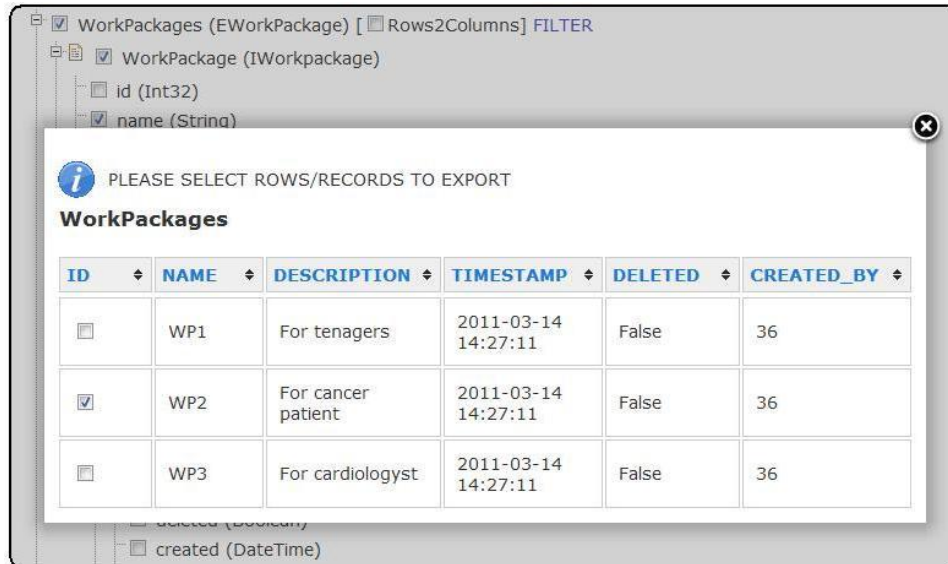


Figure 34: User interface for data export filter

To save user selection a table was created in the database (see ExportPackageRows table in Figure 20) and associated model class PackageModel was also updated. Now system not only saved the columns/fields but also the rows/records to export.

6.2 Intervention and Evaluation with End-users

Following the ADR method we once again conducted four rounds intervention and evaluation sessions. The intervention was semi structured interviews with open-ended questions carried out on-site followed by the evaluation in same session as a controlled experiment to study artifact in controlled environment for qualities (e.g., usability, utility). The interviews were recoded as well for further qualitative analysis. The first three sessions was each with different clinical researcher (end-user), having experience of data analysis, from different organisational background. The fourth intervention and evaluation session was performed with practitioner (developer) who involved in using, discussing and providing feedback on the prototype and its application in data export contexts.

At each intervention and evaluation session we started out by asking a few questions (as semi-structured interviews) about the user's view related to data export, their experience in use of data export tools and expectations with newly build prototype. After a few questions we presented a very short description of functions, intended use and interface of the prototype. This was followed by a demonstration about how to use prototype using same randomization study data used in previous evaluation session (see p. 50) and almost immediately allowing the interviewee to interact with prototype. The user interaction was part of controlled experiment where we allowed user to perform sequence of task to export data. During and after interaction with prototype we continued the evaluation with questions about the prototype and its usage.

The researchers were asked about which possibilities and gains as well as which problems and difficulties they perceived when using it. We queried them on what kind of information they thought would be interesting to export with export functionality like this (such as data and/or metadata about studies). We also asked them about whether data export module fulfil their requirements regarding its use, interface, functionality and reliability. During last session while interviewing developer my focus was evaluation of the prototype regarding design benefits like compatibility, extensibility, fault-tolerance, maintainability, modularity, reliability, re-usability, robustness, security and usability. In addition to doing audio recordings we also took notes of the sessions, which were used for recollecting the usage of prototype and feedback by the different researchers. Some of the comments regarding the data export are given below.

“I think data export is not complicated by using the system but I am interested in understanding the system more Data export interface is ok absolutely and understandable I think it looks more complicated than actually it is and its first impressions are oh my God! This is can't work but once it is taking run you start understand how it works someone need documentation or help to understand what filed should not care about while exporting (Overall) it is quite impressive and looks good I think actually this is more customizable and user friendly than I thought it would be ... I thought it would be very much hard to export data.”

