UiO : Department of Informatics
University of Oslo

Use Cases Application in Practice: A Case Study

Master's thesis, 60 credits

Behnaz Saemi

Spring 2014

Preface

This study is a master's thesis at the Department of Informatics, at the Faculty of Mathematics and Natural Sciences, at University of Oslo (UiO). The study was conducted from November 2012 until February 2014.

I would like to express my gratitude to my supervisor, Tone Bratteteig, for her invaluable and constructive advices under the course of the thesis. I want also to especially thank Kari Kapstad, the leader of the Development Unit of the IT-department at the National Institute of Public Health, for providing me the opportunity of doing the thesis, and for her precious support. Moreover, I am highly thankful to Peter Holmes and Kent Aune for their time and great contributions to this thesis, and to Henriette Fossum for her detailed and excellent feedback and advice.

Last, but not least, I want to thank my husband for his encouragements, patience, support and for taking care of our little son while I worked with this thesis. Without him finishing this study was not possible.

Behnaz Saemi

Oslo, April 2014

Summary

Functional requirements describe the functionalities of a system and are an essential part of requirements specifications. As such, the correctness and completeness of these requirements are critical to develop a correct solution that satisfies the needs of different users and stakeholders.

Various techniques are applied to represent functional requirements. The artifacts provided by the techniques have different capabilities to model certain aspects of requirements. Selecting proper techniques to specify functional requirements is generally not a straightforward task and requires a wide understanding of the techniques, their characteristics and limitations.

Use cases are commonly used and accepted to model and specify functional requirements in both small and large development projects and for different types of products and solutions. Despite their wide usage, important aspects and issues of use cases application are not well understood and agreed upon by the practitioners. As addressed by literature, very few empirical studies concerning use cases and their issues based on real-life projects are conducted.

This thesis aims to investigate techniques applied to specify functional requirements in the development project for modernization of the Cause of Death Registry, in a case study approach. The two research questions of the study concern 1) application of use cases and challenges encountered by their application in different requirements activities, 2) usage of other techniques to specify functional requirements and their benefits and contributions to address use cases challenges.

To answer the research questions, I followed the project through different phases over than one year and collected qualitative data by interviewing project members, reviewing projects documents, searching literature and observing some of the projects activities. I particularly focused on application of use cases, their development and evolution through their lifecycles.

The results of the study regarding application of use cases revealed that:

- Use cases were specified by developers and for development purposes. Use cases were not used to communicate requirements with the stakeholders. Diagram models such as use case-, sequence- and state diagrams, in addition to general visualizations were used to communicate requirements with the stakeholders.
- Further, use cases were applied differently in different phases of the project. Application of use cases varied depending on the type of requirements and developers who specified and implemented use cases.

Investigation of other techniques used in specifying functional requirements underlined that utilizing multiple techniques was necessary to achieve a complete and correct specification of the requirements. In addition, the results of the investigation showed that some of the limitations of use cases were addressed by other techniques.

Moreover, through studying use cases and analyses of use cases changes, challenges related to use cases specifications were identified:

- Determination of the content and extent of specifications of use cases elements were challenging.

- Not all of the necessary specifications of the requirements and technical solutions could be categorized in the use cases elements.
- Use cases did not state their relationships with other use cases.

My suggestions to address the challenges are:

- Development of more detailed guidelines on specification of use cases elements.
- Use of different use case templates adjusted to the type of requirements.
- Further discussions and agreements on the level of detail, mandatory use cases elements, or the minimum specification of the elements sufficient to future maintenance and the system documentation.
- Application of other recommended techniques for specific types of requirements

The results of this study can be used to create a high-level guideline for deciding on relevant techniques to be used on different types of requirements and requirements activities.

However, the development of more detailed guidelines would require a deeper understanding of the relationships between the various techniques and how they can contribute to each other, necessitating further investigations and empirical knowledge gained from real-life projects.

Table of contents

1 Introduction

1.1 Background

Functional requirements describe the functions that a system should support or be able to perform in order to satisfy the needs of different stakeholders. As such a correct and complete specification of these requirements is critical to develop a correct solution. Moreover the progress of the activities such as design, development and test is directly affected by the requirements specification and its quality (1).

Requirements in general are developed through the Requirements Engineering (RE) process, by which requirements for systems and software products are gathered, analyzed, documented, validated and managed throughout the development life cycle (2). RE is considered as the most critical and difficult phase in the software development process. Reports from Standish Group and other researchers reveal that the major problems such as cost overruns, delays and errors in the final product are related to requirements issues such as incomplete requirements specifications, lack of user involvement and uncontrolled requirements changing (3) (1) (4).

Functional requirements are specified by various techniques through the RE process. Each technique provides different models or representations of the requirements – called requirements artefact – which are beneficial to different purposes and in different phases of the project. Use cases (2.6.1), user stories (2.6.3), state- and sequence diagrams (2.6.4) (2.6.5) are among the known and commonly applied techniques to specify functional requirements.

Selecting proper techniques to specify functional requirements is generally not a straightforward task and requires a wide understanding of the techniques, their capabilities and limitations. In addition, the application of multiple techniques and how they supplement each other is an area not yet well investigated (1) (5) (6).

Empirical knowledge and lessons learned from real-life projects are especially valuable contributions to the choice of techniques for other users and projects. However, there is little empirical knowledge on how different techniques function in practice, what benefits and challenges to expect and which criteria to consider when choosing a technique. Based on the results of literature searches, I found only one recent study in evaluation of techniques applied in documenting user requirements (1). The study addresses the difficulties in choosing suitable techniques and that the choice is frequently based on ad-hoc criteria.

This study aims to conduct an empirical investigation on the techniques applied to specify functional requirements in the development project for modernization of the Cause of Death Registry. The project was conducted and directed by the IT-department at the National Institute of Public Health, my employer, who kindly allowed me to study this project in my thesis. In addition to advantages such as ease of access to the projects data and members, the use cases technique was the standard technique to specify functional requirements at the IT-department and in this project (4.2.1).

Use cases are widely used in different types of projects to specify functional requirements for different types of products and solutions[1]. Hence investigating the application of the use cases technique was highly interesting for the objective of this study. In addition there are very few studies conducted in evaluation of use cases and their challenges in real-life practices. Results of the systematic literature review related to challenges concerning use cases reported that only 7% of the reviewed studies were based on industrial practices whereas 47% of studies were based on toy examples (5).

This study was conducted in a case study approach where I followed the project activities from November 2012 until February 2014. I was not a participant in the project and the data, entirely qualitative, were collected by interviews with different project members, reviews of projects documents and observations through some of the projects meetings. In addition the literature search was an on-going activity under the course of the thesis to find support from relevant academic studies. Furthermore books and other online resources such as reports and discussions on the topic of the thesis and research problems were studied. The research design is described in detail in 4.3.

The background and motivation for modernization of the Cause of Death Registry can be summarized by the following facts which are described in detail in Chapter three:

- Old technological platform with limited lifetime and need for a technical and structural upgrade
- The central role of the registry in the National Health Register Project (7)
- The need for an electronic solution for data capture

The old solution for the registry was physically placed and administrated at Statistics Norway (8).

Some of the results of the modernization project in form of changes and improvements in the new solution are the following:

- New technological platforms in accordance with the standards for health registers
- Automation of manual task related to quality controls and administration of the registry
- Generation of ad-hoc reports
- Improvements in security mechanism and auditing

The old- and new applications are shown in Figure 1 and Figure 2 respectively.

[1] Use cases are mainly applied to model functional requirements and are not used to represent non-functional requirements.

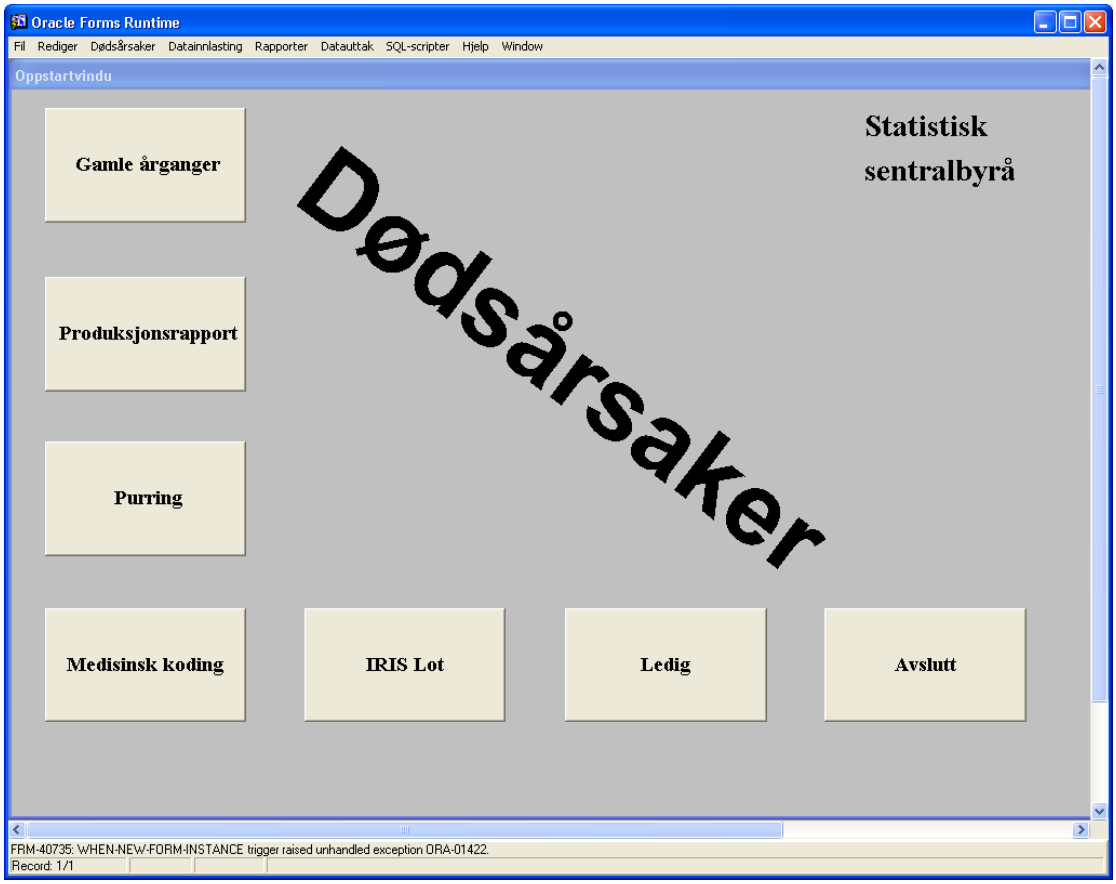

Figure 1 The old Cause of Death Registry application, Windows client (from internal documents presented in Table 5)

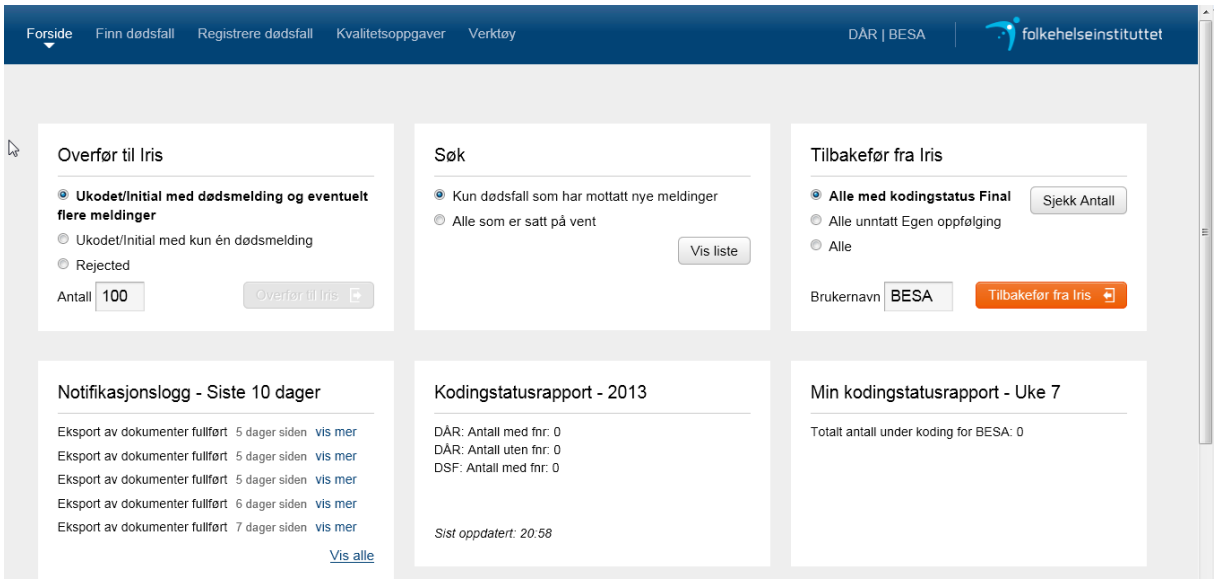

Figure 2 The new Cause of Death Registry application, web-based (screenshot from the application in the Test environment)

1.2 Research questions

Use cases (2.6.1) -the textual descriptions of the requirements - are commonly utilized in both small and large development projects to specify functional requirements for different types of products and solutions (5).

Despite the wide usage of use cases, important aspects and issues of use cases such as level of abstraction and guidance in inspection of use cases are still not well understood and agreed upon (5). Further, the usage of complementary techniques and other independently-developed solutions in industrial practices (9) addresses the limitations of use cases in providing sufficiently correct and complete specifications of requirements.

The limitations of use cases in being exclusively textual and difficulties in comprehension and validation of complex use cases with many alternative scenarios and extensions are addressed in (10). The study recommends applying supplementary techniques such as activity diagrams (11) to benefit multiple types of artifacts in capturing requirements and achieving higher quality artifacts. Another study (9) reports the benefits of using supplementary techniques like creativity workshops, visualizations, storyboarding and prototypes to obtain more complete and precise use cases.

Use cases (2.6.1) – the structured textual descriptions of requirements - were considered as the main artifacts to specify and represent functional requirements in the development of the Cause of Death Registry (CDR) (12) (13).

Understanding the application of use cases and the challenges encountered by their application were the main concerns of this study. In this regard, the following research questions were formulated:

RQ1: How was the use cases technique applied in different activities of the RE process in the CDR project and which challenges were encountered by use cases?

RQ2: How were other techniques than use cases beneficial to specify functional requirements? How were the limitations and challenges of use cases addressed by these techniques?

Further, the RE activities were narrowed to analysing, capturing and specifying requirements (defined in 2.7) in which all of the applied techniques in the CDR project were evaluated.

1.3 Thesis structure

The thesis is structured as follows:

Chapter two provides descriptions of the terms used in the thesis. Introduction and background of the CDR project is given in Chapter three, whereas the research methodologies applied in the study are described in Chapter four.

Results of analyses of use cases changes are presented in Chapter five. The application of use cases and challenges encountered by their application is described in Chapter six (answers to RQ1) while application of other techniques and their benefits are given in Chapter seven (answers to RQ2).

The results provided in Chapter six & seven are discussed in Chapter eight. The validity of the research is argued in Chapter nine. Chapter ten provides the conclusions.

Appendix and references are provided in Chapter 11 and 12 respectively.

2 Terminology

This chapter provides definitions of the central concepts and terms used in the domain of requirements engineering that are referred to in the thesis. The definitions are either based on IEEE[2] (14) standards or are derived from literature.

In addition the terms that are used in the context of the study to address certain activities are defined.

2.1 Requirements

A requirement is defined as the following (IEEE 610.12) (15):

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in 1 or 2.

2.1.1 Requirements classification

Requirements are classified in different ways. The most known and general categorization of requirements is functional- and non-functional requirements.

Functional requirements describe the functions (behaviours) of the system or what the system does. Non-functional requirements describe the quality attributes of the system or how the system does (15).

Examples of functional and non-functional requirements are given below:

Functional requirements: Applications, services, user interface
Non-functional requirements: Usability, security, performance, testability, maintainability

This study concerns only functional requirements as the techniques investigated in this study are mainly applied to specify functional requirements.

2.2 Requirements Engineering process

Requirements Engineering (RE) is concerned with analyzing, developing, verifying, validating and managing requirements, referred to as sub-processes of the RE process described below (16):

The strategies to conduct the RE process are dependent on the project's decision and affected by the software development process. Iterative and sequential requirements processes are examples of such strategies (17).

2.2.1 Requirements analysis

Requirements analysis refers to eliciting requirements from different stakeholders, analyzing the needs and business domain, uncovering possible conflicts between the requirements, and discovering missing requirements (15).

[2] Institute of Electrical and Electronics Engineers (IEEE) (14)

2.2.2 Requirements development

Requirements development is a collection of activities, tasks, techniques and tools to identify, analyze and validate requirements. It includes the process of transforming needs into requirements. The purpose of requirements development is to elicit, analyze, establish and validate requirements and the solution (or product) (15).

2.2.3 Requirements verification

Requirements verification concerns confirmation by reviewing requirements (individually and as a set) to ensure the characteristics of correct requirements are achieved (16).

2.2.4 Requirements validation

Requirements validation concerns confirmation by examining that requirements (individually and as a set), define the right system as intended by the stakeholders (16).

2.2.5 Requirements management

Requirements management involves activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service.

The purpose of Requirements Management is to manage requirements of the project's products, and components, to ensure alignment between those requirements, the project's plans and work products throughout a project's products lifecycle (development cycle and maintenance cycle) (16).

2.3 Stakeholders

Stakeholders are individuals, organizations or systems that are actively involved in the project, or whose interests may be affected as a result of the project execution or project completion.

Examples of stakeholders are: end-users, project manager, business analyst, software developer and quality assurance staff (15).

2.4 Software requirements specification

Requirements specification refers to a complete collection of requirements (incl. functional, non-functional requirements, design constraints, and attributes) of the software and its external interfaces for a specific project (or solution). The specification defines the product and may be used as a contract to build the product (17).

2.5 Functional requirements specification

Functional requirements specification is an essential part of the software requirements specification which describes details of the functions that the system needs. In addition the definition of the boundaries of the solution and the products' connection to external systems are included in this specification (17).

2.6 Techniques for specifying functional requirements

This section provides an introduction to some of the most utilized techniques which are also applied in the project under study.

2.6.1 Use cases

The idea of use cases was initially introduced by Iva Jacobsen in 1986 (18) where the term of use case referred to a technique for modelling and specifying functional requirements. Since then, this technique has been developed and improved by many others, e.g. (19) (20) (21).

A use case, simply defined, is a structured description of system behaviours under various conditions as the system reacts to a request or action from the stakeholder called primary actor. The primary actor initiates an interaction with the system to accomplish some goal (19).

Use cases are structured according to use case templates. An example of a use case template with description of its elements is depicted in Table 1 (22).

Use Case:
Properties
Goal:
Level:
Primary Actor:
Precondition:
Main Success Scenario
Step **Step** …
Extensions
Step Ref. Condition 1 Steps
Step Ref. Condition 2 Steps
…
Step Ref. Condition n Steps

Table 1 example of a use case template (22)

The header section contains the name and various properties of the use case.

Goal: addresses the very intent of the primary actor when executing the use case.

Level: indicates the goal-level of the use case. While different goal levels exist, the most important ones are summary, user goal and sub-function.

Primary actor: the actor who usually initiates the interaction in a use case to achieve a goal. In some cases the interaction is triggered by time or another actor on behalf of the primary actor.

Pre-condition: denotes a condition that must be satisfied before the use case can be performed.

Main success scenario
The most typical scenario in a use case where the primary actor's goal is reached and every involved stakeholder's interest is satisfied.

Extensions
All other alternatives to the main success scenario are described as extensions. Numbers in the extensions refer to the step numbers in the main success scenario, e.g. 4a and 4b indicate two different alternatives for step four. Each extension starts with a condition. When the condition is true the main success scenario branches to the alternate (or extension) scenario. The condition is followed by a sequence of action steps, which may lead to the fulfilment or the abandonment of the use-case goal and/or further extensions.

Exceptions for error handling in the main flow can also be included as extensions. Some use case templates have a separate placeholder for exceptions named Exceptional flows, e.g. the template used in the CDR project (11.1).

Examples of use cases of the CDR can be found in 11.3. Later in Chapter five and six, use cases of the CDR will be analysed and discussed in detail.

Use cases can have different formats and structures. The template and writing standards should be selected according to the needs of each individual project (19).

Writing effective and well-structured use cases is not an easy task and requires understanding of the technique and applying guidelines for best practices. Current practices of use cases have shown that it is easy to misuse them or make mistakes that can unintentionally turn them into "abuse cases" (22).

There are checklists (19) and measurement guidelines (23) to assess the quality of use case models in terms of the structure, consistency and completeness. A number of guidelines and templates in writing effective use cases have been proposed in (19) (20) (24).

2.6.1.1 Benefits of use cases

Use cases are reported to be used in different phases and activities of the requirements engineering process. Figure 3 shows the sub-processes that are identified through various studies and in what extent use cases have contributed to these sub-processes (5).

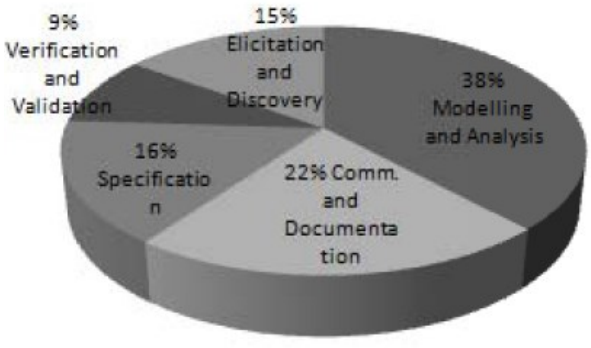

Figure 3 Sub-processes in RE process where use cases are beneficial (5).