“I think it need some training to use tree view there should be a save as function for packages system should provide facility to choose headers from selected fields while creating a data export package..... the CSV data export format looks ok after importing to statistical tool I am looking forward to exploiting the opportunities the data export module.”

“It is very good that we have these extra functions (pivoting) it is (the system) straight forward. It seems very flexible and has a lot of options I especially like the option to choose conversion of columns to rows it is very good the user interface is little bit complex I think if you not use to working this type of tree structure ... but with some practice I guess it can be done when you know what these items correspond to I guess there will be some learning curve doing this ... I mean it is manageable based on what is shown to me I can extract data. It is quite straight forward.”

“The data export engine is very good as it follows design patterns one of major concern regarding data export performance when we have huge data accumulated the good thing about data objects is that when the data export can work independent of the platform, this is really fantastic work ever new requirements comes we only need to modify it and rest of the system works as it different design aspects are well implemented.”

The generic design of data export faces same design challenge Sjöström et al., (2011) mentioned it as “*design tension - a trade-off situation between simplicity and mutability-in-use*”. Generic design of data export requires more training and usage. The data export displays all the fields/properties of the model including some fields that are stored to manage the logic e.g. timestamp, deleted, created_by etc. (see Figure 33 under WorkPackages node). Due to this user interface become little complicated at first glance. This requires understanding of the field while exporting data. The main output of the intervention and evaluation was that ***there should not be too much customization and GUI¹⁶ should not present irrelevant data***. Most of the end-users were satisfy with existing data export module, however we need more BIE cycle(s) to satisfy all end-users.

Beside complicated interface end-users agreed that either data export package are designed by the experienced developers and end-users only perform filtration and pivoting, or there should be more learning and training sessions to understand the interface better. The data export module and U-CARE project currently on-going projects. Even though current thesis is limited to present first three cycles due to time constraints, the designed artifacts and research finding are at sufficient stage to present. The results of next BIE cycles or another cycle of whole ADR process will be presented in another research publication.

6.3 Identifying Design Principles

During third cycle of BIE we merge my findings with the knowledge base in order to enrich already identified design principles during previous two cycles. Considering the feedback from intervention and evaluation sessions we continued by discussing with ADR team and drawing suggested design solutions to answer challenges and problems. Table 9 describes a set of design principles that were derived from our findings, during previous two BIE cycles. The sentences with bold letters reflect addition to existing principles.

Table 9: Design principles (version 3)

Design Principle	Description
i. User-friendly	Data export should be easy to use and data should be exported with minimum clicks.
ii. Simple	Data export features should be designed for end users (researchers) and should not require much technical knowledge.
iii. Mutable	Data export should adopt itself to the changes, as and when they occurred, in the research application/tool due to change in end-user’s requirements.
iv. Render useful output	Data export should render useful output that is importable by data analysis / statistical application.

¹⁶ Graphical user interface

Design Principle	Description
v. Require least resources	Data export should require less development/implementation resources. In other words developer should not have to spend much time developing data export functionality. They should have more focus on developing application for core business functionalities.
vi. Generic	Data export should require minimum or no development on every data export request. Strictly speaking system should handle dynamically nearly all possible present and future data export requirements. In some cases system should be require development with minimum ripple effects.
vii. Restrict data access	Data export module should have restricted access regarding who is allowed to export data, which data and when. This means users (or researchers) need data from the (clinical trial) databases for analysis, but typically should only need (or are authorized to) a certain subset of the data at specific stage (like study completion).
viii. Preserve data 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 should be impossible.
ix. Provide adequate customization	There should customization in design and GUI but it should not be too much or too difficult. GUI should also not present irrelevant data.

7 Conclusion

In order to answer to the next stage in the ADR process, formalisation of learning, we continued to merge my findings with the knowledge base in order to create design principles. We generalized the findings in the light of the research framework which resulted in these principles, meant to provide guidance for further research and design work on data export functionality, outside the domain of clinical research. At the time when this research report was written, the data export module was entering next ADR cycle. Even though the data export module is still subject to some uncertainties, further learning about the system and its feasibility can only be achieved through further development, intervention and evaluation, and having it tested by the end-users. As a result of the project so far, we have introduced a design of the data export module and few design principles for such a system.

Although we have not reached the end of the entire cycle yet, several contributions have already been made. First, the practical contribution for the case organization is in designing the data export process and the data export module for clinical researcher. Second, this study makes contribution to practice as well as research by proposing not only a novel process and system for data export in clinical research, but also the nine design principles for such a system (see Table 10). Furthermore, these design principles were reflected upon problems of similar class, in addition to the one at hand. Hence, our study also contributes by offering guidelines for building IT artefacts in clinical research organisations. The theoretical contribution is provided by enhancing our understanding of data export problems in agile development and how to solve or mitigate them with the use of generic and dynamic design. Third, the methodological contribution of this study results from applying and testing the ADR approach in the context of a process and system design initiative in a clinical research organisation. Furthermore, our case allowed us to test the IS development method Agile and IS research method ADR in an organisational setting. The evaluation of the proposed theoretical design is done by creating an instantiation of the artifact, that is, a prototype of the data export functionality, built using the design principles evolved over the time in context of U-CARE. The data export architecture represents artifact as model (see Figure 26). With this generalized architecture we contributed in laying the groundwork for further research and development of design patterns and /or extension of MVC architecture. The data export module's prototype is already integrated and running at <https://www.u-care.se>. Appendix B presents some the features of the data export prototype.

Table 10: Design Principles

Design Principle	Description
i. User-friendly	Data export should be easy to use and data should be exported with minimum clicks.
ii. Simple	Data export features should be designed for end users (researchers) and should not require much technical knowledge.
iii. Mutable	Data export should adopt itself to the changes, as and when they occurred, in the research application/tool due to change in end-user's requirements.

Design Principle	Description
iv. Render useful output	Data export should render useful output that is importable by data analysis / statistical application.
v. Require least resources	Data export should require less development/implementation resources. In other words developer should not have to spend much time developing data export functionality. They should have more focus on developing application for core business functionalities.
vi. Generic	Data export should require minimum or no development on every data export request. Strictly speaking system should handle dynamically nearly all possible present and future data export requirements. In some cases system should be require development with minimum ripple effects.
vii. Restrict data access	Data export module should have restricted access regarding who is allowed to export data, which data and when. This means users (or researchers) need data from the (clinical trial) databases for analysis, but typically should only need (or are authorized to) a certain subset of the data at specific stage (like study completion).
viii. Preserve data 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 should be impossible.
ix. Provide adequate customization	There should customization in design and GUI but it should not be too much or too difficult. GUI should also not present irrelevant data.

7.1 Concluding Discussion

Concluding the work of this thesis we have reached to several deductions regarding generic data export functionality in clinical research applications. Healthcare technology, practice and knowledge are continuously evolving at a rapid pace. The dynamic and ever changing environment of clinical research could be better understood by the usage of generic data export functionality. Agile development is a better way for research projects when domain requirements are not clear and/or changing very fast. Furthermore ADR has proven to be a suitable method for conducting exploration of the design principles for this kind of functionality. The use of the method has resulted in a set of design principles, for future research and development projects. ADR fit well with agile development and for multi-disciplinary research project like U-CARE. It is an example of ADR implementation which can be taken by academic for teaching and learning. There is a need of design patterns or libraries within the existing frameworks, to

export data, to ease the development. Table 11 presents the ADR process summary in relation to exiting research findings.