Use cases are reusable and can be used across different projects. Building up libraries of use cases will facilitate capturing and documenting requirements (5).

Use cases are used by different stakeholders such as users, developers, testers and managers for different purposes and in different phases in the development process (5).

2.6.2 Use case diagrams

In contrary to the textual presentation of requirements described by use cases (2.6.1), use case diagrams provide a graphical overview of requirements represented by use cases.

The main constructs of use case diagrams are actors, use cases, the system boundary box and relationships between use cases (25).

Use cases are represented as ellipses while actors that interact with the system or are involved in the use cases as stick figures. Figure 4 shows a clip from Data processing use case diagram (5.3). The arrow between the use cases denotes their relationship which can be of three types: 1) The *include* relation to specify sub-use cases 2) The *extend* relation, to specify the use case that extends the

behavior of the base use case 3) The inheritance relation, to model generalization and specification between use cases (26).

Actors can be human or systems. The primary actor acts on the system or initiates an interaction with the system and uses the system to fulfill his or her goal.The secondary actor is acted on or invoked or used by the system and helps the system to fulfill its goal (19).

The scope of the system(s) that the use cases belong to, is outlined by the border in the diagram.

Due to the visual illustration, use case diagrams are said to be easy to understand and to communicate to stakeholders. Further, the overview of use cases and their relationships can be illustrated by the use case diagram (1).

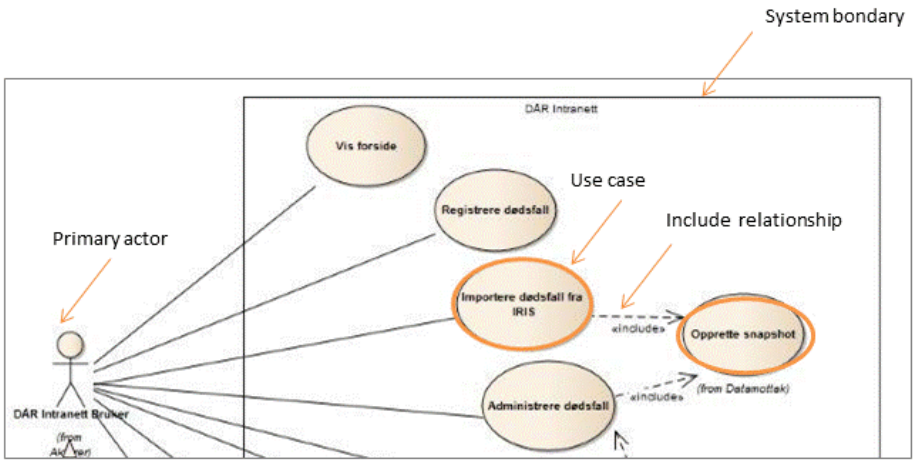

Figure 4 Clip from use case diagram of Data processing use cases (from internal documents presented in Table 4)

2.6.3 User stories

User stories were initially invented as the technique for description of the requirements in Extreme Programming (XP) software development methodology (27).

Cohn (28) defines user stories as descriptions of desired features told from the perspective of the stakeholders that need the features.

User stories do not include details of the requirements and are normally short. The details of the requirements are clarified and communicated through discussions before and during the development with the solution team and relevant stakeholders. The information about the acceptance criteria, alternative flows and scope can normally be incorporated in the acceptance test cases or other types of tests for the stories.

User stories are usually structured as follows, and can be written by any stakeholder with sufficient domain knowledge:

As a <user role> I want <goal> so that <reason> (28).

> *User role: represents the user that is performing the task or action.*
>
> *Goal: represents the action that is performed by the system*

Reason: represents the value and the reason of the requirement. It helps the solution team to understand the need and possibly find alternative solutions that could offer the same solution for less effort.

Some of the important characteristics of user stories are the following (27):

- User stories are not detailed requirements specifications but are rather helpful hints about the needed functionalities.
- User stories are short and easy to read, understandable to developers, stakeholders, and users.
- User stories represent small increments of valued functionality, that can be developed in a period of days to weeks
- User stories are not carried in large, unwieldy documents, but rather organized in lists that can be more easily arranged and re-arranged as new requirements are discovered.
- User stories are not detailed at the outset of the project, but are elaborated on a just-in-time basis thereby avoiding too-early specificity, delays in development, requirements inventory, and an over-constrained statement of the solution.

User stories were mainly developed to analyze and capture the interaction design requirements in the CDR project. Application of user stories therefore did not match their usage in agile development process (27) (28) (29).

Example of a user story developed in the CDR is:

As an executive officer, I want to select how many records I want to export to IRIS. This is for increasing the efficiency of the work and being able to finish the commenced work. This task can be executed in any time of the work day. (From internal documents presented in Table 4)

2.6.4 State diagrams

State diagrams or state machine diagrams (30) are used to represent the discrete behavior of a single object during its lifetime. The behavior is modeled by the unique states that the object goes through in response to the happening events (31).

The main constructs in state diagrams are:

1- States: A state represents a stage in the behavior pattern of an object. States can be simple, composite or submachines (32).
2- Transitions: A transition is a change from one state to another and will be triggered by an event that is either internal or external to the object. Transitions happen when the object or entity reacts to the events that occur.
3- Guards: a guard is a condition that must be to true so that a transition is executed.

Figure 5 shows a clip of the CDR state diagram (11.10) developed to model states of a record, and transitions between the states through the data processing flow.

The lanes depict the main categorization of the states: 1) Not-finished (do not wait) 2) Not-finished (under coding) in the diagram. Not-finished (under coding) category is used to model the records states in IRIS.

There are transitions between State 4 which represent Un-coded state to State 6 and to State 5 which represent Under-coding status. The transition between 4 and 6 occurs by export of records to IRIS while transition between 4 and 5 occurs if a record should be manually processed.

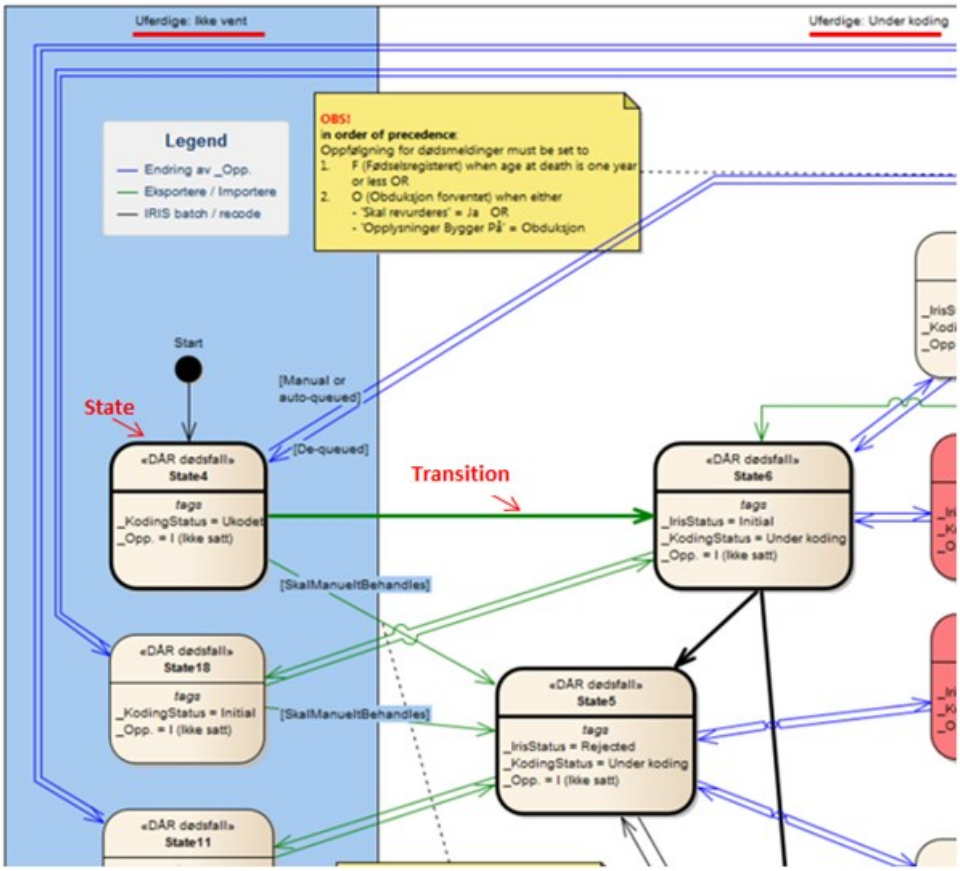

Figure 5 Clip of the CDR state machine diagram (11.10) (from internal documents presented in Table 4)

2.6.5 Sequence diagrams

The main purpose of sequence diagrams – also called interaction diagrams -is to illustrate the interactions between objects (e.g. actors, systems) in the sequential order that those interactions occur.

Sequence diagrams are used to represent the dynamic behavior of the system and are especially capable of capturing the order of interactions performed between the objects. The first interaction is represented in the top leftmost side and the last interaction in the bottom rightmost side of the diagram (33).

Sequence diagrams can be used in different phases of RE process; high-level diagrams for analysis and communication of requirements with the stakeholders and more detailed diagrams for design and development purposes (34).

The main constructs of sequence diagrams are as the following (35) (36):

1- Participants: represent objects or entities that interact in the diagram.
2- Messages: represent communications between participant objects. Messages can be simple, timeout, synchronous or asynchronous.
3- Axes: horizontal axes represent the participant that is acting, vertical axes represent the lifelines.
4- Activations: represent the time an object needs to complete an interaction.

A clip of the CDR sequence diagram developed to model the interactions between the CDR and IRIS is shown in Figure 6.

The records are selected (after certain criteria) by the user and exported to IRIS (the message named Insert (Lot) in the figure). The states of the records are updated (Update (state) message in the figure) by the export. The interactions of the export function continue until the report (notification message) is sent back to the user (shown by Activation component in the figure).

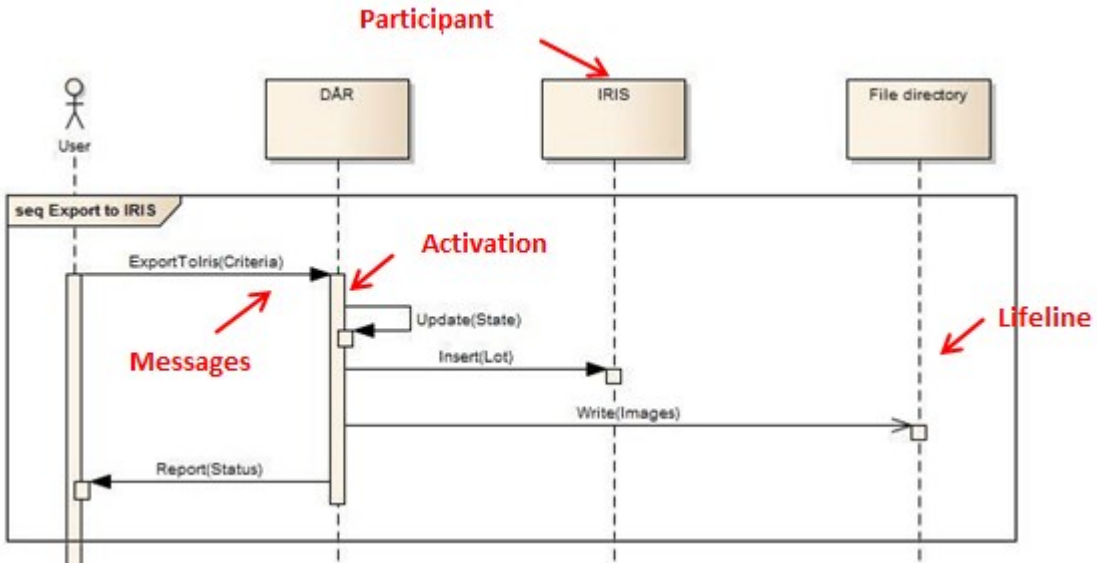

Figure 6 Clip of the CDR - IRIS sequence diagram (from internal documents presented in Table 4)

2.7 Definitions in the context of the study

The terms that I have used to address specific activities in the context of this study are defined in this section. The definitions are derived by investigation of the RE activities in the CDR project (Chapter three).

2.7.1 Preliminary analysis

This activity refers to high-level analyses and investigations of the business domain, understanding and identifying the needs of the solution in the early stages of the requirements process.

2.7.2 Analyzing requirements

This activity refers to the process of understanding and analyzing requirements by various techniques such as interviews, use cases or user stories. Domain knowledge, information received from relevant stakeholders and available data sources such as documentations of the existing solution are examples of input to this activity.

2.7.3 Capturing requirements

This activity refers to the process of discovering or extracting necessary details of the requirements so that the target features can be developed and completely implemented. The process of elaboration of the requirements mainly occurs during the development of the features.

2.7.4 Specifying requirements

This activity refers to documenting requirements so that the specification includes the necessary descriptions and details of the functional requirements or the technical solutions that satisfy the functional requirements.

2.8 Terms related to the Cause of Death Registry

2.8.1 Death certificate

All deaths in Norway are issued by either a doctor- or hospitals on a prescribed form, called death certificate. The certificate contains information about the deceased (personal identification number, residence, etc.) and the information that might be of importance when cause of death will be determined. Death certificates are sent to probate and then to the municipal doctor (kommunelege) in the municipality where the death occurred. Death certificate is one of the six types of messages that are sent to the CDR (37) (12).

2.8.2 Death messages

Six types of messages related to deaths sent to the CDR are1) death certificate, 2) police report, 3) autopsy report, 4) additional information, 5) death abroad and 6) others.

2.8.3 Death record

A death record (hereafter called record) refers to the deaths that are registered in the CDR.

3 Modernization of the Cause of Death Registry

In this chapter the case of the study – the development project for modernization of the CDR – is presented.

The CDR is one of the central health registers in Norway (38). The common goal of the modernization of central health registers – directed by the national health register project – is to provide continuously updated, reliable and privacy-protected knowledge of the population's health status, and better quality of treatment. Good health registers – better health (39).

In the following, the CDR, the background for modernization of the CDR, the National Institute of Public Health (NIPH) and the CDR project are presented.

3.1 The Cause of Death Registry

All deaths are reported by doctors who are required to complete a death certificate (2.8.1). Death certificates are collected by the Cause of Death Registry for coding of information based on an international system which determines the underlying cause of death to be used in the cause of death statistics (12).

The coding system used is the International Classification of Diseases (ICD) (40). Using this system, mortality in different countries can be compared and the development of various causes of death can be followed over time.

The cause of death data is useful and valuable for many reasons and usages; amongst the others the contribution to the international diseases development observations and to research purposes both abroad and at home can be mentioned (12).

A part of the causes of death report for 2012 provided by Statistics Norway is shown in Figure 7 (13).

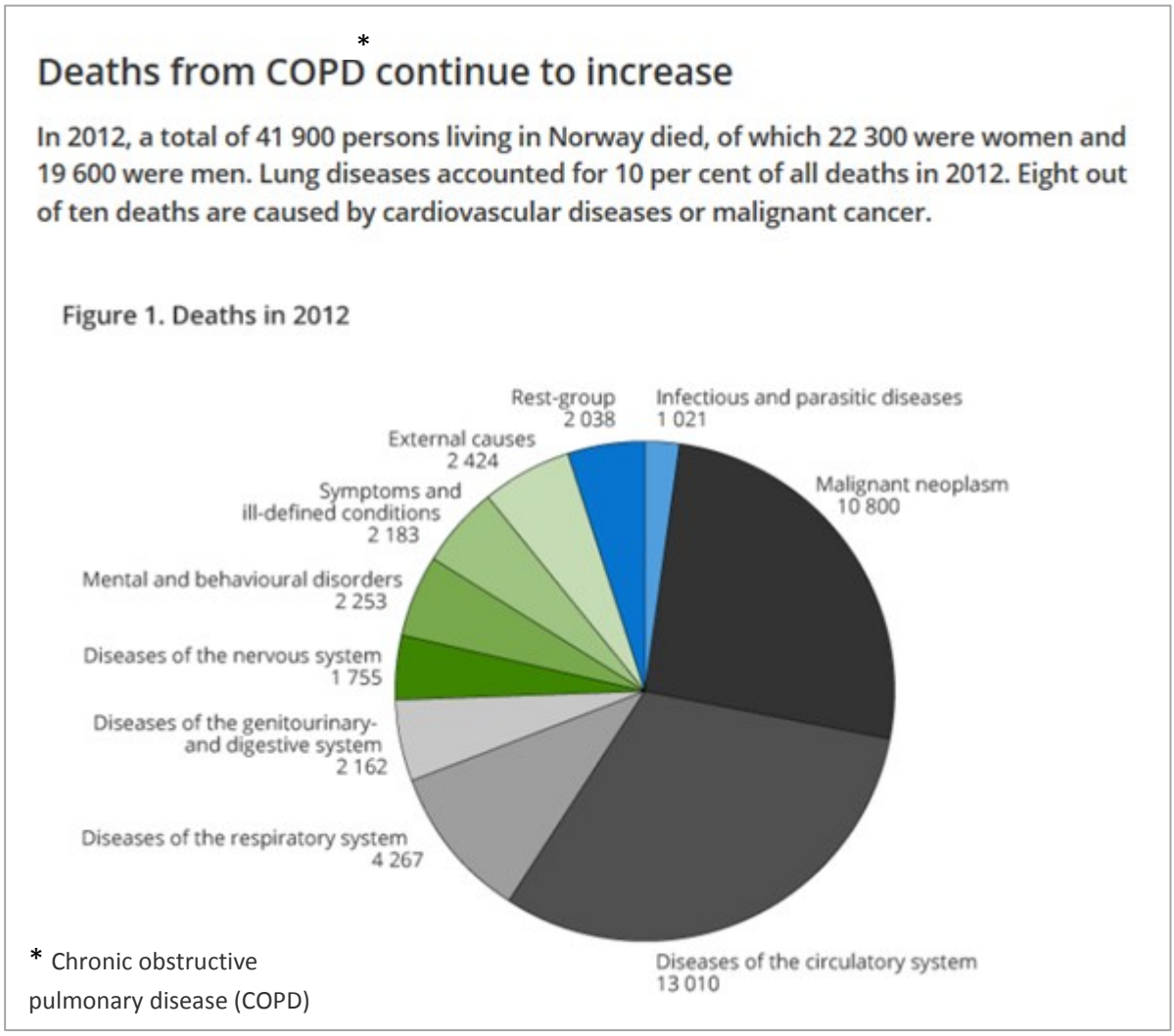

Figure 7 The Causes of Death report 2012 (13)

The Section for Health Statistics at Statistics Norway was the Data Processor[3] for the registry from 1925 to 2012, while the NIPH was the Data Controller[4] (12).

3.2 Why a new CDR?

The information given in this section is derived from the report prepared by the working group and the pre-project report prepared by the project manager (from internal documents presented in Table 5).

The National Cause of Death Registry in Norway was established at Statics Norway in 1925. The Section for Health Statistics at Statistics Norway had the responsibility of administering, data processing and producing reports until 2001. In 2001, the new health register law ((7), § 2) and the registry regulation recognized the NIPH as the responsible for administrating the registry. However NIPH and Statistics Norway agreed that the tasks were still carried out at Statistics Norway. The agreement was formalized through a contract.

[3] Data processor: databehandler (71)
[4] Data controller: databehandlingsansvarlig (71)

The IT-solution of the CDR was initially developed in 1998. The solution was changed and extended by new capabilities needed over time like integration with data systems ACME[5] and IRIS[6]. The specification of the technological platform and software for the existing (old) IT-solution is found in 11.7.

The consideration for a new registry organization was realized in 2012 when the CDR received a central role in the National Health Register Project (7). In addition, the need for an electronic solution for data capture to attain an updated registry with higher data quality strengthened the need for the reorganization. As the project owner and data controller for the CDR, the NIPH recognized the need to have a better control on the operation and administration of the CDR, including the tasks that Statics Norway performed as data processor.

In January 2012, the Ministry of Health and Care Services assigned the NIPH to propose a process plan and recommendation for termination of the current contract (between Statics Norway and the NIPH) and for transfer of the CDR and tasks to the NIPH.

A time plan and recommendation for the transfer solution were proposed by the working group with representative members from the NIPH and Statics Norway.

The time plan was suggested based on the following premises:

- The CDR's existing IT-solution needed an upgrade within summer 2013.
- Statics Norway planned to move to new locations in February 2014, therefore establishing a new operational environment for the CDR would be inappropriate if the solution was not intended to be a permanent solution.

For the CDR solution needed by September 2013, it was recommended to develop a new minimum solution compatible with the common development platform and standard technologies used in other health registers at the NIPH. The architectural aspects of the solution should be validated by summer 2013. Development and finalization of additional functionalities and establishment of the electronic data capture were postponed to after summer 2013.

Based on the recommendations from the working team and with wide considerations concerning technological platforms, costs, risks and solution lifetime, the NIPH and Statics Norway agreed on building a new solution based on the standard technology and platform for health registers at the NIPH.

3.3 The Norwegian Institute of Public Health

Development of the new solution for the CDR was organized and directed by the IT-department – called IT and e-Health -at the NIPH.

The NIPH is placed directly under the Ministry of Health and Care Services, alongside the Norwegian Directorate of Health, the Norwegian Board of Health Supervision and the Norwegian Medicines Agency (41).