Table 11: Summary of the ADR Process

Stages and Principles		Artifact
Stage 1: Problem Formulation		
Principle 1: Practice - Inspired Research	Research was driven by the need for generic Data export functionality in data-centric clinical research application.	Recognition: Shortcomings of the existing system and challenges to export data from dynamic and generic data-centric clinical research application (U-CARE platform).
Principle 2: Theory - Ingrained Artifact	The theory used was MVC design pattern, Agile principles and data transfer patterns like DTO and SDO.	
Stage 2: BIE		
Principle 3: Reciprocal Shaping	Problems encountered were iteratively addressed and formulated as early design principles in collaboration with practitioners.	Alpha Version: The prototype was designed to conceive design ideas which lead to early design principles. Beta Version: The prototype enhanced and upgraded to implement and evaluate design principles in wider organizational context. The prototype's architecture is designed and refined
Principle 4: Mutually Influential Roles	The ADR team included (IS) researchers and practitioners (U-CARE platform developers) in order to include theoretical, technical, and practical perspectives.	
Principle 5: Authentic and Concurrent Evaluation	Data export module was first evaluated within the ADR team and then in the wider setting of end-users at U-CARE.	
Stage 3: Reflection and Learning		
Principle 6: Guided Emergence	The ensemble nature of the data export artifact was recognized. Furthermore, design elements for the data export module's components and changes to assumptions related to data export practices emerged.	Emerging Version and Realization: New challenges and requirements for the data export artifact emerged during BIE stage. Each BIE cycle leads to revision of design principles.
Stage 4: Formalization of Learning		
Principle 7: Generalized Outcomes	A set of design principles to design data export functionality in a clinical research application were articulated, positioning U-CARE platform as an instance.	Ensemble Version: An ensemble embodying the design principles and data export artifacts for such functionality development and implementation.

7.2 Future Work

The prototype produced has possibilities for future continued exploration, as it constitutes an example of possible design solutions in an area that is largely unexplored. The concept of models use in data export brings up interesting issues, well suited for future research. Our firm belief is that research on the development and possible uses of data export architecture in other domains could result in even more understanding of how to design for generic data export functionality. We conclude our thesis report by indicating further possible advancement in existing design:

1. **GUI improvement:** Development of an interface for developers (and/or data coordinator / clinical research associate¹⁷) to select the attributes those should be displayed to user. Another interface for end-users so that they can select what column(s) should be treated as header(s) and what as data.
2. **Data mart development:** Design and development of a system to provide facility to configure a data mart so that data is exported to data mart after the clinical trial study is completed. After that align current data export module so that it can export data from data mart by displaying minimal fields at user interface.

¹⁷ In any clinical trial data coordinator / clinical research associate is responsible for quality control of data. This person is also responsible for generating edit queries and data requests, for processing patient registrations, and for maintaining all study files. The data coordinator assists the statistician in preparing data sets for analysis, and he or she is the primary contact with the trials personnel at the participating sites.

8 References

- Abrahamsson, P., Conboy, K., Galway, N. & Wang, X., 2009. 'lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems*, 18(4), pp. 281-284.
- Andersson, G. et al., 2005. Internet-based self-help for depression: randomised controlled trial. *British Journal of Psychiatry*, pp. 456-461.
- Andersson, G. et al., 2006. Internet-based self-help with therapist feedback and in vivo group exposure for social phobia: A randomized controlled trial. *Journal of Consulting and Clinical Psychology*, p. 677–686.
- Barak, A. & Grohol, J. M., 2011. Current and Future Trends in Internet-Supported Mental Health Interventions. *Journal of Technology in Human Services*, 29(3), pp. 155-196.
- Beck, K. et al., 2001. Principles behind the Agile Manifesto. [Online] Available at: <http://agilemanifesto.org/principles.html> [Accessed 10 March 2012].
- Bennett, T. A. & Bayrak, C., 2011. Bridging the data integration gap: from theory to implementation. *ACM SIGSOFT Software Engineering Notes*, 36(3), pp. 1-8.
- Berger, T., Hohl, E. & Caspar, F., 2009. Internet-based treatment for social phobia: a randomized controlled trial. *Journal of Clinical Psychology*, pp. 1021-1035.
- Bordbar, B. et al., 2005. Integrated Model-Based Software Development, Data Access, and Data Migration. Montego Bay, Jamaica, *Proceedings of 8th International Conference, MoDELS 2005, LNCS 3713*, pp. 382-396.
- Brandt, C. A. et al., 2002. Metadata-driven creation of data marts from an EAV-modeled clinical research database. *International Journal of Medical Informatics*, 65(3), pp. 225-241.
- Casteleyn, S., Daniel, F., Dolog, P. & Matera, M., 2009. Engineering Web Applications (Data-Centric Systems and Applications). 1st ed. New York: *Springer*.
- Ceri, S., Fraternali, P. & Matera, M., 2002. Conceptual modeling of data-intensive Web applications. *Internet Computing, IEEE*, 6(4), pp. 20-30.
- Cockburn, A., 2001. Agile Software Development. Boston: *Addison-Wesley Professional*.
- Codd, E. F., 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), p. 377–387.
- Cooper, C. J. et al., 2006. Web-based data collection: detailed methods of a questionnaire and data gathering tool. *Epidemiologic Perspectives & Innovations*, 3(1).
- Croaken, M., 2004. Review of Doron Swade 'The Cogwheel Brain: Charles Babbage and the Quest to Build the First Computer'. *The British Journal for the History of Science*, Volume 37, pp. 351-352.

- Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11), p. 27–34.
- Fowler, M. et al., 2002. Patterns of Enterprise Application Architecture. 1st ed. Boston: *Addison-Wesley Longman Publishing Co., Inc.*.
- Fris, M., Nilsson, M. & Sollerhed, V., 2011. Real-Time Social Network - Exploring the Design Space for a Multi-User Real-Time Visualisation Tool for Social Network Analysis. [Online] Available at: <http://hdl.handle.net/2077/26689> [Accessed 20 01 2012].
- Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995. Design Patterns: Elements of Reusable. 1st ed. Boston: *Addison-Wesley Pub Co.*
- Gregor, S. & Jones, D., 2007. The anatomy of a design theory. *Journal of the Association for Information*, 8(5), p. 312–335.
- Hevner, A. R., 2007. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19(2).
- Hevner, A. R., March, S. T., Park, J. & Ram, S., 2004. Design Science in Information Systems Research. *MIS Quarterly*, pp. 75-105.
- IBM & BEA, 2006. Service Data Objects. [Online] Available at: <http://www.ibm.com/developerworks/library/specification/ws-sdo/> [Accessed 10 05 2012].
- Järvinen, P., 2000. Research Questions Guiding Selection of an Appropriate Research Method. Vienna, *Proceedings of the 8th European Conference on Information Systems*, p. 124–131.
- JCP, 2007. JSR 235. [Online] Available at: <http://jcp.org/en/jsr/detail?id=235> [Accessed 10 05 2012].
- Kelders, S. M., Oskam, M. J., Bohlmeijer, E. T. & Gemert-Pijnen, J. V., 2012. Human Centered Development of a Web-based Intervention for the Prevention of Depression. Valencia, Spain, *The Fourth International Conference on eHealth, Telemedicine, and Social Medicine*, pp. 6-11.
- Kelvin, W. T., 1889. Popular lectures and addresses. Vol. I, p. 73. [Online] Available at: http://openlibrary.org/books/OL7047041M/Popular_lectures_and_addresses. [Accessed 01 June 2012].
- Knatterud, G. L., 2002. Management and Conduct of Randomized Controlled Trials. *Epidemiologic Reviews*, 24(1), pp. 12-25.
- Lempinen, H., Rossi, M. & Tuunainen, V. K., 2012. Design Principles for Inter-Organizational Systems Development – Case Hansel. In: K. Peffers, M. Rothenberger & B. Kuechler, eds. Design Science Research in Information Systems. Advances in Theory and Practice, *Lecture Notes in Computer Science. Berlin / Heidelberg: Springer*, p. 52–65.
- Lempinen, H. & Tuunainen, V. K., 2011. Redesigning the supplier reporting process and system in public procurement – case Hansel. *Int. J. Organisational Design and Engineering*, 1(4), pp. 331-346.