The NIPH is responsible for 10 of 17 central health registers in Norway. The central health registers are nationwide and required to be reported to. National vaccination registry (SYSVAK),

5 ACME: The Automated Classification of Medical Entities program (72)
6 IRIS: The application for classification of underlying cause of death (61)

Cardiovascular registry (Hjerte- og karregister), Cancer Registry and the Norwegian Patient Registry (NPR) are as such. The core registries are primarily used for health monitoring in terms of health statistics and readiness, quality of healthcare, research, administration and management (42). Overview of the central health registers can be found at (38).

The IT and e-health department at the NIPH has over 60 employees working on development, management and administration of various systems, support and infrastructure. In addition a large proportion of core systems, especially central health registers, are self-developed at the IT-department (43).

3.4 CDR project organization

Being a large project with many stakeholders, the CDR project was organized as a "program" composed of four sub-projects as depicted in Figure 8[7]:

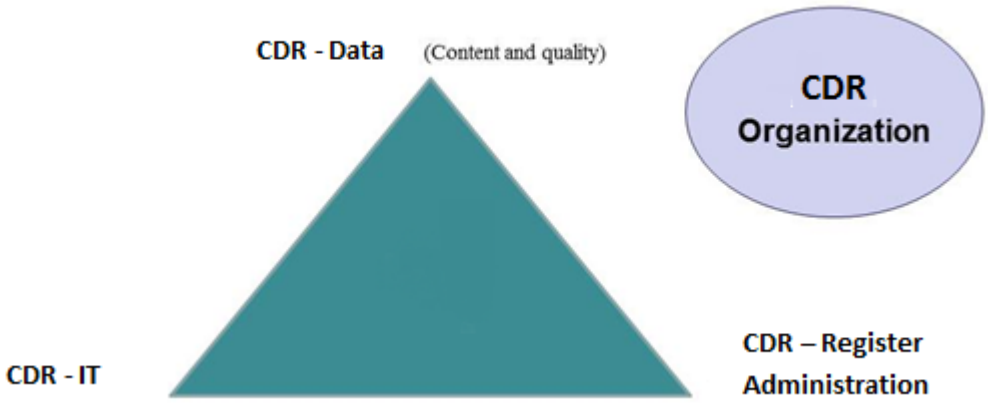

Figure 8 The CDR program organization (from internal documents presented in Table 5)

1- CDR-Data: had the responsibility for controlling and directing the decisions concerning content and quality of the data in the CDR.
2- CDR-organization: had the responsibility for creating and staffing the organizational unit of the registry who performed the daily data processing routines of the registry.
3- CDR-registry administration: had the responsibility for ensuring timely and correct performance of registry administration processes.
4- CDR-IT: was responsible for the development and implementation of the new IT-solution to support the functional needs of the registry.

3.5 Project implementation plan

Based on the recommendation of the working group, the implementation of the new CDR was divided and planned in two phases; 1) preparation and development of an initial solution to cover the fundamental requirements related to the core tasks of the registry in the first phase followed by

[7] The original figure was on Norwegian. I chose to translate the terms to provide better understanding.

2) the development of additional functionalities and formal transfer of the registry administration to the NIPH in the second.

In order to achieve the time plan, reliable estimations of tasks vs. time and necessary resources were critical for mitigating the risk of a possible delay. Hence a pre-project was organized to establish a detailed time plan for the project activities and to start the technical development from autumn 2012.

3.5.1 Time plan

The first phase of the development of the CDR started in November 2012 and was finished on summer 2013.The second phase started in august 2013 and continued until January 2014. The timeline for the main tasks of each phase are shown in Figure 9.

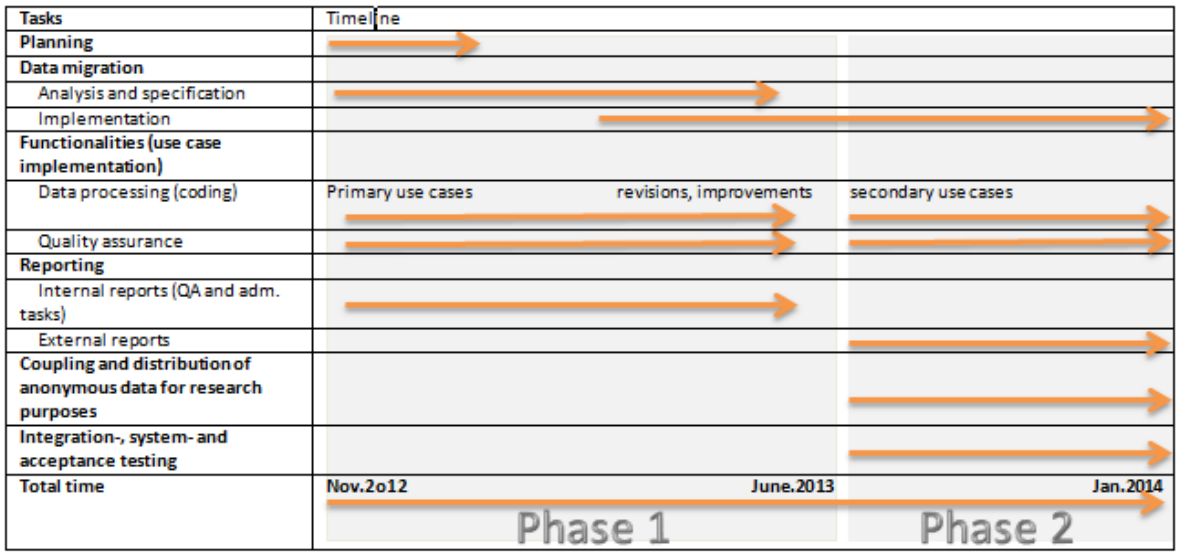

Figure 9 The CDR project time plan (from internal documents presented in Table 5)

3.6 IT Solution team

The IT- team (also called solution team) for development of the CDR consisted of nine members with the following roles:

Project manager (1)
External program manager assistant (1)
System architect (1)
Internal developers (3)
External developers (3)

3.7 Stakeholders and users

3.7.1 Stakeholders

Death related data are reported by many public and private institutions to the CDR. Likewise cause of death data are used for many purposes, amongst the others for research, statistics reports, to contribute to improvement of life quality of the population and global observation of diseases. Figure 10 shows the context model for the CDR solution (created by me) and different stakeholders that

interact with the CDR. The model is created based on the reviews of the projects documents related to identification of stakeholders.

The medical stakeholders send death related data to the Data capture unit of the CDR who registers and forwards messages to the CDR database.

The CDR sends statistics reports to WHO (44), Eurostat (EU) (45) and Statics Norway according to the required formats and contents of the reports.

The CDR exchanges information with several central national registers such as population register, cancer- and birth registries to ensure the quality of the registered data and to receive additional information on deaths.

The civil stakeholders (that do not send death related data to the CDR), have direct communication with the CDR administration.

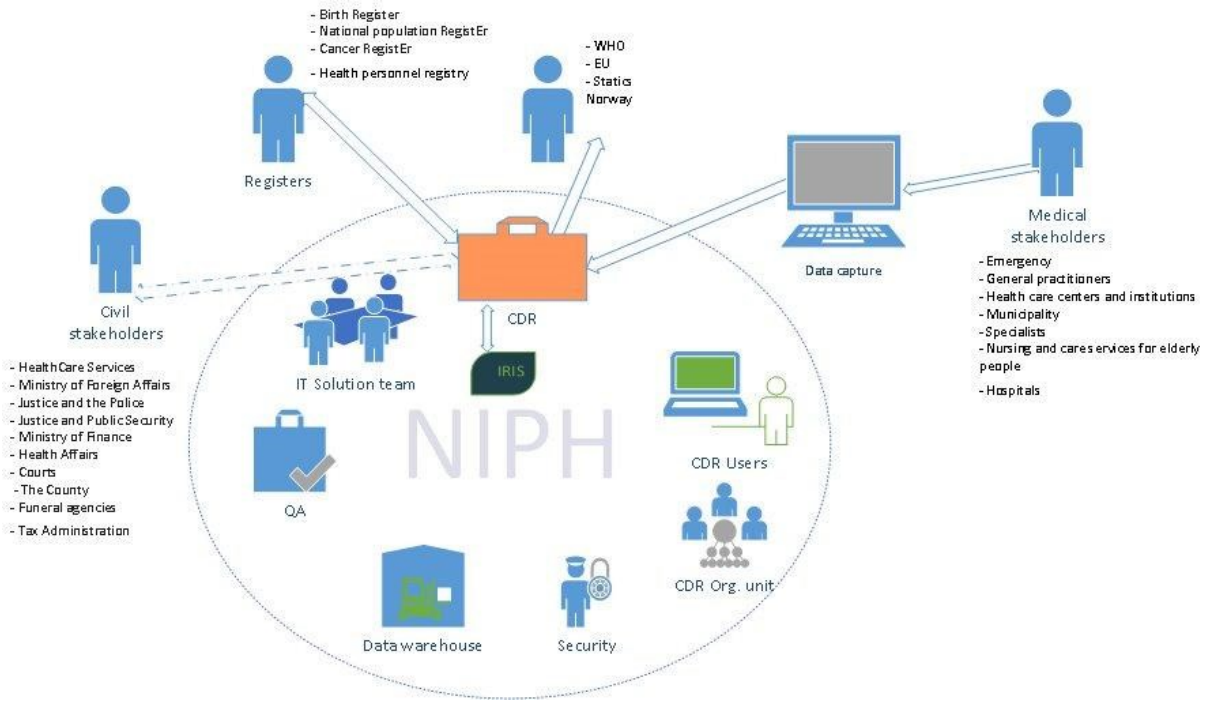

Figure 10 The CDR context mode

3.7.2 Users

The primary users of the CDR who conduct the tasks of registering data, classifying causes of deaths, controlling the quality of data and preparing reports are referred to as users in the thesis.

3.8 Development iterations

The information given in this section is derived by following project activities in Team Foundation Server (TFS) and the project SharePoint site (3.10), in addition to knowledge gained from interviews with the members of the solution team.

The development process was incremental and conducted in iterations. The collected data in the study are from the first five iterations.

At the beginning of iterations, the use cases to be developed and other necessary tasks (see the tasks in Figure 11) were decided by the system architect in cooperation with the solution team.

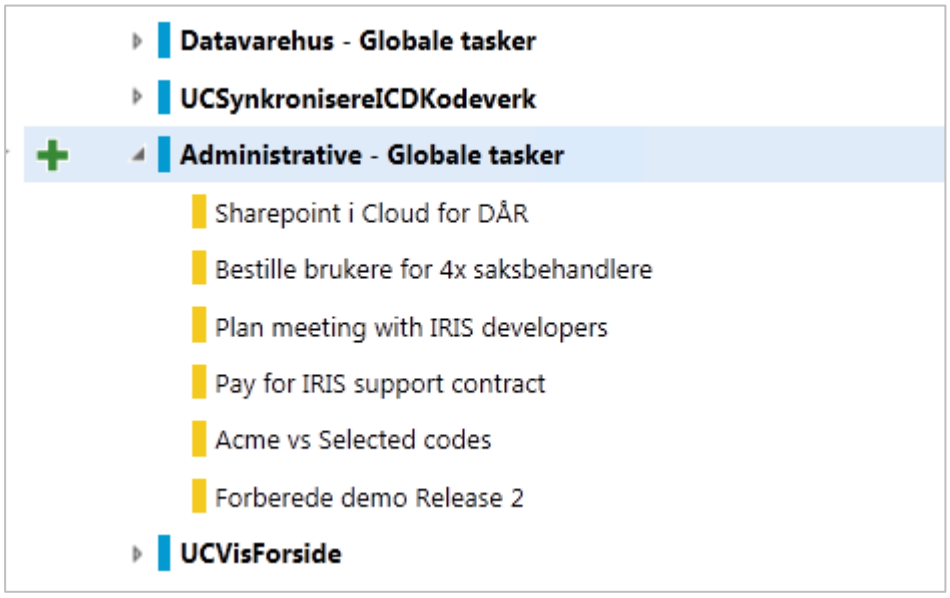

Figure 11 Use cases and tasks in release 2 (screenshot from the tasks in TFS, 3.10)

The plans of iterations and features to be developed were presented to the users at the beginning of iterations (from internal documents presented in Table 5).

The duration of iterations varied from 18 to 40 workdays depending on the volume of the tasks. All the tasks and use cases belonging to iterations were registered in Team Foundation Server (TFS) (3.10) with an assigned time and a responsible. The status of the tasks and remaining time were updated during the iterations. The Burn down graph was used to monitor the progress of the achievement of the tasks and the remaining work versus available time.

The items of the first iteration and tasks related to one of the use cases (*UC Export Records To IRIS*) in TFS are shown in Figure 12.

The users (or stakeholders) related to the features or functions being implemented were informed and involved in decision making and testing during the development of the functionalities.

At the end of iterations, the implemented functionalities were presented to the stakeholders in a demo.

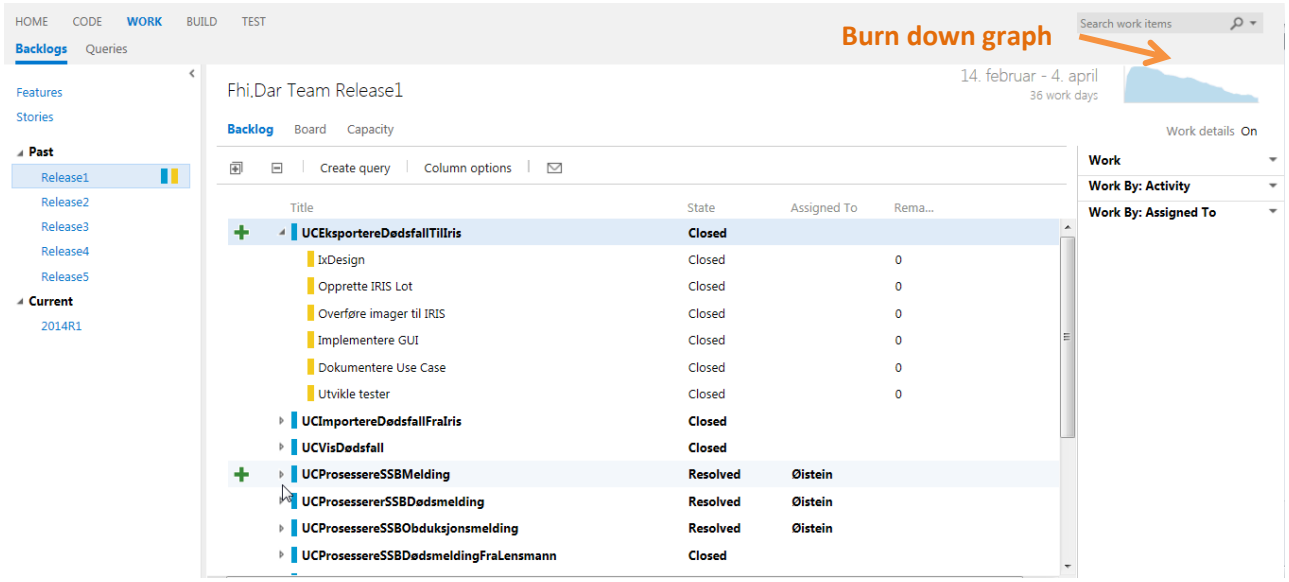

Figure 12 Use cases and tasks in Release 1 (screenshot from TFS, 3.10)

3.9 Testing

Automatic tests were created during the development of use cases by the developers. Manual tests were conducted by the users and solution team:

- Relevant users and stakeholders (the ones that had relations to the functions) tried the functionalities after implementation. Errors and feedbacks from these tests were communicated to the solution team.
- Tests conducted by the solution team (project manager and system architect) were structured with predefined test data and expected results.

Further, test scripts based on use case specifications were created when the implementation of the use cases was completed.

3.10 Technological platform and tools

The specification of the software development technologies used in the development of the CDR is provided in 11.6.

The following tools were used by the solution team for project management:

- Team foundation server (TFS) (46): was used to manage source codes and development iterations by registration of activities, responsible person and with estimated time to complete. The Burn down graph in TFS was used to monitor the progress of completed tasks against the remaining time.
- Enterprise architect (47): was used as the modeling tool. The artifacts such as use cases-, sequence-, component- and activity diagrams were created by this tool.
- Microsoft SharePoint (48): was used to manage projects documents published on the internal projects website.
- Microsoft office products

3.11 Challenges

The most critical challenges in the project, understood and derived from interviews with the system architect and project manager and reviews of project status reports, were the following:

- Time pressure: the time schedule of the project was tight as the existing solution at Statics Norway could not be supported after December 2013.
- Scope: Decisions and agreements on the scope of the solution and functionalities were a time-demanding process due to the complexity and size of the project with regard to the number of stakeholders. Political issues, bureaucracy, legal decisions and clarifications were among the factors that influenced the decisions of scoping.
- The new organizational unit: Establishing a new organizational unit to administrate and support the CDR at the NIPH was challenging due to the new roles, tasks and employees that followed the registry from Statics Norway.
- Communication and interaction with Statics Norway: the solution team needed often to contact users to clarify decisions, issues or to ask for missing information. This was a time-consuming process due to the distance, organizational differences, differences in users' skill levels and conflicts in the time and availability of the required resources.
- Technical challenges related to the technology shift in the new solution.

3.12 Risks

Some of the risks, obtained from Risks status presentations, concerning requirements specification under the progress of the CDR project are summarized in Table 2.

Risk	Risk mitigation tasks
Not all the necessary details in the requirements are discovered due to reasons like: • Distance between the solution team and users • Users available time to contribute with requirements specification • Differences in professional languages (IT language and CDR terms and concepts)	• Higher user involvement • Well documented and specified requirements (in form of use cases and user stories) • Users physical presence at NIPH at least one day per week
Not all the dependencies between the CDR and external systems / stakeholders are identified (incl. knowledge of stakeholders, systems, processes, etc.)	Same as above
Delay due to decision makings, particularly concerning scope of the system	Involve the stakeholders with knowledge and right to make decisions

Table 2 Risks related to the requirements specification in the CDR project (from internal documents represented in Table 5)

4 Research methods

In this chapter, the research methodologies and their application are presented.

4.1 Research methods selection

Case study is the main research methodology in this thesis. This method is appropriate for studies that investigate contemporary phenomena in their context (49), which applies to the research problem of this thesis and the project under study.

Further, the share of empirical studies in the field of software engineering is considerably small and, experimental research and quantitative approaches have been more frequently applied in empirical studies (49). Another motivation to choose the case study approach was found in the results of a systematic mapping study in empirical evaluations of requirements specification techniques (6) which revealed that few case studies in this area have been conducted.

Ethnography (4.5) is considered as the secondary research method. By passive participation in some of the project meetings and close distance to the solution team, the communication cultures between the members of the solution team and with the stakeholders were understood.

4.2 Qualitative case study

The case study methodology is well suited for many kinds of software engineering research, as the objects of study are contemporary phenomena, which are hard to study in isolation (49).

A case study is described to be an observational research method where the researcher monitors a project and collects data over time. The researcher cannot interfere in projects activities in the research period (50).

Case studies can be exploratory, descriptive or explanatory (51). Another categorization defines three types of case study depending on the research perspective, positivist, critical and interpretive (52).

Qualitative data obtained by interviews and observations in form of textual descriptions, diagrams and figures constitute the main data sources in the case study research. Qualitative data are analyzed using categorization and sorting (52).

The boundaries of data collection domain is decided by the unit of analysis, which refers to the actual unit being studied such as a company, a project, a team, individuals or an event (51).

Results of case studies are said to be difficult to generalize, and more dependent to the researcher's interpretation and bias. These weaknesses can be reduced by applying proper research methodology practices as well as reconsidering that knowledge is more than statistical significance (49).

Case studies are relatively easier to conduct compared to other research methods and can provide valuable information on aspects like complexity, scale, unpredictability, and dynamism (53).

In this study, I tried to follow the guidelines for conducting case studies recommended by (49).

4.2.1 Selection of the case

The most important criteria for selection of case of the study – the CDR development project - were:

- A development project which could be followed from the beginning
- Requirements process that could be followed through software development and implementation with respect to the available time for the thesis
- The use cases technique was applied to specify requirements

The CDR development project satisfied the above mentioned criteria in addition to have advantages such as:

- Ease of access to information (both to the project data and personal)
- Close distance to the project and ease of participation in project activities when needed
- Familiarity with some of the members of the solution team

4.2.2 Data collection methods

In case studies, data can be collected by different methods and from different sources. Six recommended sources of data for case studies are (54):

- Documentation
- Archival Records
- Interviews (or surveys)
- Direct observation
- Participant observation
- Physical artifacts

It is important to use data from multiple sources to mitigate the effect of interpretations that are gained only from one source. Also, conclusions that are drawn and supported by different sources are more reliable (49).

Data collection in this study was started in November 2012, almost at the same time as the CDR project was initiated.

The main data collection methods in this study are described in the following:

4.2.2.1 Interviews

The interviews were generally semi-structured and informal. In addition to the questions that I had predefined, new ideas and topics for discussion were brought up and explored during the interviews. An example of an interview guide is found in 11.2.

The duration of the interviews varied from half- to one hour. I wrote down notes under the interviews and tried to use the exact words and sentences used by interviewees in my notes. The notes were structured and rewritten with details after the interviews.

The interviewees were selected with regard to their role in the project. The guideline found in (49) was followed in planning and conducting interviews.

The interviews conducted under the course of the thesis are listed in Table 3.

Interviews (number) 01.2013 – 02.2014	
Interviewee	Topics
Project manager (3)	• Project status • RE process and techniques • User stories • Interaction design • State diagram
Developer 1, Data capture (1) Developer 2, Data processing (1) Developer 3, ETL (1)	• Use cases under development • Requirements analysis • RE process techniques • Involved requirements artifacts • Experiences with use case modeling • Benefits and challenges of use case modeling • Communication and contact with users • Use cases changes and reasons for changes • Testing
System architect (8) (in form of meetings every 2. or 3. week)	• Project status • Iteration plans, time- and activity schedule • Use cases under development • Interaction design • Issues • Risks • Stakeholders • Domain model • IRIS • Use case diagrams
Assistant program manager (1)	• Correct description of processes • Stakeholders • Direct communication and user involvement • Project organization • Challenges in communication between IT and other stakeholders • Importance of scoping (what to/not to implement) • Importance of decision makings • Priorities • Effective description of requirements • Available time and resources

Table 3 Data collection, interviews

4.2.2.2 Document reviews

All the available electronic documents in the project's intern website have been skimmed through to find related, beneficial or supportive data to the research problems of the study.

The requirements artifacts that are studied and referred to in the study are summarized in Table 4.

The reviewed documents and what they described are summarized in Table 5.

In addition the project status, tasks and activities in the development iterations were regularly followed in TFS and the projects SharePoint site (3.10).

Requirements artifacts (number)	Description / purpose
User stories(18)	Created to analyze and capture design requirements
Design prototypes	Approx. 10 prototypes were created and modified 11 times
Modeling diagrams (existing system) (4)	Message flow, Registry processes, Work flow, Data flow
Task-oriented workflow (new system) (1)	High-level task-oriented workflow
The state diagram (1)	Death record states in CDR and IRIS and transition between the states
Sequence diagrams (2)	CDR and IRIS integration (Import from /Export to IRIS), high-level process flow
Domain models (3)	Main domain model in addition to domain models for data warehouse, delivery database and for message processing
Use cases (12)	Selected use cases from three functional are (in order to select these use cases, all of developed use cases were reviewed)
Use cases snapshots & versions (48)	Snapshots and available history log from Jan.2013 – Jan.2014[8]
Scanned paper notes, photos of whiteboards sketches	From various meetings
Use case package diagram (1)	Functional areas and their use cases
Use case diagrams (3)	For each functional area
Use case diagrams snapshots (12)	Four snapshots for each use case diagram

Table 4 Data collection, requirements artifacts

[8] The number of snapshots and versions varied depending on the change frequency.

Documents (number)	Purposes / content
Workgroup report (initial report before the start of pre-project) (1)	• Project background • Objectives of the new solution • Issues with the existing system • Alternative solutions • IT considerations and recommendations
CDR-IT pre-project report (1)	• Project background • Organization of the project • Responsibilities and roles of the units in the program • CDR program (Data, Organization, administration and IT) • Solution suggestions and recommendations • Risks
CDR program status meetings presentations (5)	• Overview of tasks and activities (related to RE activities) • Issues, challenges • Progress vs. time • Risks
CDR-IT Status reports (2)	• Development progress • Iterations- and future plans • Overview of activities (related to RE activities)
Risk status presentations (5)	• Understanding the project circumstances and tasks with high risks
CDR project plan	• Project time plan and activities
Milestone plan for development (1)	• Iterations plans • Activity plan
List of stakeholders (1)	• Stakeholders
User manual of the old system (1)	• Requirements background
IRIS user manual (1)	• Background information
IRIS Integration presentation (1)	• IRIS and IRIS integration
CDR system documentation (1)	• Use cases and other requirements
User meetings presentations (7)	• Presentation of workflows, scenario walkthroughs • Domain model • Changes in the new solution • Presentation of use cases (by use case diagrams) • presentation of similar solutions in other health registers
CDR regulation (Forskrift) (37)	• To understand the requirements imposed by the regulation
Interaction design (4)	• Activity plan • Power point presentations for user meetings • Status updates • Application of user story technique • User stories in RE process

Table 5 Data collection, reviewed documents

4.2.2.3 Passive participation

I was invited to the meetings the project manager or system architect considered as informative and beneficial for the thesis. Participation in the meetings helped understanding the viewpoints of different project members. In addition extra details through the conversations between attendants were noted.

The knowledge obtained through observations and passive participation contributed to understand the communication culture between the project members and the viewpoints of different roles. The data collected through this method constituted the minor part of the collected data.

4.2.2.4 Informal conversations

Short conversations with project members during café-breaks and other occasions provided brief updates of the project status, current challenges and recent happenings.

4.2.2.5 Literature search

An extensive literature search was conducted under the course of the thesis to find relevant studies related to the problem area of the thesis. Google scholar search was mainly used for literature search.

4.2.2.5.1 Keywords composition

The keywords were selected with respect to the accepted and commonly used terms in the domain of the research problem, namely empirical investigations and evaluations of techniques applied in specifying functional requirements.

As the results of the searches usually included studies of non-functional requirements, the searches were explicitly repeated without “non-functional” in order to filter these studies. This was done by setting “non-functional” as excluded in the advanced search settings in Google scholar.

4.2.2.5.2 The overall search

The overall search composed of the following keywords:

General words: empirical, software, development
Exact phrase: "functional requirements specification"

4.2.2.5.3 The detailed search

As the use cases technique is considered as the main technique of specifying the functional requirements in the CDR project, the term “use case” was added to the keywords above. The keywords compositions are shown in Table 6.

Keywords composition		
Always included	Empirical, software, development, functional requirements, use case	
		“Non-functional” excluded
Variable keywords in exact phrases	Nr of matches	Nr of matches
Specification techniques	45	16
Documentation techniques	6	2
Requirements engineering	958	260
Requirements analysis	724	197
Requirements documentation	131	21
Requirements process	201	35

Table 6 Literature search keywords and composition

4.2.2.5.4 Priorities

The results of the searches have been prioritized according to the following criteria:

- Empirical studies in investigation and evaluations of use cases

- Studies with investigation of challenges and issues in use case modeling
- Use cases compared to other functional requirements techniques
- Evaluation of functional requirements specification techniques
- Conducted in 2009 - 2013

4.2.2.5.5 Selection of literature

The approach to select relevant literature was as follows:

1- The title and abstract of the search results were studied.
2- In case of relevancy, the results and conclusions were investigated.
3- The selected articles were completely studied and analyzed.

4.3 Research design

The design of the study and how to explore the problem domain was not clear from the beginning. However, through the collected data (4.2.2) and better understanding of the project's circumstances (Chapter three) the following approach was selected (Figure 13):

I was given access to the project's documents and management tool (3.10) through which I was able to follow project activities, development plans, use cases to be implemented and other tasks of the development iterations. In addition I interviewed members of the solution team, participated in some of the projects activities and studied literature under the course of the study.

The first approach to understand use cases application was to select use cases from different functional areas and follow their development through their life cycle (Chapter five). This process continued during the first five development iterations where I saved snapshots of some of the selected use cases to investigate use cases evolution and analyze changes. This is illustrated by the development timeline and the five iterations in Figure 13. Use cases were followed from different iterations depending on when their development was started. Some of the use cases were selected later and after their development was finished. For these use cases, the changes history (versions of use cases) available on the source code management tool, TFS (3.10) was studied. In Chapter five detailed information about snapshots and versions are provided.

Various use cases with respect to their involved stakeholders (particularly primary actor), type of requirements, available changes history and advices from the solution team were followed and studied. Based on these criteria and categorization of use cases (described in 6.3), 12 use cases were selected to be analysed (Chapter five).

Further the relation between use cases development and the RE activities became more apparent and I aligned the different forms of use cases during their evolution (initial, under development and final) with these activities.

Investigation of the other techniques and their artefacts, and in which RE activities they were applied was the second approach to understand the benefits of other techniques and whether (or how) they addressed challenges of use cases. This is illustrated in Figure 13 by the dashed lines between the other requirements artifacts such as user stories, design prototypes and the state diagram which were developed in parallel with the development of use cases and contributed to use cases and specifying functional requirements.

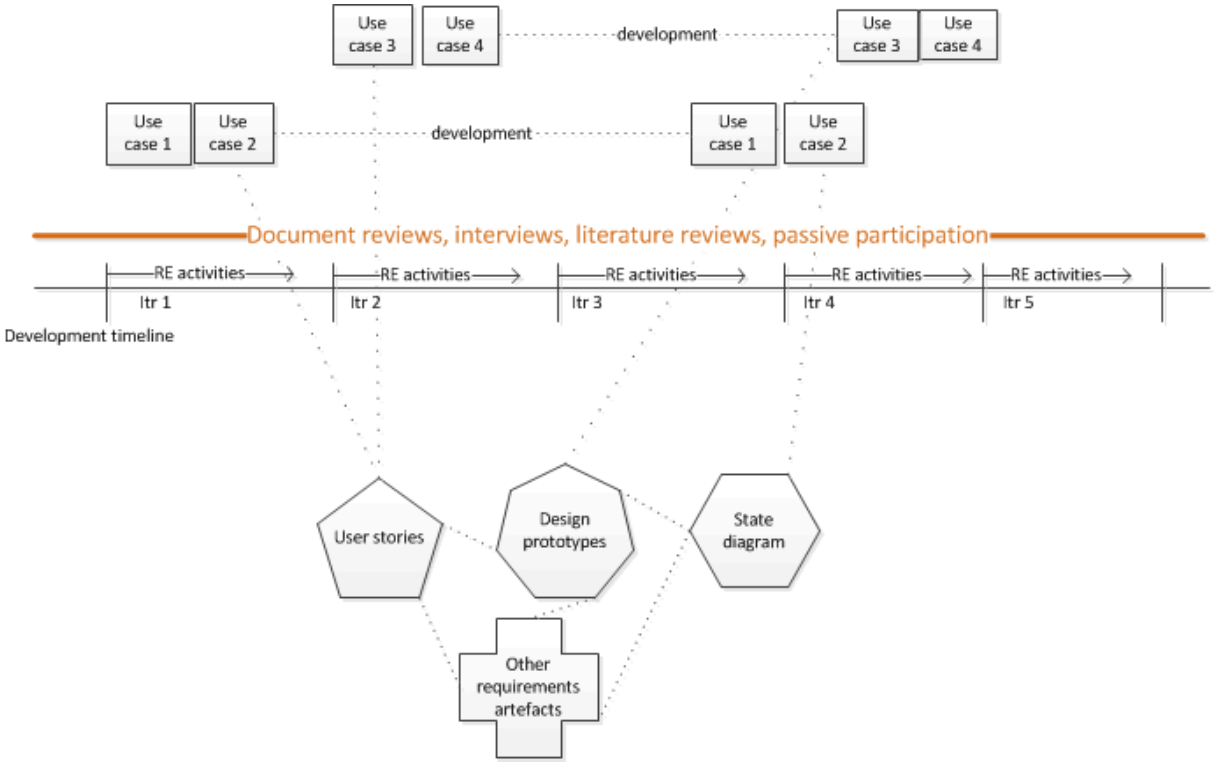

Figure 13 Illustration of study design

The research questions had to be adjusted and refined several times during data collection and data analysis described in the next section.

4.4 Data analysis

The main goal of data analyses is to derive conclusions from the data while keeping a clear chain of evidences, i.e. a reader should be able to follow the derivation of results and conclusions from the collected data (51) (49).

The analysis of case study evidences is one of the least developed and most difficult aspects of doing case studies (51).

As described in ((51) chapter 5), I attempted to structure and manipulate the collected data in the following ways:

- Putting information into different arrays
- Making a matrix of categories and placing the evidences within such categories
- Creating data displays—flowcharts and other graphics—for examining the data
- Tabulating the frequency of different events
- Putting information in chronological order or using some other temporal scheme

From the four general analysis strategies presented in ((51), chapter 5), the descriptive approach of the case study was the closest strategy that suited the circumstances of this study. Due to the lack of

clear and well-defined hypotheses or propositions in the problem domain of the study the strategy of *relying on theoretical propositions* could not be applied. However a comprehensive literature search and reviews of various studies related to the problem domain were applied to provide support to data analysis approaches and conclusions.

Four types of triangulation which refer to the use of multiple approaches in investigating the research problems are recommended to increase the validity of the data (49):

1- Usage of multiple methods in data collection (e.g. interviews, documents and observations)
2- Usage of multiple data sources (e.g. by documents reviews, interviews and observations)
3- Usage of more than one researcher or observer in the study
4- Using alternative theories or viewpoints

To follow the above mentioned recommendations, multiple methods of data collection and data sources were utilized in this study. However having another researcher was not possible. Through many discussions with my colleagues and members of the solution team, following blogs and discussion lists related to application of the requirements techniques (55) (56) (57), I hope some knowledge of different viewpoints was gained.

In the following sections, the analysis approaches of data obtained by interviews and document reviews are described (4.4.1.1 -4.4.1.2). Further, two analysis strategies to answer the research questions are outlined (4.4.1.3).

4.4.1 Analysis approaches

The analyses of data were performed iteratively through the course of the thesis and with data that were available at the time, mainly interview- and meeting notes and the current requirements artifacts. The analysis approach for the collected data through interviews and document reviews is described in the following.

4.4.1.1 Interviews

The interview- and meeting notes were reviewed in several rounds to understand the answers, extract and categorize the data with regard to the research questions. The interview questions were adjusted if necessary for the later interviews.

Answers to similar questions were set against the role of the interviewee in the project and the present phase of the project. Differences and similarities in the answers were noted.

In addition the interview notes were sorted chronologically in order to relate the data to the different phases of the project.

4.4.1.2 Document reviews

The project's documents were mainly reviewed to gain knowledge about the project status, on-going activities and future plans. Various presentations and reports were among the most studied documents in the study (Table 5).

In the reviews I tried to connect and categorize the documents to the activities they originated from or were related to. Further I investigated the purposes of the documents, to whom (e.g. which stakeholders) they might be directed and their content. In this way I could map the documents to different usage areas in my study.

The documents named in Table 5 are particularly used in the presentation of the project (Chapter three), RE process and techniques. The requirements artifacts that are studied and referred to (Table 4) were the basis for understanding RE activities and how the techniques were applied in different phases of the project.

4.4.1.3 Requirements artifacts analysis strategies

The following strategies were used to analyze the requirements artifacts and answer the research questions:

1- Analysis of the Requirements Engineering process
   The goal of this analysis was to identify the techniques applied in the different phases of the RE process and the purposes of their application. The results of this analysis were used to address the characterization and capabilities of the techniques and how they contributed to specify different types of functional requirements.

2- Investigation of use case evolution
   Evolution of use cases through the process of use case development was studied. The goal of this analysis (chapter five) was to understand how use cases were developed, which elements of the use cases were changed and what the changes were about. The results of the analysis were used to address the application of use cases and whether there was a relation between the changes, limitations and challenges of use cases and the usage of other requirements techniques.

4.5 Ethnography

Ethnography research method in software engineering aims at understanding cultural aspects, social behaviors, and communication strategies of the technical communities that collaborate to perform a task mainly through observations. Ethnography builds therefore local theories and cannot be used to prove hypothesis or theories (58).

Ethnography in empirical software engineering is beneficial due to the importance of impacts of social behaviors and interactions between the people involved in software practices (59). A major challenge in ethnography research is to perform a detailed observation, to collect data through observations and data analysis (58).

Ethnography is considered as the secondary research method in this study and was mainly practiced through passive participation in the meetings that were suggested by the solution team.

However doing this study on the CDR project conducted by my employer and colleagues that I knew has certainly had effects on my interpretations of collected data.

Being both an insider (knowing the organization and people involved in the project) and outsider (not a part of the project) have had both negative and positive effects. Ease of access to the projects material and members in addition to close distance to conduct interviews and ask questions were positive. Being familiar to the interest areas of people and which part of the project they worked on was also beneficial to know what to ask and who to contact to get the answers. However this knowledge may have influenced the questions I asked and how the questions were formulated. In addition my interpretations can be biased, being an insider researcher. To reduce this effect, I tried as possible as I could, to conduct the case study according to the guidelines and recommendations. In

addition I tried to provide data from different sources so that the conclusions were based on multiple sources. Further I tried to be fact-oriented in the study and analysis of data.

5 Analysis of use cases changes

Use cases were initiated by high-level descriptions of the functionalities and eventually elaborated and completed with details during the development activities. Use cases lifecycle will be discussed in detail in 6.2. Three versions of *UC Import Records from IRIS* under development are provided in 11.9.

In this chapter the results of the analyses of use cases changes through the first five development iterations are presented. The changes are investigated to understand how different use cases were developed and whether changes could be related to challenges or limitations of use cases.

In addition the changes analysis contributed to understand how the use cases technique was applied by different developers through the RE process.

Use cases from three functional areas (Data capture, Data processing and ETL) were followed in the period of January.2013 - February.2014. The three functional areas were selected to represent different types of requirements and stakeholders, particularly primary actors. Totally 12 use cases were selected and analyzed (the categorization of use cases is explained in detail in 6.3). The selection of use cases occurred under the progress of the project and during the first five iterations. Use cases were selected based on 1) the criticality of the functions (core functions of the CDR with higher priorities to be implemented) and 2) type of requirements they described and 3) their primary actors, described in 5.2 – 5.4.

An overview of the functional areas, selected use cases and their primary actors is provided in Figure 19, which shows the package diagram of the CDR use cases.

To follow use cases from the Data processing area, I regularly (almost every two weeks) saved copies of the use cases - called as snapshots - from January 2013 until August 2013. For the other functional areas, I used the versions of use cases[9] that were available on the project source code management tool (TFS, 3.10).

The number of snapshots for different use cases varied from two to eight; depending on how often use cases were changed and when the snapshots were taken. Therefore the snapshots cannot represent a complete history of use cases changes. The same applies for versions as the frequency of changes was dependent on how often developers chose to update the versions[10].

The rest of the chapter is structured as follows:

The approach of changes analysis is described in 5.1.

The results of the analyses for the studied use cases are presented in 5.2-5.4.

The results of the overall investigation of changes in all of the studied use cases are presented in 5.5.

[9] Some of the use cases (e.g. ETL use cases) were selected to be analyzed after they were developed. Therefore it was not possible to take snapshots of these use cases and I had to use the available versions.

[10] Number of snapshots and versions for each use case are specified later in the analysis (Chapter five).

5.1 Analysis approach

The approach to analyze the evolution of use cases was based on the comparison of a new version or new snapshot of a use case with its previous one. In this way the elements that were changed over time were identified. The changes in various elements were studied to understand what the changes were about and what they reflected. Further, the changes that were more critical, e.g. the changes in the main flow that reflected changes in the requirements (or scope of functions) were identified. These types of changes were considered as more critical than modifications in Issues/note elements.

Moreover, the number of changes of the same type was counted within each use case element. This was done to identify what the majority of changes were about and if any pattern of changes could be recognized.

Finally, a summary of the changes for each use case element was provided in each functional area (5.2 -5.4).The total overviews of changes can be found in 11.4.

To illustrate how the analysis was conducted, comparisons of two snapshots of the use case *Export records to IRIS* taken on 14.02.2013 and 28.02.2013 and results of the comparison are provided below.

ExamDiff Pro (60) was used to compare versions or snapshots of use cases. The colors indicate types of changes: Blue (deleted), Dark red (Added), Yellow (Changed) and Pink (Changed in changed).

The left panes on the figures are from 14.02.2013 while the right panes are from 28.02.2013 in all of the figures below.

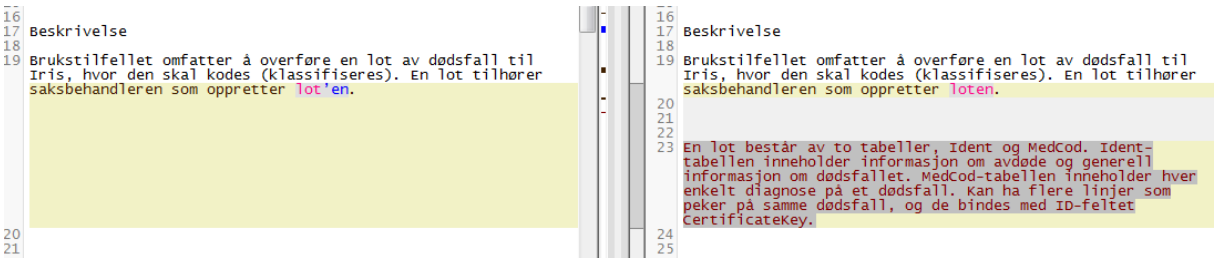

Figure 14 Use case Export Records to IRIS, changes in Description element

Two changes in the Description field are identified in Figure 14:

1- Syntax change (lot'en changed to loten), categorized as minor changes and are not included in the analysis.
2- Description of *lot* (in dark red on the right pane): this type of changes is categorized as additional description or specification of technical terms or components of the CDR or external systems such as IRIS in this case.

Figure 15 Use case Export Records to IRIS, changes in Description element (GUI)

One change (deletion of the text in blue) is identified in the comparison shown in Figure 15. The GUI related specification of *Status* field was removed. This type of changes is categorized as modifications of GUI specifications. There are no further descriptions of GUI changes as the GUI specification was removed from the use cases in the final stage (6.2.3)

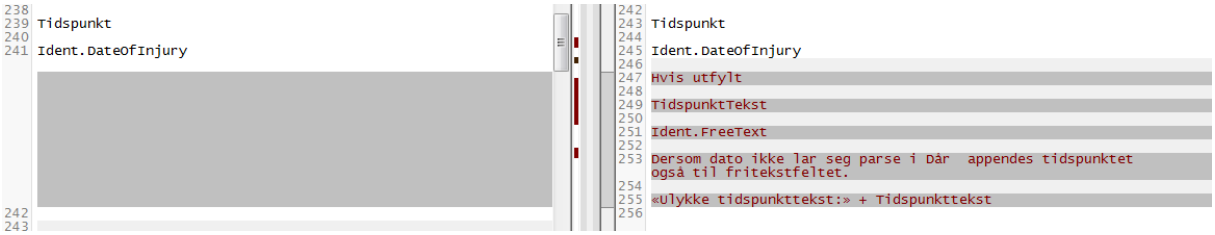

Figure 16 Use case Export Records to IRIS, changes in Description element (IRIS integration)

Two changes (addition of the text in red) are identified in the comparison shown in Figure 16.