- Morris, H. et al., 2008. Bringing Business Objects into Extract-Transform-Load (ETL) Technology. Xi'an, China, *Proceedings of ICEBE '08. IEEE International Conference on e-Business Engineering*, pp. 709-714.
- Perini, S., Titov, N. & Andrews, G., 2009. Clinician-assisted Internet-based treatment is effective for depression: Randomized controlled trial. *Australian and New Zealand Journal of Psychiatry*, pp. 571-578.
- Racz, N., Weippl, E. & Bonazzi, R., 2011. IT Governance, Risk & Compliance (GRC) Status Quo and Integration: An Explorative Industry Case Study. Washington, DC, *Proceedings of 2011 IEEE World Congress on Services*, pp. 429 - 436.
- Rini, C., Williams, D. A., Broderick, J. E. & Keefe, F. J., 2012. Meeting them where they are: Using the Internet to deliver behavioral medicine interventions for pain. *Translational Behavioral Medicine*, 2(1), pp. 1-11.
- Schwabe, D. & Rossi, G., 1998. An object-oriented approach to web-based application design. *Theory and Practice of Object Systems*, 4(4), p. 207-225.
- Sein, M. K., Henfridsson, O., Purao, S. & Rossi, M., 2011. Action Design Research. *MIS Quarterly*, 35(1), pp. 37-56.
- Sjöström, J., 2010. Designing Information Systems: A Pragmatic Account. PhD Dissertation. Uppsala: *Uppsala University*, Sweden.
- Sjöström, J. & Ågerfalk, P. J., 2009. An Analytic Framework for Design-oriented Research Concepts. San Francisco, California, *Proceedings of the Fifteenth Americas Conference on Information Systems*.
- Sjöström, J., Ågerfalk, P. J. & Lochan, A. R., 2011. Mutability matters - baselining the consequences of design. Cyprus, *Proceedings of the 6th Mediterranean Conference on Information Systems (MCIS2011)*.
- Titov, N. et al., 2009. Clinician-assisted Internet-based treatment is effective for generalized anxiety disorder: randomized controlled trial. *Australian and New Zealand Journal of Psychiatry*, pp. 905-912.
- Troyer, O. D. & Leune, K., 1998. Wsdm: A user centered design method for web sites. *Computer Networks and ISDN Systems*, 30(1-7), p. 85-94.
- U-CARE, Uppsala University, 2012. Uppsala University Psychosocial Care Programme: U-CARE. [Online] Available at: <http://www.u-care.uu.se/> [Accessed 01 March 2012].
- Venable, J., Pries-Heje, J. & Baskerville, R., 2012. A Comprehensive Framework for Evaluation in Design Science Research. In: K. Peffers, M. Rothenberger & B. Kuechler, eds. Design Science Research in Information Systems. *Advances in Theory and Practice, Lecture Notes in Computer Science. Springer Berlin / Heidelberg: Springer*, pp. 423-438.

Appendix A: Glossary

Introduction

The following terms defined as they have either not been defined or have not been explicitly defined within the body of the thesis. The purpose of the glossary is to provide a common understanding of the terms used in thesis. Terms that are not defined in the glossary are defined as they occur within the various chapters.

Definition of terms

Design considerations

The terms related to design consideration are taken from the source as it (without modification) for only quick reference and depict our understanding of the terms while referring them in the thesis.

Compatibility - The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.

Extensibility - New capabilities can be added to the software without major changes to the underlying architecture.

Fault-tolerance - The software is resistant to and able to recover from component failure.

Maintainability - The software can be restored to a specified condition within a specified period of time. For example, antivirus software may include the ability to periodically receive virus definition updates in order to maintain the software's effectiveness.

Modularity - the resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.

Reliability - The software is able to perform a required function under stated conditions for a specified period of time.

Re-usability - the modular components designed should capture the essence of the functionality expected out of them and no more or less. This single-minded purpose renders the components reusable wherever there are similar needs in other designs.

Robustness - The software is able to operate under stress or tolerate unpredictable or invalid input. For example, it can be designed with resilience to low memory conditions.

Security - The software is able to withstand hostile acts and influences.

Usability - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

(Source: http://en.wikipedia.org/wiki/Software_design, accessed on 10 January 2012)

Appendix B: Features of Data Export Module

1. Flexible, scalable and reusable design of data export module results in far more tractable in organizing efficient data export management.
2. Easy-to-use interface is based on generic design provides the user with many options as well as the flexibility to control data export. Using the data export module does not burden the user with a complex series of SQL queries for a one step request.
3. By moving the data export process away from the DBMS level, the data export module granted much more flexibility to developers in how the data is to be exported (data export is no longer static; it takes place on-the-fly as part of the web application itself).
4. Developers are no longer troubled by the need for external tools or services.
5. The data export module not only export the data but also can export metadata of the research instruments (e.g. questionnaires) which make exported data independent of the software application. In long run if we have data independent of application we can use it for secondary use.
6. Seamless exporting of data enables users to exploit extensive U-CARE platform for further possible data analysis.
7. Furthermore the design is also platform independent as it is based on MVC model concept and MVC is available in all major languages
8. Data objects class decouple the data export module from U-CARE platform