1- The specification of translation rule for *Tidspunkt* field was added (by *Hvis utfylt*).
2- A new field, *TidspunktTekst,* in the CDR, its respective field in IRIS *Ident.FreeText* and the technical solution to handle the date field if it could not be parsed in the CDR was added.

These changes are categorized as 1) modification in specification of translation rules and 2) modification in the specification of IRIS integration.

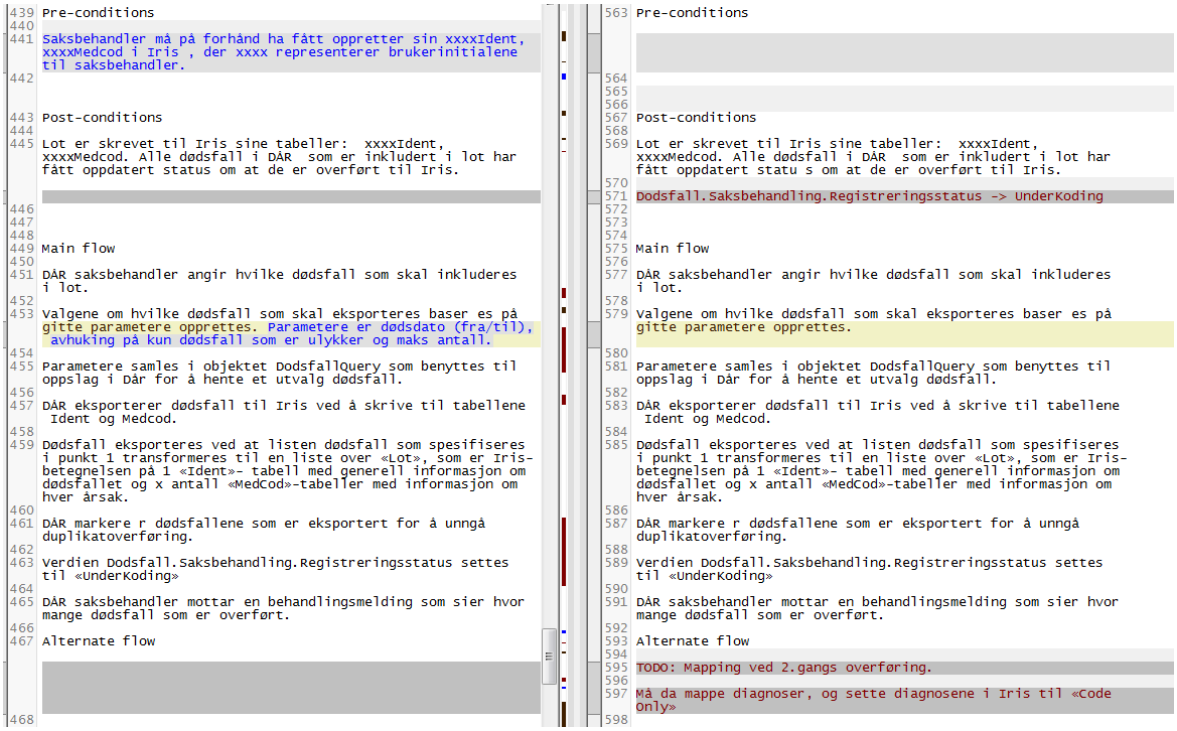

Figure 17 Use case Export Records to IRIS, changes in Pre-conditions, Post conditions, Main flow and Alternate flow

The following changes are identified in the comparison shown in Figure 17:

1. Deletion of Pre-conditions
2. Addition of update rule for the field *Dodsfall.Saksbehandling.Registreringsstatus* to *UnderKoding* in Post-conditions.
3. Deletion of the specification of input parameters *(dødsdato (fra/til), avhuking på kun dødsfall som er ulykker og maks antall* in the Main flow.
4. Addition of to-do and technical notes in Alternate flow

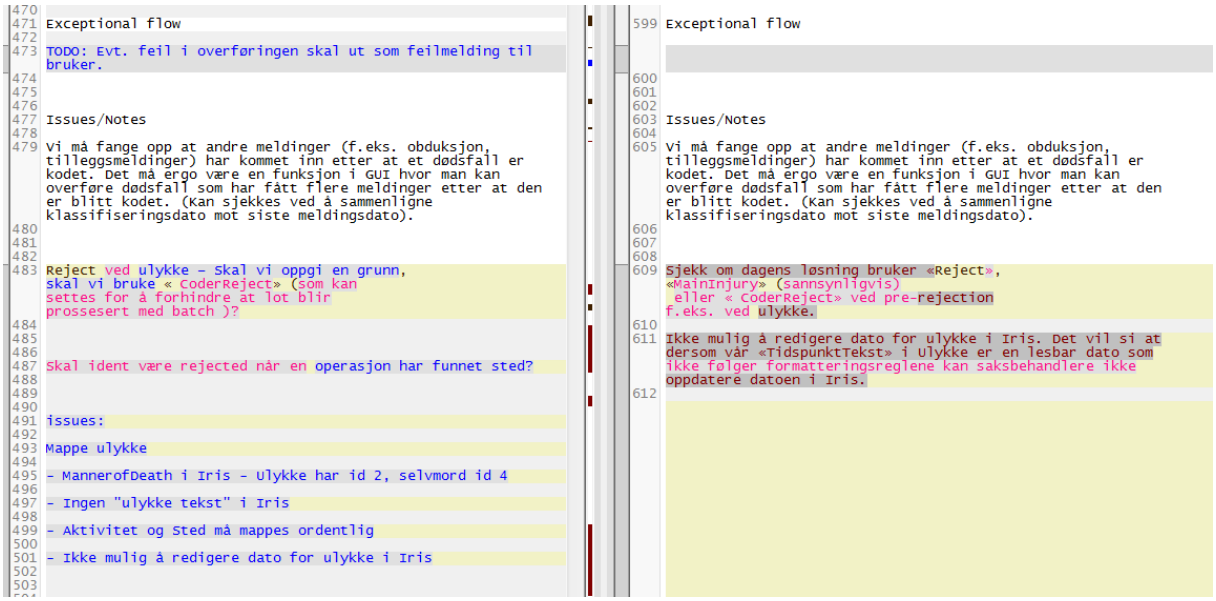

Figure 18 Use case Export Records to IRIS, changes in Exceptional flow and Issues/notes

The following changes are identified in the comparison shown in Figure 18:

1- Deletion in Exceptional flow the to-do is removed
2- Modifications (both addition and deletion) in the Issues/Notes

The results of the above mentioned comparisons are documented as follows (Table 7):

Versions comparisons	UC element	Changes	Comments
V2-V3 (14.02.2013-(28.02.2013)	Description	• Additional description of the IRIS component • GUI specification modified • changes in the specification of translation rules √ • New row specification of IRIS integration√(Figure 14, Figure 15, Figure 16)	
	Pre-conditions	Removed (Figure 17)	
	Post-conditions	The rule for update of Coding status added (technical note) (Figure 17) √	
	Main flow	Information of input parameters removed (Figure 17)	
	Alternate flow	To-do notes added (Figure 17)	
	Exceptional flow	To-do notes deleted (Figure 18)	
	Issues / notes	Modifications (new notes added, some of the old removed (Figure 18))	

Table 7 Analysis approach, example of comparisons of two versions of UC Export records to IRIS

The changes that are considered as critical in that they reflect changes in the functionality or scope of the functions are marked with √ in the Changes column. Question mark is used when the interpretation of changes was uncertain.

The following sections (5.2-5.4) describe the functional areas and the use cases that are analyzed. Further the overall investigation of the results of the analyses is presented in 5.5.

In Figure 19, functional areas, selected use cases and their primary actors are represented.

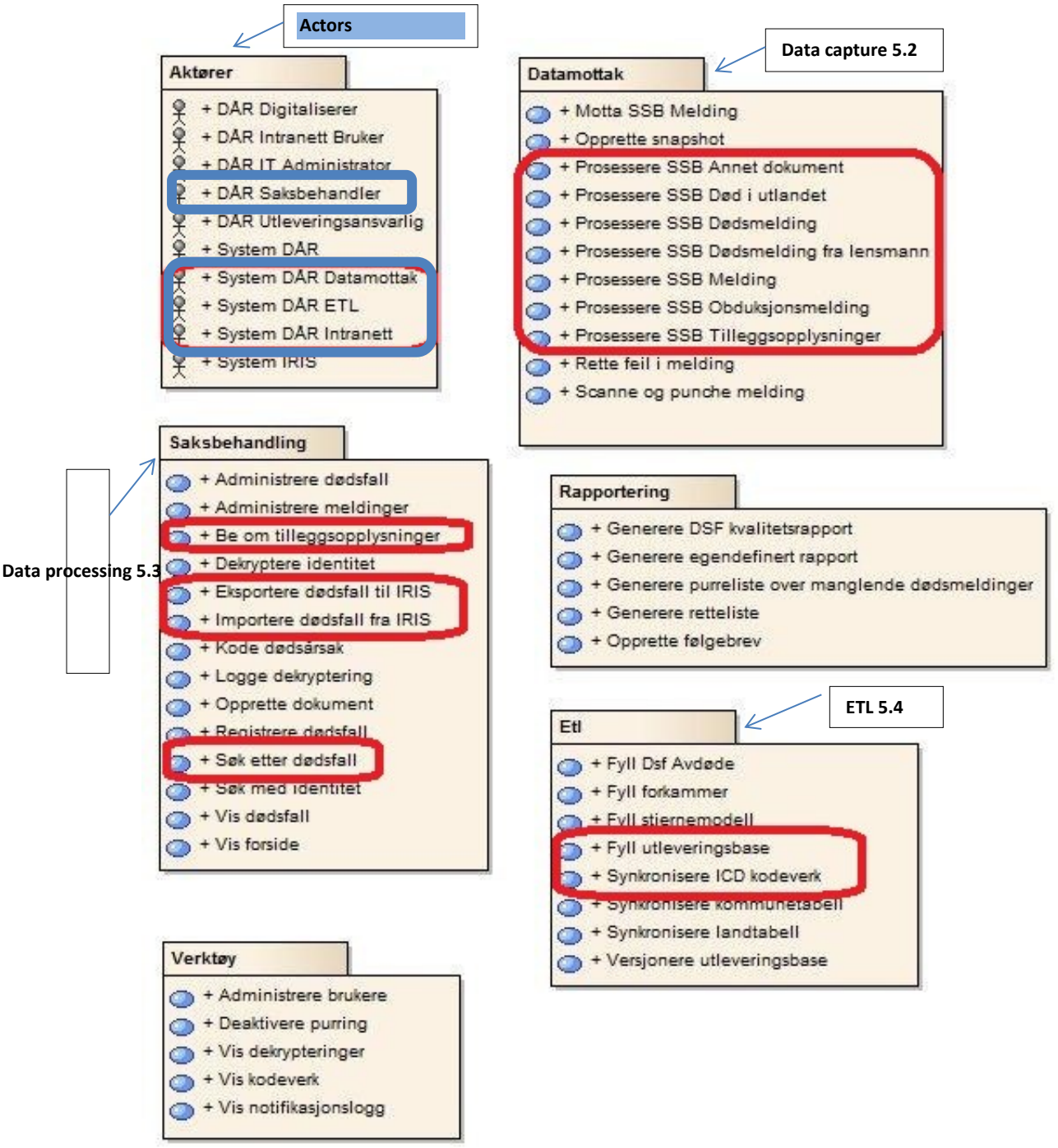

Figure 19 The CDR Package diagram, selected use cases in each functional area are marked in the red outlines. In addition the primary actors involved in use cases are outlined.

5.2 Data capture

The main task of the Data capture was to receive and process different types of cause of death data, referred to as "messages".

There are six types of messages processed by the Data capture unit; 1) death certificate, 2) police report, 3) autopsy report, 4) additional information, 5) death abroad and 6) others.

The file format and data fields were the same for all of the types, though they differed on mandatory data fields.

The file format and fields were not changed in the new solution.

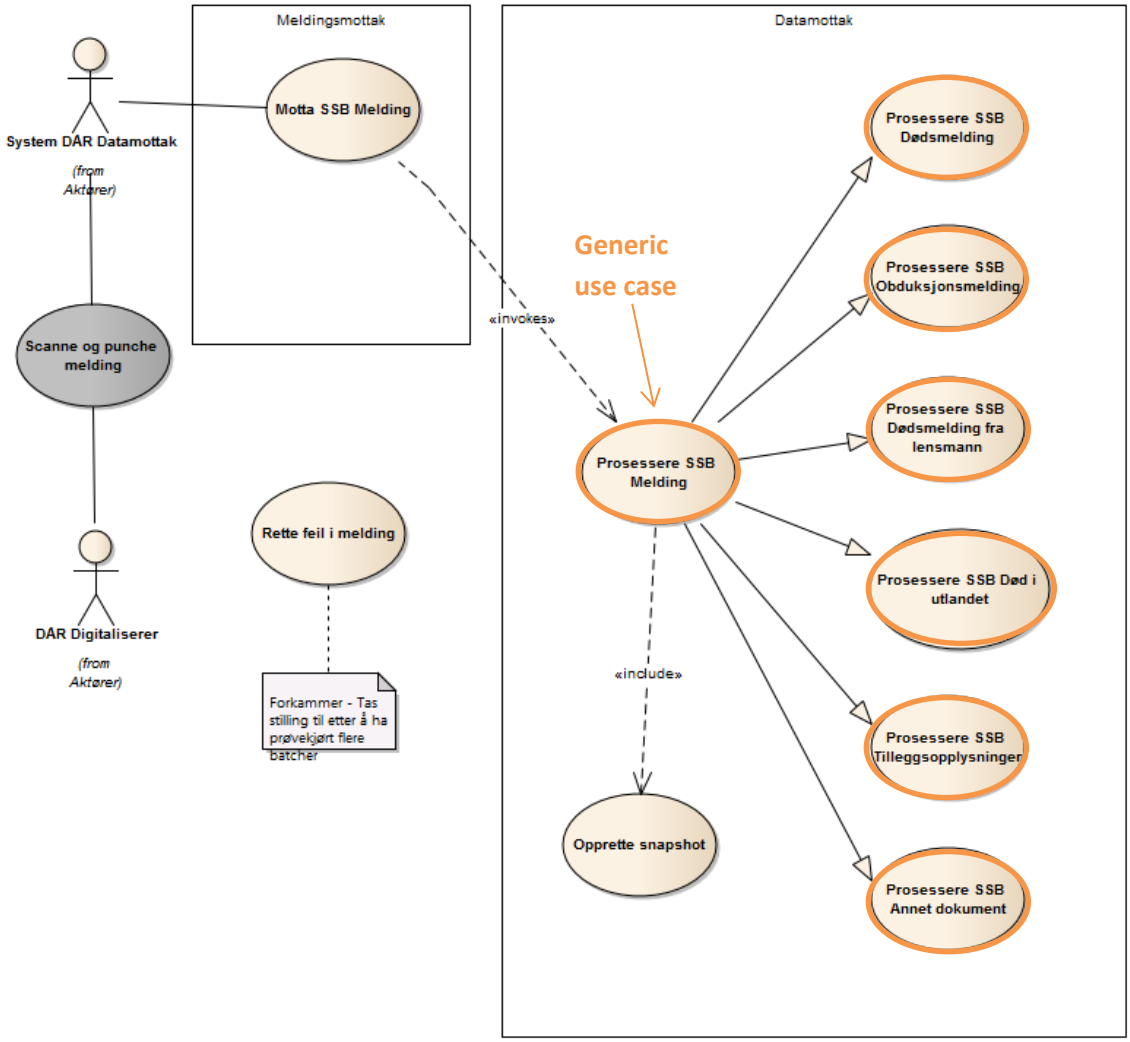

Figure 20 Data capture use case diagram, studied use cases are marked with orange circles.

5.2.1 Use cases overview

Six use cases (Table 8) for handling and processing the six types of messages were specified. These use cases were special cases of the generic use case *UC Process SSB Message*[11] which described the

[11] Since all of these messages are received from Static Norway (Statistisk Sentral Byrå), the name of the use cases includes SSB.

common part of the use cases such as message validation rules, handling duplicate death records[12] , general rules for mapping of fields (from the fields in the messages to the CDR fields) and exception scenarios in all of the use cases. The use case diagram for Data capture functional area is shown in Figure 20. The studied use cases are listed in Table 8.

Functional area: Data capture Primary actor: System CDR Data capture
UC Process SSB Message (Generic use case)
1. UC Process SSB Death Message
2. UC Process SSB Autopsy Message
3. UC Process SSB Police Message
4. UC Process SSB Death Abroad Message
5. UC Process SSB Additional Info Message
6. UC Process SSB Other Message

Table 8 Data capture use cases

5.2.2 Use cases changes

Four versions were available for each of the six use cases[13]. The versions of use cases in the chronological order were compared to each other to identify the elements that were changed in the use cases.

The results of comparisons are summarized in Table 9. See 11.4.5 for the total overview over changes.

Use case element	**Changes (number of changes)**	**Comments**
Trigger	Was not changed	
Alternate flow	• Removed due to changes in the Main flow and Post-conditions in handling duplicate messages related to the same death record(5) √ • Substituted with reference to another use case (2)	
Description	• Removed totally (2) • Substitutions with reference to the generic use case (UC Process SSB melding) in the content (4)	
Exceptional flow	Removed (Substituted with the reference to *UC Process SSB melding*)(6)	Was specified in the generic use case
Main flow	• Substitutions with references to other use cases (2) • Addition of reference to exceptional flow (1) • Update rules for existing records when new messages are received (5) √ • Registration in the notification log if the record has under coding status (5) √ • All the steps in the main flow added (1)	
Pre-conditions	Was not specified	

[12] A death record refers to a death that is registered in the CDR. The registration occurs when a message (of any of the six types) is received.

[13] The generic use case (UC Process SSB Message) is not included in the summary as only two versions of this use case were available and the changes were not critical.

Use case element	Changes (number of changes)	Comments
Post-condition	• Modified due to changes in the requirements of the output (5)√	Modified (two possible post-conditions were either a record created or a record updated with a new message)
Issues/notes	Modified in relation to the changes in the use cases	

Table 9 Changes analysis in use cases of Data capture (six use cases)

"Changes in the use cases were made ad-hoc when necessary. Situations that led to use case updates were the following:

- *Issues that had been clarified, decided and resolved during the development*
- *Missing information that was discovered during the development*
- *Errors that arose during the development"*

(From interview with the involved developer)

5.3 Data processing

The use cases of this area described the requirements for the core functionalities of the CDR which are briefly outlined in the next section. The term of Data processing (Saksbehandling in Norwegian) refers to the tasks for administration, handling and processing the records in the registry.

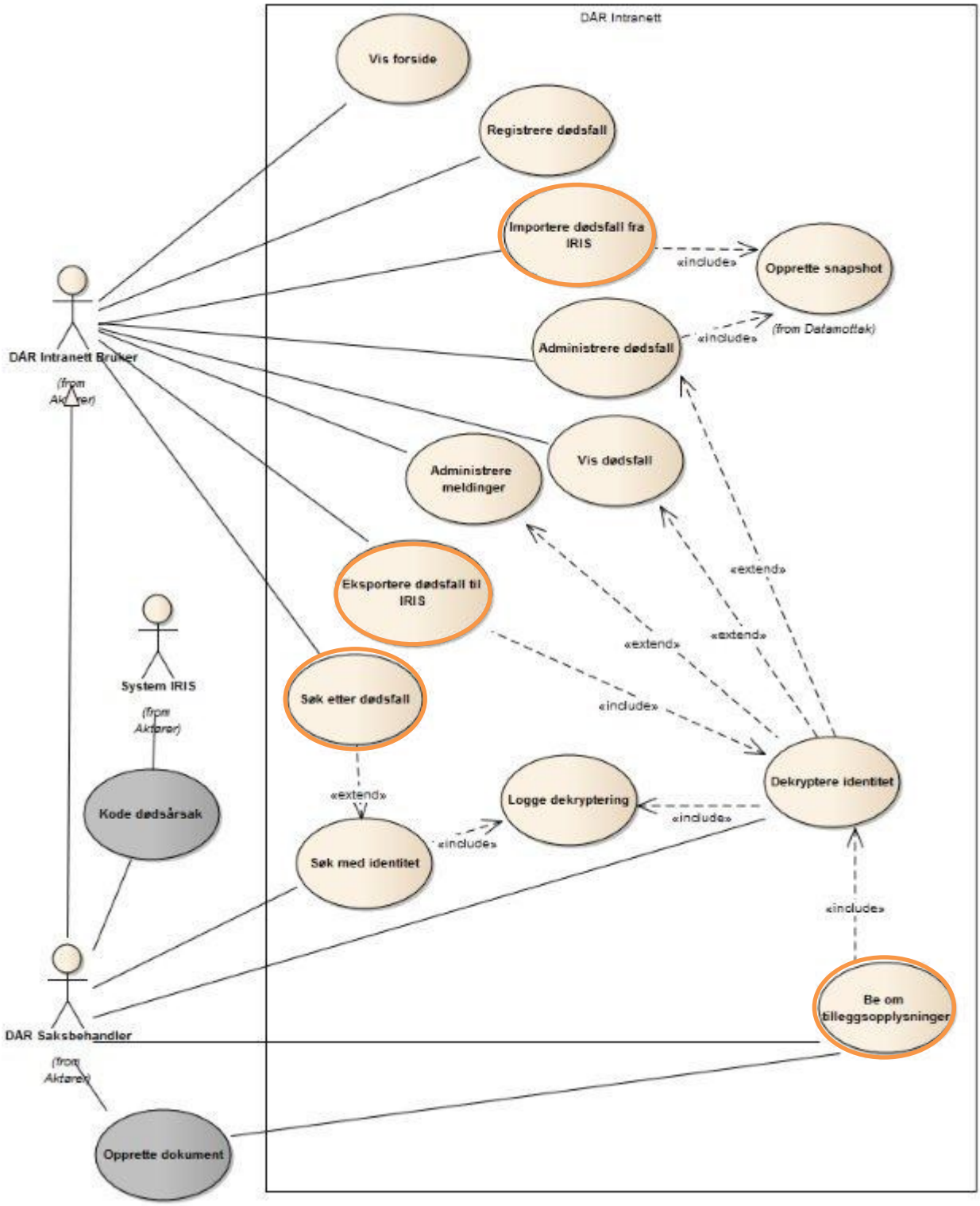

Figure 21 Data processing use case diagram, studied use cases are outlined with orange circles.

5.3.1 Use cases overview and changes

Four use cases from Data processing functional area (Table 10) were followed and analyzed. These four use cases were selected since they represented two categories of the studied use cases discussed in 6.3. In addition the level of complexity and detail in these use cases were different. Further these use cases represented the core functions and requirements of the system.

The use case diagram for Data processing is shown in Figure 21. The studied use cases are listed in Table 10.

Functional area: Data processing Primary actor: CDR Intranet user
1. UC Export Records To IRIS (5.3.1.1)
2. UC Import Records From IRIS (5.3.1.2)
3. UC Search For Records (5.3.1.3)
4. UC Request For Additional Documents (5.3.1.4)

Table 10 Data processing use cases

5.3.1.1 Export records to IRIS (UC Export Records To IRIS)

The most central task of the CDR is to classify cause(s) of death based on the available information such as medical history, personal info or environmental information provided by different sources. Cause(s) of death are classified by IRIS (61), which has been utilized for automatic coding (classification) since 2011 in the CDR (from internal documents, presented in Table 5).

IRIS is an interactive system for coding of multiple causes of death and for selecting the underlying cause of death. IRIS is based on the international death certificate form provided by WHO' (44). Causes of death are coded according to the ICD 10[14] rules and guidelines in IRIS which is used by as many as 20 countries around the world. IRIS is highly preferred in Norway among researchers because of the international code standardization it provides (61).

Nearly 50% of deaths are coded automatically in IRIS, while the rest are coded manually in the CDR (from interview with the system architect and IRIS presentation)

UC Export Records To IRIS specified the export function and how a selection of death records is exported from the CDR to IRIS.

Eight snapshots of this use case were analyzed and compared. The summary of changes is provided in Table 11.

Use case element	**Changes (number of changes)**	**Comments**
Trigger	Was not changed	
Alternate flow	• To-do notes added (1) • To-do about the second time processing of a record removed (1) • Five criteria for alternative flow added (1) √(?)	Alternate flows criteria were specified but there was no connection between the criteria and Main flow (in which step the criteria were checked?)

[14] ICD: International Classification of Diseases (ICD) (40)

Use case element	Changes (number of changes)	Comments
Description	• Addition of new specifications in IRIS integration table (7)[15] √ • Modifications of specifications in IRIS integration table (9) √ • Additional description of IRIS components (1) • Modification of GUI specification (5) • Addition of technical solution for the second time export of a record (1) • Addition of new conditions for exceptional flow (1) √	Description element was used to - specify conditions of alternate flows (pre-rejection rules for records in IRIS) - GUI specification - IRIS integration - Alternate flow (second time export of a record)
Exceptional flow	Removed (to-do notes deleted) (1)	No specification in the last version
Main flow	• Input parameter added (1) • Input parameter deleted (1) • Technical description of *Dodsfallquery* added(1) • Description of IRIS component (lot) added (1) • Description of IRIS component (lot) removed (1) • Rule for update of coding status added (1) √ • Pre-condition was included in step 4 (1) √	Five changes of seven reflected changes in additional description of components and parameters. Only two changes referred to changes in functionality or requirements.
Post-conditions	Update rule for coding status added (1) Update rule for Coding status removed (1)	The reason for deletion can be related to addition of the rule in the main flow
Pre-conditions	Removed (included in the main flow)	Not specified in the last version
Issues/notes	• How to capture new messages related to existing records after a record have been processed (coded) in IRIS. • Need for a function that can export records that are already coded but have received new messages was registered. • New issues on IRIS integration added, questions about different possible solutions added	Due to diversity of changes, the number of changes is not counted. The element was at least 5 times changed.

Table 11 Changes analysis in use case Export death records to IRIS use

For the total overview over changes see 11.4.1.

[15] In each addition several fields were added.

5.3.1.2 Import records from IRIS to CDR (UC Import Records from IRIS)

This use case specified the import function of a selection of death records from IRIS to the CDR. After completed import, the records and their status were updated in the CDR. The records that were imported back, were deleted from IRIS.

Six snapshots of this use case were analyzed and compared. The summary of changes is provided in Table 12.

Use case element	Changes (number of changes)	Comments
Trigger	Was not changed	
Alternate flow	Was not specified	Specified in Description
Description	• Headline of the table of the specification of IRIS integration added (1) • Modification of GUI specifications (4) • Modification of specification of IRIS integration table (14) √ • Cases for records that cannot be imported back added (1) • Description of lot added (1)	Description field was used for specification of alternate flow (the records that could not be imported back)
Exceptional flow	Was not specified	
Main flow	• High-level description of main scenario added (in one sentence) (1) • Steps of the main flow (2, 3a-3e and 4) added (1) √ • Status Final was removed from the selection criteria in IRIS for records to be imported (1) √ • Coding statuses were changed to initial, rejected and final by import if the cause of death was defined (1) √	
Post-condition	Added (1) √	
Pre-conditions	Added (1) √	
Issues/notes	• Issues and questions about the coding statuses, follow-up queues and IRIS integration added • Removed	

Table 12 Changes analysis in use case Import death records from IRIS

For the total overview over changes see 11.4.2.

5.3.1.3 Search records (UC Search for Records)

This use case specified the technical requirements of the search function. The search function was primarily developed to select the records for export to IRIS. The function was further expanded to provide advanced searches to support administrative and quality control tasks.

Requirements for the search criteria and search results were specified by the interaction designer and through user stories and design prototypes.

Three versions of this use case were analyzed and compared. The summary of changes for each use case element is provided in Table 13. For the total overview over changes see 11.4.3.

The use case was not often changed as the major part of the changes was conducted on the design prototypes.

Use case element	Changes (number of changes)	Comments
Trigger	Not changed	
Alternate flow	Was not specified	
Description	• Description of the initial purpose of the search function in relation to export and import functions added (1) • Division of simple and advanced search described (1) • Limitation of the number of the results discussed (1) • Possibility to export to IRIS for search results described (1) • Sort on all of the columns described (1) • Possible criteria and precedence of *Ident* over other criteria described (1) • Reference to extended use case added (1)√ • Requirement of audit logging by decryption of sensitive info removed (1)√	Requirements of audit logging were specified in the extended use case.
Exceptional flow	Was not specified	
Main flow	Not changed	
Post-condition	Removed (1)√	Due to the usage of extended use case which specified the requirements of audit logging
Pre-conditions	Was not specified	
Issues/notes	References to GUI Interaction design, domain model and implementation added (1)	

Table 13 Changes analysis in use case Search death records

5.3.1.4 Request for additional documents (UC Request for Additional Documents)

This use case specified the feature that provided relevant personal- and death information of records to be used in the letter of request for additional documentation sent by the CDR to stakeholders such as Cancer registry and Birth register. Creation of the request letter and addressing was though a manual process and not a part of this feature.

Four snapshots of this use case were analyzed and compared. The summary of changes is provided in Table 14. For the total overview over changes see 11.4.4.

Use case element	Changes (number of changes)	Nr of changes in four versions
Trigger	Was not changed	
Alternate flow	Was not specified	
Description	• Complementary info about the process of letter generation added and removed (1) • Specification of the fields to fill out by requesting additional info added and removed (1) • Template (reference to predefined templates that are found) added and removed (1) • Business rules added and removed (1) • Description of generation of the additional document added and removed (1)	The changes reflected changes in the scope of the function.
Exceptional flow	Was not specified	
Main flow	• Steps (3) of the flow added (1) • 2 steps modified (1) √ • 2 steps removed according to the changes in the function (1) √	Changes were related to the changes in the scope of the function
Post-condition	Added and modified (2) √	
Pre-conditions	Added (1)	
Issues/notes	Questions, solution suggestions, information added and removed	

Table 14 Changes analysis in use case Request for additional doc.

5.4 ETL

ETL refers to Extract, Transform, and Load and is usually utilized in data warehouse solutions to load and process large amounts of data between various sources (62). The use case diagram for ETL use cases is shown in Figure 22.

Due to the information-centric character of data warehouse solutions, specification of information requirements -which concerned 1) the data that must be accessible in a data warehouse, 2) where the data come from (which databases), 3) how they are transformed and organized, as well as 4) how they should be represented e.g. aggregated or calculated- was an essential part of the development of the data warehouse use cases.

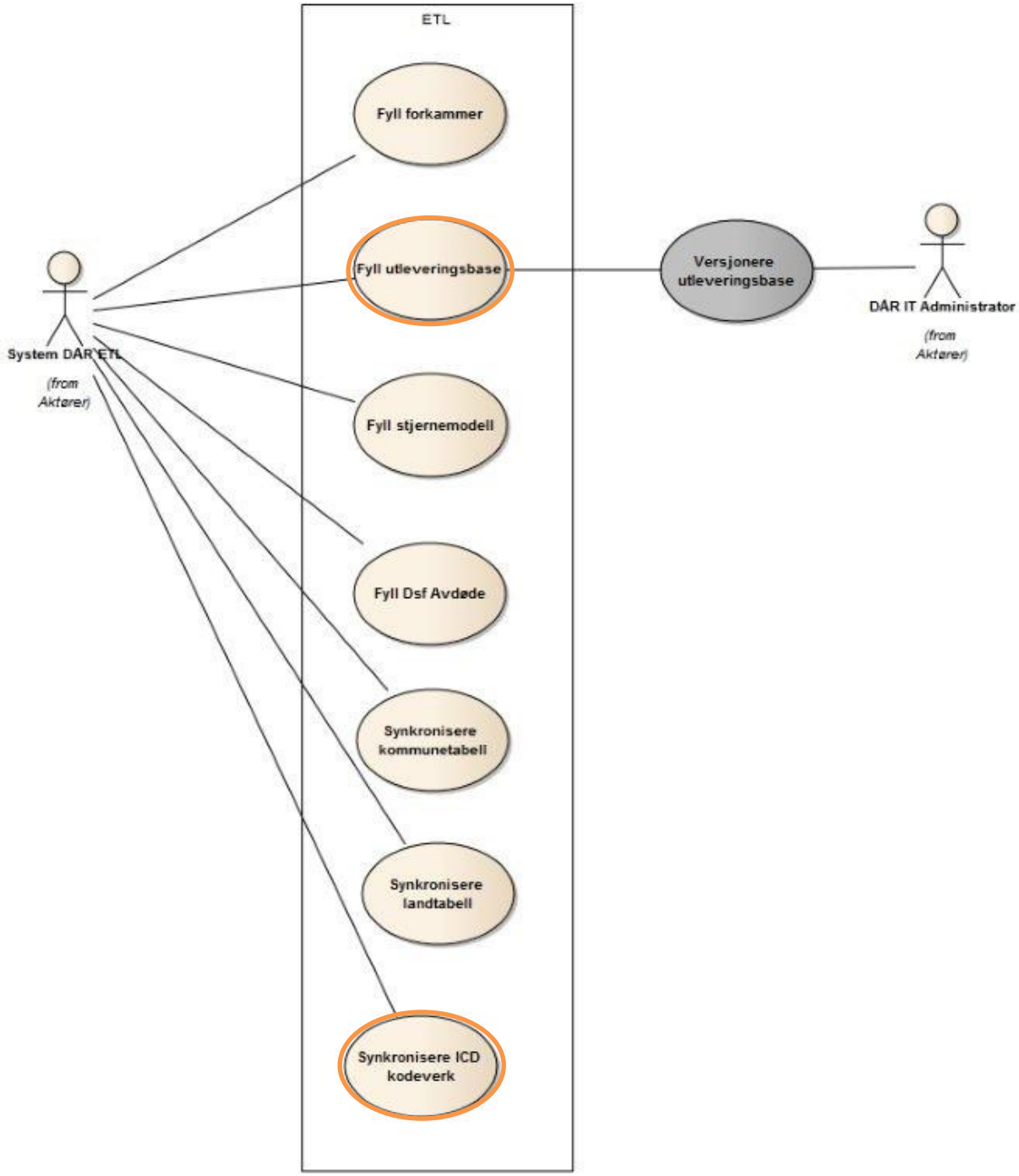

Figure 22 ETL use case diagram, studied use cases are outlined with orange circles.

5.4.1 Use cases overview

Two use cases for data warehouse solution and data synchronization from ETL functional area are studied. The studied use cases are listed in Table 15.

Functional area: ETL Primary actor: System CDR ETL
1. UC Fill Delivery Data base (5.4.1.1)
2. UC Synchronize ICD Codes (5.4.1.2)

Table 15 ETL use cases

Description of the use cases and changes analyses are provided in 5.4.1.1-5.4.1.2.

5.4.1.1 Data warehouse

UC Fill Delivery Database described the ETL service that processed and loaded a specific set of data including - personal data from the CDR database and health data from the data warehouse - to the database used for delivering data to external users and organizations (delivery database).

Five versions of this use case were analyzed and compared. The summary of changes for each use case element is provided in Table 16. For the total overview over changes see 11.4.6.2.

UC element	Changes (number of changes)	Comments
Trigger	Added (as a job running once a week) (1)	The change reflect the decision of the trigger
Description	• Databases names précised (1) • Additional description of batch transfer, freeze, robust transfer, job table, status of jobs, parameters, solution package added (1)	Description was used to describe various aspects of the technical solution.
Pre-condition	• Name of databases précised (1) • Additional description about the synchronization of databases in the data warehouse solution added (1)	
Issues /notes	• Technical issues such as network problems, efficiency, Oracle drivers, memory and efficiency added (1) • The resolved issues on differences between test and production environment removed (1)	
Main flow	• All the original steps removed (1) • The steps (7) added (1) √	Changes could be related to the refinements of the main flow to match the actual implementation of the function.
Post-condition	Was modified with reference to the domain model (1)	
Alternate flow	Was not specified	
Exceptional flow	Was not specified	

Table 16 Changes analysis in use case Fill Delivery Database

5.4.1.2 Data synchronization

UC Synchronize ICD Codes described how the existing diagnosis codes in the CDR database were synchronized with the ICD codes (40). The ICD codes in form of CSV[16] files were uploaded from WHO[17] or were maintained locally dependent of the version of codes. The synchronization was triggered when the intermediate database (forkammeret) was uploaded with the reference data (ICD codes and their related data).

Two versions of this use case were analyzed and compared. Due to few versions of the use case, interpretation of changes was difficult. Therefore changes that could be related to changes in

[16] Comma-Separated Values (CSV)
[17] WHO: World Health Organization (44)

requirements or to the implementation are denoted with question marks. The summary of changes is provided in Table 17. For the total overview over changes see 11.4.6.1.

UC element	Changes (number of changes)	Comments
Trigger	Was not changed	
Invariant	• Description of target tables addedV (?) • Description of source files added • Handling different ICD codes versions addedV (?)	
Description	• Additional description on download of ICD codes added (1) • Description of the comparison of reference data in the CDR with the ICD codes added (1)	
Main flow	All the previous steps removed and 13 new steps added (1) V (?)	
Exceptional flow	Removed	Not specified in the last version
Post-conditions	The tables that will be updated were specified. (1) V (?)	
Pre-conditions	Was not changed	
Alternate flow	Was not specified	
Issues/notes	Was not changed	

Table 17 Changes analysis in use case Synchronize ICD Codes

5.5 The overall investigation of changes

The results of overall investigation of changes in the studied use cases revealed the following results:

- Trigger was not changed in 11 out of 12 use cases.
- Exceptional flow and Alternate flow were either not specified or removed during the development in 12 of 12 use cases.
- The use cases elements that were changed by changes in the requirements, scope of the functions and implementation (during the development of the use cases) are listed in Table 18. Main flow and Post-conditions mainly reflected these types of changes.

Use case	Use case element	Comment
Data capture use cases (Table 8)	Alternate flow	Changes in the requirements
	Main flow	Changes in the requirements
	Post-conditions	Changes in the requirements
UC Export Records to IRIS	Alternate flow	Changes in implementation
	Description	Changes in implementation
	Main flow	Changes in implementation
UC Import Records from IRIS	Description	Changes in implementation
	Main flow	Changes in implementation
	Post-condition	Changes in implementation
	Pre-condition	Changes in implementation
UC Search for Records	Post-condition	Changes in implementation
UC Req. for additional documents	Main flow	Changes in the scope of the function
	Post-condition	Changes in the scope of the function
UC Fill Delivery Database	Trigger	Changes in implementation?
	Main flow	Changes in implementation?
UC Synchronize ICD Codes	Invariant	Changes in the implementation?
	Main flow	Changes in implementation?
	Post-conditions	Changes in implementation?

Table 18 Use case elements with critical changes

6 Use cases challenges and limitations

Use cases were considered as the main artifacts to represent functional requirements in the CDR project. Hence investigating the application of the use cases technique in different phases of the RE process and for different types of requirements were essential to understand the capabilities and limitations of the technique and to answer the first research question of the study (1.2).

This chapter is structured as follows:

1- Description of the RE process in the CDR development (6.1)
   - Description of the RE activities (6.1.1)
2- Use cases lifecycle through RE process (6.2)
   - Different forms of use cases (6.2.1-6.2.3)
3- Use cases categories (6.3)
   - Analysis of use cases development in each category
4- Summary of results (6.4)

6.1 The CDR Requirements Engineering process

The Requirements Engineering process (RE) in the development of the CDR was iterative and intertwined with the development activities.

The RE process is illustrated by the model in Figure 23. The model is developed based on:

- Knowledge gained by following project activities and my interpretations of the activities
- Knowledge of RE process and activities through literature
- Interviews with the solution team
- Studying the requirements artifacts (Table 4)
- Review of presentations and documents related to the requirements artifacts (Table 5)

The four activities of the process were iterative; shown by the round arrow in the activity boxes. The three activities which were conducted in the development iterations are placed in the dashed rectangle.

The arrows between the activities in Figure 23 show the order and iteration between the activities and how they can trigger and affect each other by their outcomes. For instance the need of extra information for development of the requirements triggers further analyses. Missing requirements and errors found in the reviews and quality assurance trigger changes in the requirements and consequently affect the development of the requirements.

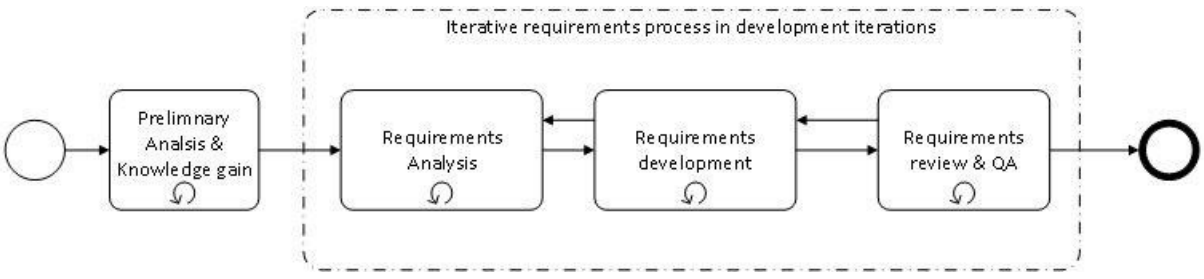

Figure 23 Requirements engineering process in the CDR development

The definitions of the RE activities in the context of this study are provided in the following section (6.1.1). However there are references to literature to address the variations in definitions and to contrast differences. In addition a comprehensive description of the RE process in the CDR development is provided in (11.5) where the activities, inputs, outputs, techniques used to conduct the activities and the roles involved are described in detail.

6.1.1 Activities

6.1.1.1 Preliminary analysis and knowledge gain

This activity refers to the high-level analyses and investigations of the business domain - the CDR - and understanding and identifying the needs of the solution. The main outcomes of this activity were the following:

- The scope of the solution: refers to the definition of the boundaries of the solution against the external systems or stakeholders that interact with the CDR (see Figure 10, the context diagram). In addition the scope of the solution defines the functionalities that should be implemented so that the solution has the necessary capabilities and can satisfy the needs of the stakeholders.

- Identification of stakeholders: refer to individuals or groups or systems that are involved in the solution or whose interests may be affected as a result of the project execution or project completion (15). The identification of stakeholders, their roles and responsibilities was done during the analysis phase.

- Priorities: were mainly decided based on the criticality of the functions, i.e. the functions that were necessary to perform the core tasks of the CDR were prioritized. For instance receiving and processing death messages and classifying causes of death were among the highly prioritized functions.

The constraints, cost estimations and risks were already clarified in the pre-project (ref. pre-project report, Table 5)

The literature includes the definition of the terminology of the business domain in this activity (17). In addition the identification of conflicts between the requirements of different stakeholders is mentioned as a task of the analysis phase (15).

6.1.1.2 Requirements analysis

This activity refers to analysis of those requirements that were going to be implemented in a development-iteration. The goal of the analysis was to understand and characterize the requirements by gathering and investigating the available data and artifacts obtained from the preliminary analysis and during the implementation of the requirements. The results of the analysis provided the necessary building blocks for development of the requirements artifacts such as use cases or modeling diagrams. In addition, exploring different possible solutions and identifying issues concerning alternative solutions were conducted.

6.1.1.3 Requirements development (elaborating)

This activity refers to the development (elaborating) of the requirements artifacts based on the knowledge gained through the analysis (described in 6.1.1.1) and the details captured under the development of the solution. The requirements were elaborated under design, implementation and

test activities in the development iterations. These activities had direct impact on the evolution of the requirements. Changes in the design of the solution led to changes in the requirements specification. Further, during the implementation, the requirements were detailed by new findings and discoveries. Results of tests in the form of errors or feedbacks from users led also to changes in the specification of the requirements.

The development (elaborating) of the requirements lasted until the implementations of the features were finished (according to the agreed scope) and approved by the users.

Literature defines requirements development (elaborating) as a collection of activities, tasks, techniques and tools to identify, analyze and validate requirements. It includes the process of transforming needs into requirements. The purpose of requirements development is to elicit, analyze, establish and validate customer, product, and product component requirements (15).

6.1.1.4 Requirements review and quality assurance

Requirements artifacts such as use cases were reviewed and quality assured when the implementation of use cases was completed. The goal of the reviews was to ensure the correctness and completeness of the specification of the requirements and that they reflected the actual implementation of the solution.

Various manual and automatic tests were conducted by different members of the solution team during these reviews (3.9).

6.2 Use cases lifecycle

The lifecycle of use cases is divided into three stages in the CDR project: Initial, Under development and Final. These stages aligned with the RE process (6.1) and activities of the development iterations are depicted in Figure 24.

The different forms of use cases through their lifecycle are discussed in 6.2.1 – 6.2.3. In section 6.3 the development of the studied use cases is categorized into three groups – with respect to the types of the requirements - and limitations or challenges of use cases in each group are addressed.

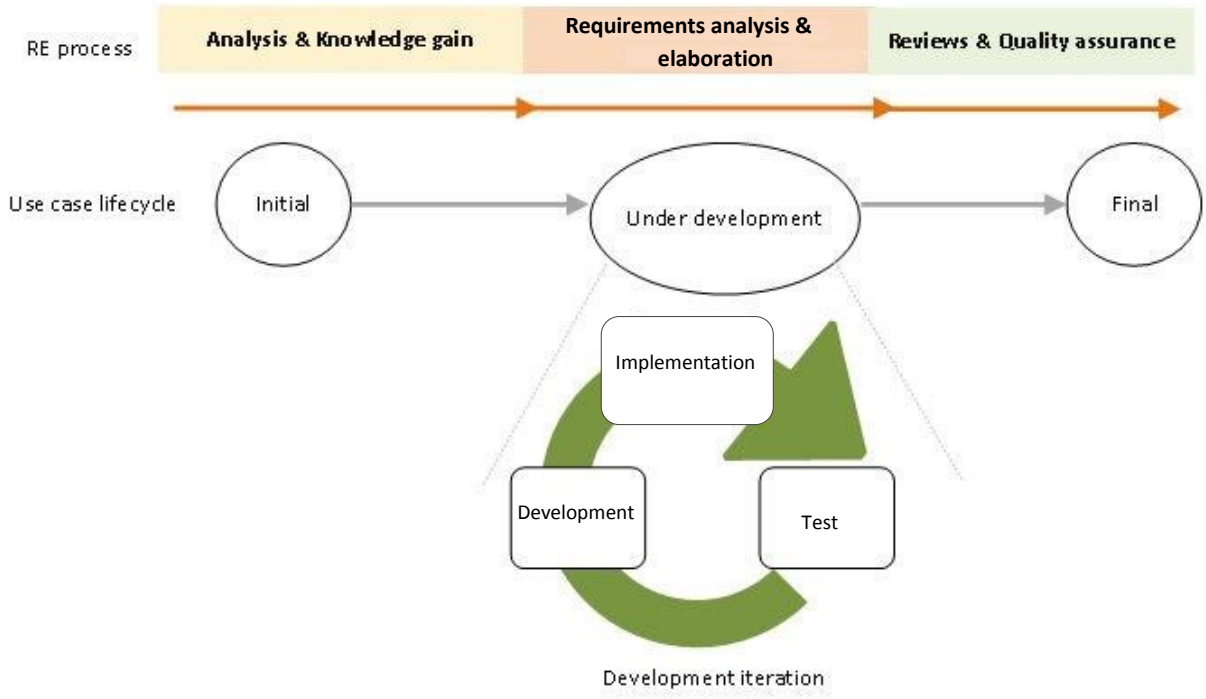

Figure 24 Use cases lifecycle aligned with the RE process and development iterations

The findings reported in this section and its sub sections are based on:

- Study of the selected use cases and understanding the requirements described by them
- Changes history of use cases and analysis of changes (Chapter five)
- Interviews with responsible developers for use cases
- Interviews with the system architect and project manager
- Study and investigation of other techniques and requirements artifacts
- System documentation of the CDR
- Literature related to use cases, their limitations and benefits

6.2.1 Initial stage

Use cases in the initial stage contained a high-level description of the functionalities, in the form of short sentences in the Description element. The other elements of the use cases (except for name and Id) had no or little content. All use cases were created based on the use case template found in section 11.1. Figure 25 shows an example of a use case in the initial stage.

A major part of the use cases were identified and initiated in the preliminary analysis phase of the RE process. The snapshot of the package diagram (Figure 26, taken on 28.01.2013) shows the functional areas and use cases that were identified in the initial phase of the development.

Id	UCImportereDodsfallFraIris
Navn	Importere dodsfall
Trigger	DÅR saksbehandler velger å importer en lot av dødsfall til Iris.
Beskrivelse	Snapshot Overskrive Slett fra IRIS Status – ligger ikke lengre i IRIS Mapping på tilbakeføring til domene
Pre-conditions	
Post-conditions	
Main flow	
Alternate flow	
Exceptional flow	
Issues/Notes	

Figure 25 example of a use case in the initial stage

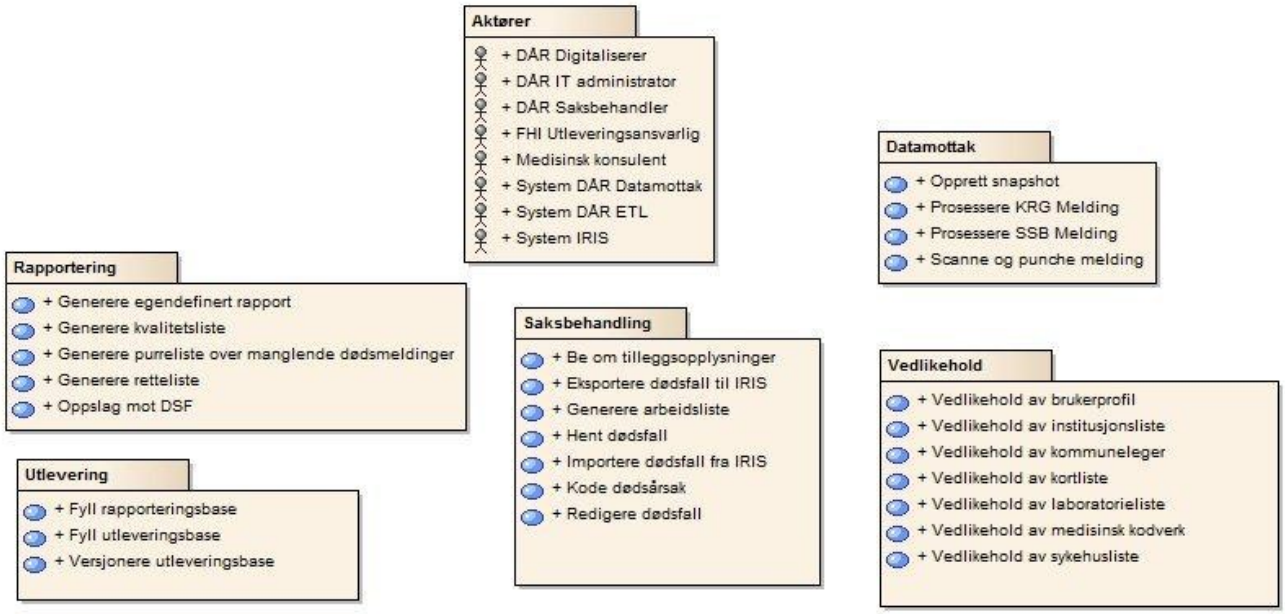

Figure 26 Package diagram, snapshot from 28.01.2013

The fact that the solution team had previous experiences from the development of other health registers and knowledge of critical tasks such as data capture, data processing and reporting had a considerable effect on quick identification of the functional areas and use cases. However, use cases were removed and new use cases were created during the progress of understanding and analyzing the requirements. The relationships between use cases were also changed. Three Snapshots of the use case diagram of the Data processing functional area taken on 06.11.2012 (Figure 27), 10.11.2012 (Figure 28) and 13.01.2013 (Figure 29) address examples of these changes. New use cases such as *Export records to IRIS* and *Import records from IRIS* were created while the two existing use cases in the first version (Figure 27), (*Code deaths, Validate registration*) were removed. In the third version (Figure 29), the new use case, *Create snapshots* was created and included in *Import from IRIS* and *Modify death.*

The lifetime of use cases in this stage was dependent on the priorities and the criticality of the features specified by use cases.

Use cases in their initial form were not used to communicate and exchange knowledge with the stakeholders. Use cases in this stage served as placeholders for the main features to be built. As such, they provided an overview of needed functionalities.

Figure 27 Data processing use case diagram 06.12.2012

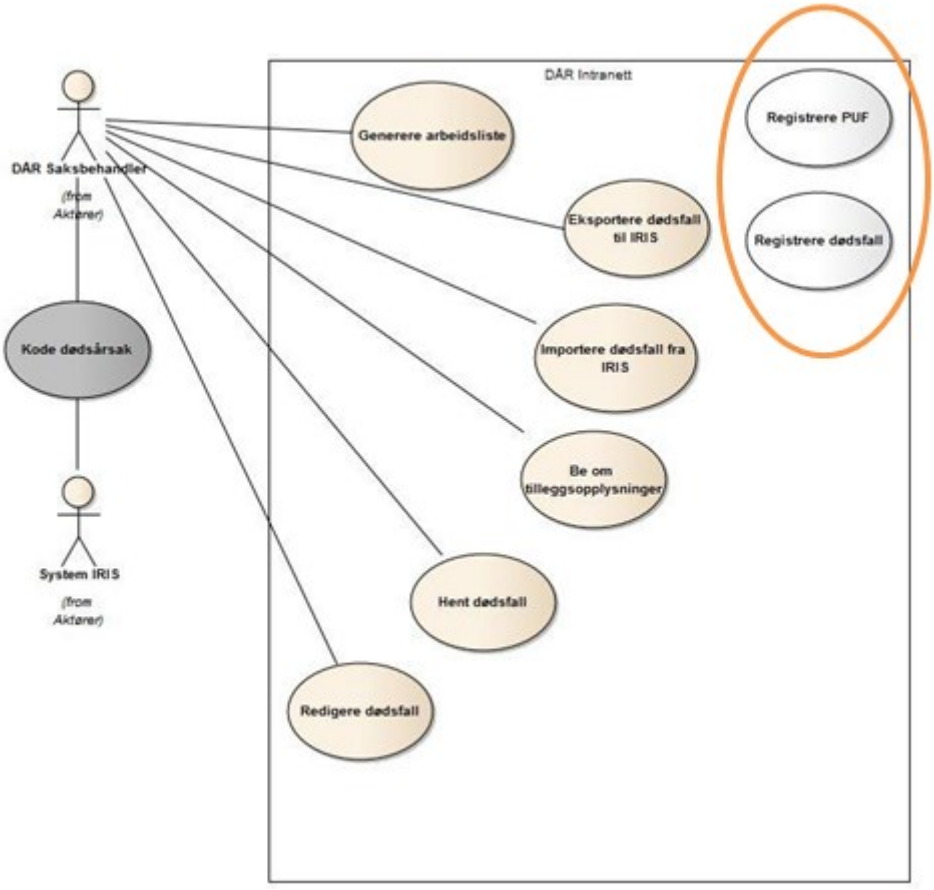

Figure 28 Data processing use case diagram 10.12.2012

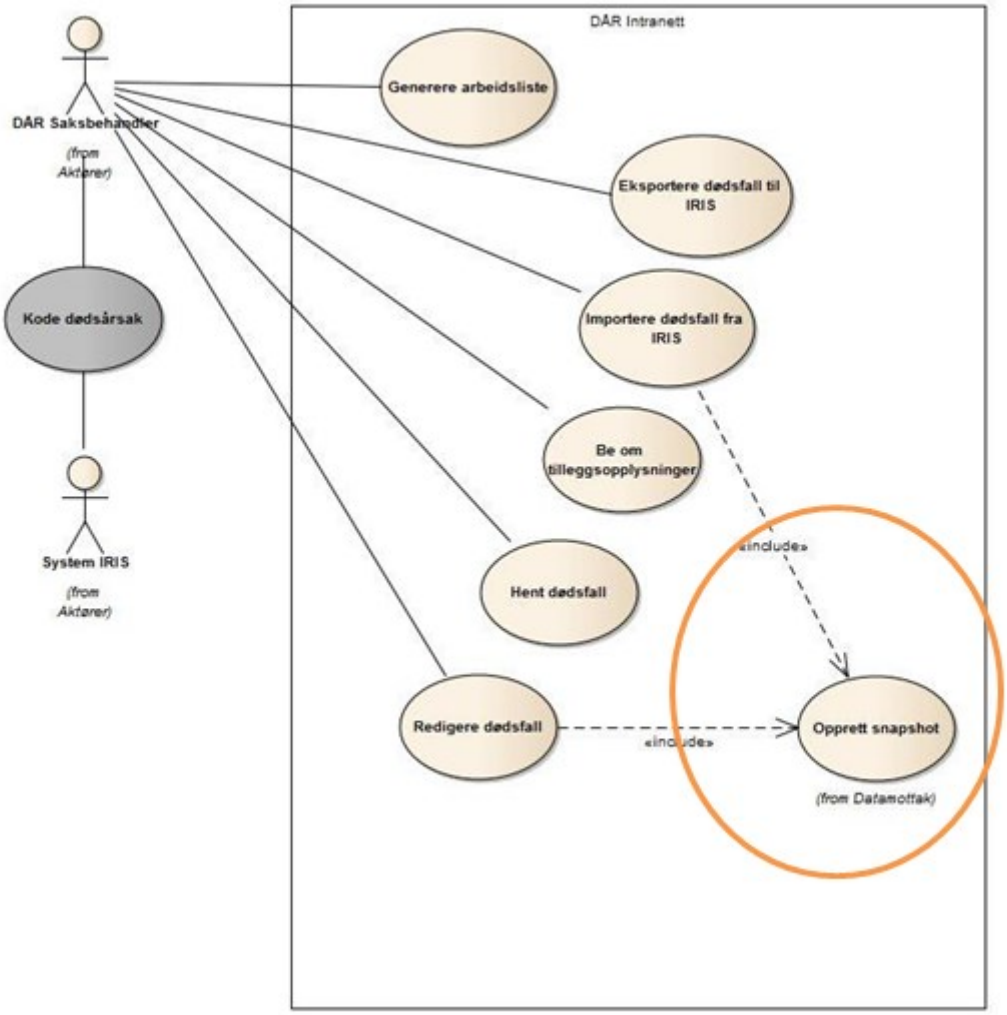

Figure 29 Data processing use case diagram 22.01.2013

6.2.2 Under development stage

"Use cases were not detailed at the start of iterations; details were mostly understood and discovered under the development." (From an interview with a developer)

The development stage in the use cases lifecycle started with the initiation of development-iteration and when the use cases that were planned to be developed in iteration, were assigned to the responsible developers.

As an input to design, development and test activities, a use case was iteratively elaborated and adjusted (with corrections and necessary changes) in the content of its elements by the responsible developer to become as correct and complete as possible specification of the requirements (Figure 24).

Other requirements artifacts were developed in parallel with the development of use cases during the iterations. The details captured by other requirement artifacts were beneficial and sometimes necessary to use cases to become complete and ready for implementation. Examples of such use cases and supplementary artifacts are provided in 6.3.1-6.3.3.

The lifetime of use cases in this stage continued until their implementation was considered as completed according to the scope of the functionalities and approved by the stakeholders. Three versions of the use case *UC Import records from IRIS* during the development of this use case are provided in section 11.9.

6.2.3 Final stage

The final stage in the use cases lifecycle started when the iterative process of use case elaboration was finished and use cases became adequately complete.

At this stage, use cases were reviewed and refined (if necessary) according to the actual implementation. Examples of such refinements are observed in the Description element where additional descriptions of the solution were added (e.g. in *UC Search for records*). The steps of main flows were also detailed (e.g. in *UC Fill Delivery Database*).

Another example was the user interface specifications that were removed from Description elements and substituted with references to the interaction design and domain model (e.g. in *UC Export records to IRIS* and *UC Import from IRIS* use cases, 5.3).

In the final stage use cases were prepared to be converted to documentation of the technical solutions for the functional requirements. Finally use cases were included in the system documentation to describe the functionalities of the system.

6.3 Use cases categories

After studying selected use cases – to understand the requirements they described - and analyzing their changes histories – to understand how they were developed -, three categories of use cases were identified:

1) Use cases with limited user interaction (6.3.1)
2) Use cases with user interaction (6.3.2)

3) Use cases with no user interaction (6.3.3)

In the following sections the evolution of use cases in each category and the challenges encountered by use cases and how they were addressed are explained.

The aspects that are investigated to understand the evolution and limitations (or challenges) of use cases are the following:

- The challenging part of the requirements (related to the studied use cases)
    - The challenging parts are identified with respect to the types of requirements
- Usage of other (supplementary) techniques during the development of use cases
    - To investigate the limitations and challenges encountered by use cases and how they were addressed by other techniques
- Use cases changes
    - To understand application of use cases over time and how use cases were elaborated, and if the changes could be related to challenges or limitations
- Usage of Description element
    - The Description element was used to specify different aspects of the requirements or solutions. Hence the content of this field was given attention to in the analysis of use cases.
- Ability of use cases in analyzing, capturing and specifying requirements
    - To evaluate use cases and compare to other techniques in these activities

6.3.1 Use cases with limited user interaction

UC Export records to IRIS and *UC Import records from IRIS* (5.3.1.1 and 5.3.1.2) are categorized in this group.

These use cases involved integration and data exchange with IRIS (61) which was used for classification of causes of deaths.

6.3.1.1 Challenging part of the requirements

The challenging part of the development of the import and export functions was to specify the respective fields in the CDR and IRIS and translation rules to convert fields' values by data exchange between the systems. These specifications required wide knowledge of IRIS data structure, fields and their values (from interview with the system architect).

6.3.1.2 Description element

The Description element in the use cases was used to include:

- IRIS integration specification (Figure 30)
- GUI specifications (Figure 31)
- Specify conditions that could lead to alternate flows (pre-rejection rules for records in IRIS, in *Export records to IRIS)*
- Specification of the alternate flow for second-time export of a record (in *Export records to IRIS)*

- Specify situations where pre-conditions of the use case was not true (the records that could not be imported back from IRIS) (in *UC Import records from IRIS)*

Dodsfall.Ulykke		
Dodsfall.Ulykke	Ident.MannerOfDeath = 2 Ident.Status = Rejected Ident.Reject = "MainInjury"	Dersom ulykke-entiteten ikke er null er dødsfallet en ulykke. Da blir manner of death 2 (ihht iris-dokumentasjon). Settes som rejected i IRIS. Se issue 1.
Tidspunkt	Ident.DateOfInjury	
Tekst	Ident.FreeText	Appender teksten til fritekstfeltet i Iris. Ingen egen felt for ulykketekst

Figure 30 IRIS integration specification table in the Description element of Export and Import use cases (5.3)

GUI

Saksbehandlere kan overføre dødsfall til Iris tre steder i applikasjonen. På forsiden kan man hurtigoverføre dødsfall basert på kun tre valg: *Ukodet/Initial med kun ett dokument, Rejected (flere dokumenter mulig)* og *antall*.

I tillegg til forsiden kan man laste inn lister med dødsfall fra applikasjonens søkeside. Her får man flere valg på avgrensing av utvalget. Kriteriene man bruker for å søke opp dødsfall for overføring til Iris samles i et internt objekt, heretter kalt *DodsfallQuery*.

Kriterier	Default verdi	Kommentar
Kodingstatus	Blankt	Sjekkboks. Statuser er Ukodet, Initial, Rejected, Under Koding og Final.
Oppfølging	Blankt	Sjekkboks. Mulige valg er I, K, Z, F, C, M, G, O, E og T. Disse er interne oppfølgingsstatuser på dødsfall.
Sist endret kodingstatus	Alle saksbehandlere	Radoknapp. Her velger man hvilken saksbehandler som har sist forårsaket en endring på dødsfallet. Valgene er samtlige saksbehandlere pluss et valg for «Alle saksbehandlere».

Figure 31 GUI specification in the Description field of the Export to IRIS use case (5.3)

6.3.1.3 Use cases elements without specification

Exceptional flow, post-conditions and pre-conditions were not specified in *Export records to IRIS*. Post-conditions and pre-conditions were removed and included in the main flow during the elaboration of the use case (Table 11).

Alternate flow and exceptional flow were not specified in *Import records from IRIS.*

The summary of the elements that were not specified is provided in Table 19.

Use case	Elements removed or not specified	Comments
UC Export records to IRIS	Exceptional flow (removed)	
	Post-conditions (removed)	Was specified in the main flow
	Pre-conditions (removed)	Included in the main flow during the elaboration (refinement) of use cases
UC Import records from IRIS	Alternate flow (not specified)	
	Exceptional flow (not specified)	

Table 19 Elements not specified in Export and Import use cases

6.3.1.4 Critical changes

In *UC Export records to IRIS,* Alternate flow, Description and Main flow elements were changed by addition of alternative scenarios in Alternate flow, IRIS integration specification in Description and addition of update rule of coding status in the Main flow. These changes are considered as of higher importance compared to other changes in the use case.

In *UC Import records from IRIS* Description, Main flow, Post-conditions and Pre-conditions were changed due to changes in specification of IRIS integration, addition of update rules of coding status and addition of pre- and post-conditions.

A summary of abovementioned changes is provided in Table 20. The reasons for the importance of the changes are explained in the Changes column.

Use case	Elements with critical changes	Changes
Export records to IRIS	Alternate flow	Addition of alternative scenarios
	Description	IRIS integration specification
	Main flow	May reflect changes in the implementation (related to coding status)
Import records from IRIS	Description	IRIS integration specification
	Main flow	May reflect changes in the implementation (related to coding status)
	Post-condition	Addition of post-conditions
	Pre-condition	Addition of pre-conditions

Table 20 Critical changes in Export records to IRIS and Import records from IRIS

6.3.1.5 User interaction

The user interaction in these use cases was limited to selecting the search criteria to obtain the desired records for export and import. The GUI specification which was updated in Description element during the development of the use cases was removed and substituted with the reference to interaction design in the final stage (6.2.3).

The export function (on the left side) and import function (on the right side) are shown in Figure 32.

Figure 32 Screenshot from CDR intranet, Export To IRIS (left section), Import From IRIS (right section) and Simple search function (in the middle)

6.3.1.6 Other techniques

The IRIS user manual and system documentation were reviewed by the developer.

A sequence diagram (Figure 40) was developed to analyze the interaction between the CDR and IRIS. The details captured by the diagram were later incorporated in the scenarios of the *Export to IRIS* and *Import from IRIS* use cases (5.3).

The coding statuses of the records were changed by Import and Export functions. The details of coding statuses and valid status transitions involved in these use cases were provided by user stories and the state diagram (explained in 7.1.9 and 7.1.11 respectively).

6.3.1.7 Highlights of changes review

Analyses of the changes history of these use cases revealed that Description and Main flow were more frequently changed than the other elements. The majority of changes in Description concerned changes in the IRIS integration and GUI specification, whereas changes in Main flow were caused by two reasons; 1) adjustments in additional description of the components and parameters and 2) changes that could be related to changes in implementation of functions such as update of coding statuses in the CDR by export and import (Table 11, Table 12).

6.3.1.8 Conclusions

The challenging part of the development of use cases was the integration with IRIS.

Description element was used to include the specification of IRIS integration, specification of GUI[18], conditions of Alternate flows (pre-rejection rules in Export use case), the Alternate flow for the second-time export of a record and the cases where Pre-conditions was not true (in Import use case). Possible reasons or explanations and comments for usage of Description for these specifications are provided in Table 21.

Usage of Description element	Reasons/Comments
Specification of IRIS integration	Could not be categorized to any other use case elements. In addition IRIS integration was only applicable for Export and Import use cases. Saying this, and according to the guideline of use case template (11.1), such specifications could be included in Description.
Conditions of alternate flows (pre-rejection rules in Export use case)	The specification in Description was not completely identical with the specification in Alternate flow.
Description of the second-time export of records (can be considered as an alternate flow)	?
Cases where pre-condition was not true (in Import use case)	Could not be categorized to any other use case elements.

Table 21 Usage of Description element in Export and Import use cases

Under the development of use cases, Pre-conditions and Post-conditions were removed and included in the main flow of the Export use case.

Changes in Description and Main flow elements of use cases were of higher importance compared to the other elements. Changes in Description may refer to implementation changes (e.g. in IRIS integration) whereas changes in the main flows were related to 1) adjustment of additional descriptions of components and parameters and 2) changes in implementation of use cases. The majority of changes in the main flow were of the first category.

The type of requirements of these use cases is considered as technical. The user interaction in these use cases was limited to the selection of search criteria for records to import or export.

The use cases served as drafts or logs of issues, questions, notes, solution suggestions and to-do lists under the development (or implementation) of the use cases.

Based on the analyses of changes, determination of the content of the elements and the extent of specification of the components of the solution were challenging in these use cases. Changes in

[18] The GUI specification was removed and replaced with reference to interaction design in the last version of the use cases. Therefore this specification is not mentioned as a part of Description element.

Exceptional flow, main flow, alternate flow, pre- and post-conditions of Export use case are examples of such challenges (Table 11).

Further the situations where Pre-conditions of use cases were not true could not be categorized to any of the use case elements.

To understand and analyze the requirements of these functions, IRIS documentation was reviewed and a sequence diagram in addition to the use cases was developed.

Use cases in conjunction with the other techniques (documents reviews and sequence diagram) were applied to analyze the requirements. Use cases could properly specify the functional solution of the requirements. However, capturing some of the details needed by the use cases required contribution from other artifacts such as the state diagram and user stories. The details of coding statuses were captured and specified by the state diagram to these use cases.

6.3.2 Use cases with user interaction

UC Search for records and *UC Request for additional documentation* (5.3.1.3 and 5.3.1.4) are examples of use cases that involved higher user interaction. The requirements of interaction design related to these features were gathered through interviews with the users and documented as user stories by the interaction designer.

6.3.2.1 Challenging part of the requirements

The challenging part of the search functionality was to identify the criteria upon which the search should be based (from interview with the project manager). The requirements of the search criteria were understood and captured by the user story that sounded as the following:

"We wish to see how many messages(records) and additional documents exist in the work pool, and whether these documents are related to an existing record and which coding status records have (initial, under progress, waiting, etc.) in order to be able to plan and prioritize the tasks within the available time." (from the documentation of the user stories, Table 4)

The challenging part of *UC Req. for additional documents* was the definition of the scope of the function. Changes in Description, Main flow, Pre- and Post-conditions elements of *UC Req. for additional documents* showed that the scope of the function was changed under the development.

6.3.2.2 Description element

The Description element of the use cases was used to include the following:

UC Search for records (11.3.3)

- Simple- and advanced search
- Usage of AND operator between the search criteria
- Limitation of number of search results
- Sort possibility on all of the results (columns)

UC Request for additional documents (11.3.4)

- Choice of the type of additional documents
- Possibility for uploading documents
- Possibility for opening scanned documents related to a record

6.3.2.3 Use cases elements without specification

Alternate flow, Exceptional flow and Pre-conditions (only in the Search use case) were not specified in the use cases. The Post-conditions of the search use case was removed due to usage of extended use case (*UC Search with identity*). (5.3.1.3 and 5.3.1.4)

Use case	Elements removed or not specified	Comments
UC Search for records	Alternate flow	
	Exceptional flow	
	Post-condition (removed)	Due to usage of extended use case (UC Search with identity).
	Pre-condition	
UC Req. for additional documents	Alternate flow	
	Exceptional flow	

Table 22 Elements not specified in Search and Req. for additional doc use cases

6.3.2.4 Critical changes

Post-conditions and Main flow elements of use cases were changed according to the changes in the scope of the function. The summary of the elements with critical changes is provided in Table 23.

Use case	Elements with critical changes	Changes
UC Search for records	Post-condition	Was removed due to the usage of extended use case (*UC Search with identity*).
UC Req. for additional documents	Main flow	Changes in the scope of the function
	Post-condition	Changes in the scope of the function (from generating a letter of request changed to providing information about the deceased and death in the tabular format that could be used in the request letter by a manual process)

Table 23 Elements with critical changes in Search and Request for additional doc. use cases

6.3.2.5 Other techniques

Development of the search use case was dependent on the identification of unique coding statuses and follow-up queues[19]. The state diagram (7.1.11) was developed to capture and specify these requirements.

[19] Follow-up queues were used to tag records in different groups such the group for the records that needed additional information from cancer registry.

In addition, design prototypes (7.1.10) developed by the interaction designer were used by the developers to implement the functions according to the desired design (both search and request for additional documents).

In case of search function, combinations of search criteria (e.g. coding status and messages) and how search results should be presented were developed based on the specification of the prototypes.

The design of the simple search function and results is shown in Figure 33.

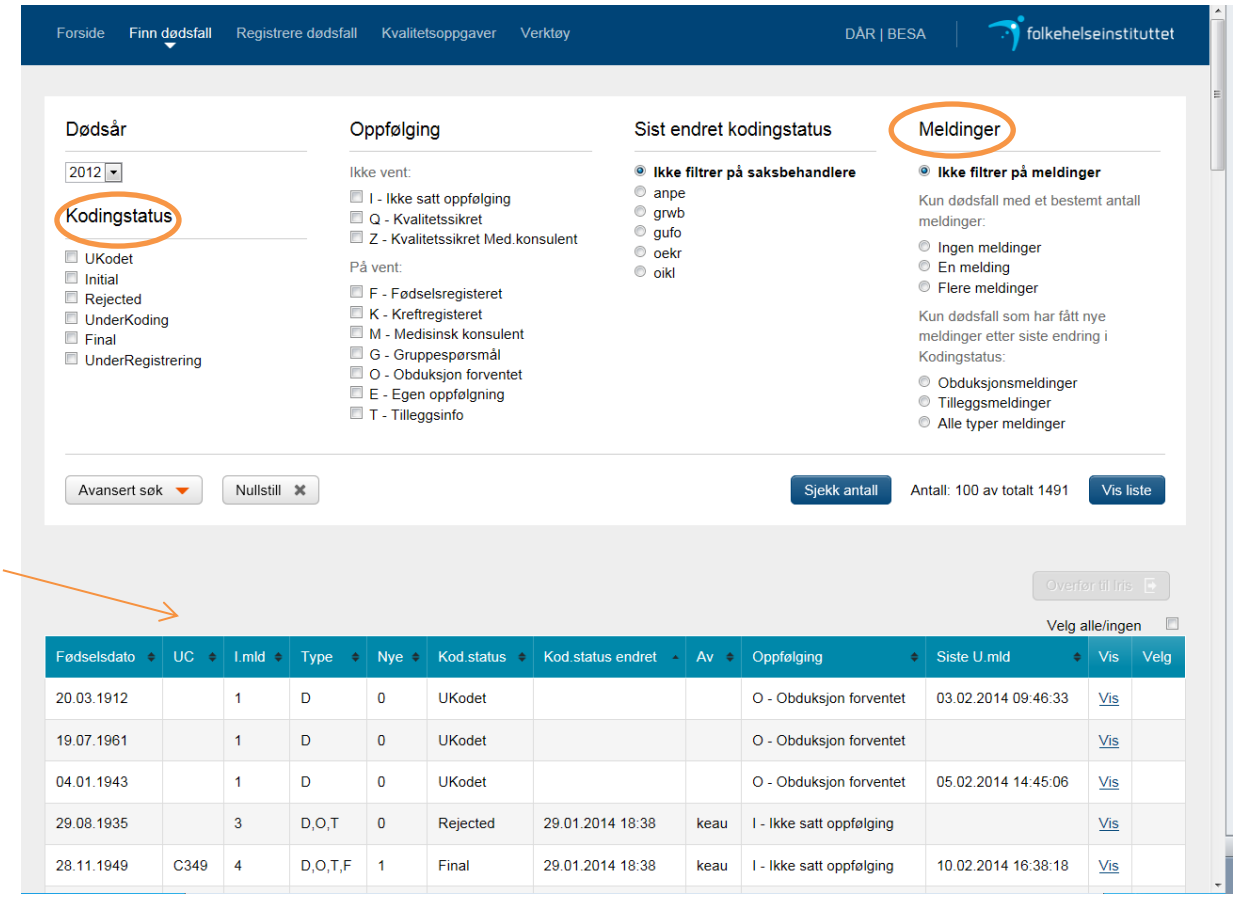

Figure 33 Simple search function and search results

6.3.2.6 Highlights of changes review

Reviewing the changes history of the search use case, shows that there were no changes in the Main flow of the search use case (Table 13). In contrary, the design prototypes for the search feature were updated at least ten times during the development of the design requirements (ref. design prototypes history log, Table 4). Two examples of the design prototypes for advanced search function are shown in Figure 46 and Figure 47.

6.3.2.7 Conclusions

The challenging part of the development of the search function was to identify the search criteria whereas determination of the scope was challenging in *Request for additional documentation* function.

Description element was used to include the description of the implemented solutions for the functions. Changes in Description reflect changes in the implementation and scope of the functions.

Post-condition of the search use case was removed due to the usage of the extended use case.

Changes in Main flow and Post-conditions were mainly related to changes of the scope of functions.

The type of the requirements of these use cases is considered as more user-dependent. Requirements of interaction design were highly involved in these use cases.

User stories and design prototypes contributed to analyze and capture the search criteria and interaction design requirements while the state diagram provided the specification of coding statuses and follow-up queues to the search function. As such use cases were not applied to analyze and capture the requirements of this category. In addition, design prototypes facilitated the realization of the solutions and communication between the solution team and users.

Use cases by themselves were insufficient to specify the search and request for additional documents functions. The state diagram and interaction design were referred to in the use cases.

6.3.3 Use cases with no user interaction

Use cases from Data capture (5.2) and ETL (5.4) functional areas belong to this category since the primary actors in these use cases were not a human actor. The interactions in these use cases were executed by the system users; System CDR Data collector and System CDR ETL.

Although the primary actor in the use cases were of the same type, the Data capture and ETL use cases differed in the type of requirements and how they were developed. Therefore they are described separately in the next sections (6.3.3.1 -6.3.3.2).

6.3.3.1 Data capture

The use cases of Data capture area (5.2.1) described how the six types of messages sent by various stakeholders were processed and mapped to the respective data fields in the CDR.

The same developer was responsible to develop and implement all the use cases of the Data capture functional area.

6.3.3.1.1 Challenging part of the requirements

The challenging part of the development of these use cases was to handle and resolve mismatches and conflicts of mapping between fields of the CDR and messages (from interview with the developer).

The development of the use cases were mainly about the following:

- Specification of the data fields in the messages [20]
  Example of such a specification for Personal number (Fødselsnr) on position 1 in the messages is shown in Figure 34.
- Identification of respective fields in the CDR and the rules of data translation
  An example is shown in Figure 35.

[20] D,L,O,T, A and U stands for the six types of messages.

- Distinguishing the identical parts of the use cases
  The identical parts of the six use cases were identified (by the developer) and specified in the generic use case (*UC Process SSB Message*) which was referred to in the Exceptional flow, and Description of the six use cases.

SSBRecord

Representerer innholdet i en record som er blitt punchet basert på papirskjema hos SSB. En SSBRecord kan representere en dødsmelding, lensmannsmelding, obduksjonsrapport, tilleggsmelding eller annet dokument.

- Makslengde på strenger er gjeldene, også for selve puncheapplikasjonen (GUI).
- Punch applikasjonen hos SSB har siden september 2012 tilgang til daglig oppdatert freg.

Attributt	Type	K						Beskrivelse
		D	L	O	T	A	U	
Fodselsnr (pos 1)	String (11)	1	1	1	1	1	1	Avdødes fødselsnummer. • Det utføres moduluskontroll på all fødselsnummer. • Det utføres en kontroll for dødsmeldinger (ikke andre) på at personen er registrert som død i Freg og at dødsdato er lik med Freg.

Figure 34 Specification of data fields in SSB messages

SSB record	Domenemodell	Regel
Fodselsnr	Avdode. Ident	Må være et gyldig fødselsnummer
Dodskommune2	Dodsfall. Kommune	Verdi: 2599 (utland) — Oversettes til: 2590 (Folkeregisterets kode for utland) Verdi: Ikke utfylt — Oversettes til: 9999 (Ukjent) Dersom kommunekoden ikke finnes i Kommune opprettes kommunen.

Figure 35 Respective data fields in the CDR and SSB messages in addition to translation rules

6.3.3.1.2 Description element

Description element of the generic use case (*UC Process SSB Message*) was used to specify:

- General rules of mapping between messages and the CDR fields
- General rules for message validation
- Handling mandatory data fields of the CDR
- Criteria for creation of new records
- Message receipt with status back to sender in case of error in the messages (also specified in exceptional flow)
- Description of error handling
- Criteria for identification of duplicate messages
- Special fields in the CDR

Description element of the six use cases (special cases of the generic use case) contained:

- References to the generic use case for validation rules, error handling, duplicate messages and special fields
- Specification of messages special validation rules and fields (e.g. in *UC Process SSB Death message*)
- References to other use cases (e.g. in *UC Process SSB Death message From Police*)

6.3.3.1.3 Use cases elements without specification

Alternate flow was removed due to the changes in handling duplicate messages received on a record.

Pre-conditions of none of the use cases was specified.

6.3.3.1.4 Critical changes

Alternate flow, Main flow and Post-conditions of the use cases[21] were changed due to the changes in the requirements of the use cases (e.g. update rules of existing records, registration in notification log, handling duplicate messages).

6.3.3.1.5 Highlights of changes review

The results of changes history analyses (Table 9) show that Main flow was most frequently changed. The changes mainly concerned changes in the requirements (11.4.5)

6.3.3.1.6 Other techniques

The specification of the fields of the messages was documented in a Word document, named SSB Data contract. The specification of the respective fields in the CDR and translation rules between the messages and CDR was specified in a Word document, named Mapping of SSB record to domain model.

Further handling changes in the coding status of existing records which were updated by new incoming messages, needed the details of the coding statuses and their valid transitions. These details were specified by the state diagram.

6.3.3.1.7 Conclusions

The challenging part of the development of use cases was to specify the mapping of messages to fields of the CDR.

Description element of the generic use case was used to include the common descriptions that applied to all the special use cases (six use cases for six types of messages) such as general mapping rules, messages validation rules and error handling.

Description element of the six special use case contained references to the generic use case in addition to special specifications of validation rules and fields related to the use cases.

Pre-condition of none of the use cases was specified, while Alternate flow was removed from the six use cases due to the changes in handling duplicate messages received on a record.

[21] Six use cases for six messages

The changes in Alternate flow, Main flow and Post-conditions of the six use cases were related to changes in the requirements, as such these changes are considered as more critical compared to other changes. Main flow was more frequently changed than the other elements.

The type of requirements of the use cases is considered as technical. The primary actor of the use cases was *System CDR Data capture* (system user). However relevant stakeholders (the Data capture unit and stakeholders with knowledge of messages and the message registration system) were involved to specify the fields and mapping rules of messages.

Use cases were applied to analyze and capture the requirements. However, details of the coding statuses were captured and specified by the state diagram.

The complete specification of the data capture solution included two other documents (SSB Data contract and Mapping of SSB record to domain model) in addition to the use cases.

6.3.3.2 ETL

The ETL use cases (5.4) described the requirements concerning i) the data warehouse solution and ii) data synchronizations with external data sources.

The same developer was responsible to develop and implement the ETL use cases.

6.3.3.2.1 Challenging part of the requirements

The challenging part of the data warehouse use cases was to specify the information requirements which included the definition of the data fields that should be accessible, the sources of data (databases) and representation of data in the target database (from interview with the responsible developer).

6.3.3.2.2 Description element

The following were specified in Description element of *UC Fill Delivery database*:

- The source and target databases and their schemas
- Data upload method
- Data transfer method
- Description of the service table which contained among the others statuses of the transfer jobs
- Description of the service parameters

In *UC synchronize ICD codes*, Description was used to describe which files to download and how the ICD codes (in specific tables) of the CDR should be updated by new codes.

6.3.3.2.3 Use cases elements without specification

Alternate- and Exceptional flow in the use cases were not specified.

6.3.3.2.4 Critical changes

Identification of changes that could be related to changes in the requirements or scope of the functions was difficult in these use cases. Main flow of both of the use cases was totally removed and replaced with new and different steps compared to the previous ones. The few available versions of the use cases imply that use cases were not often updated. Therefore the differences between the versions were more comprehensive.

6.3.3.2.5 Highlights of changes review

Description element was more frequently changed in the use cases (5.4).The changes included addition of complementary specifications of the solutions (11.4.6).

6.3.3.2.6 Other techniques

The information requirements of *UC Fill Delivery database* were documented in the tabular format in two Word documents (domain model for delivery database and domain model for star schema) to which use cases referred. The domain model for the target databases (delivery database) described 1) the data fields that should be accessible in the database and 2) the source databases where the data come from. The facts- and dimension tables were described in a separate Word document and in a tabular format, named the star schema.

The specification of the following components of *UC Synchronize ICD codes* was documented in the *Invariant* element added by the developer to the use case in order to achieve a more complete specification of the solution(from interview with the developer):

- Specifying the source files (e.g. ICD CSV files)
- Specifying the target (e.g. tables, their structure and schema) that are loaded and compared to the data in the CDR tables
- Specifying the rules of data conversion by synchronization
- Handling different ICD codes versions

6.3.3.2.7 Conclusions

The challenging part of development of data warehouse solution was to specify the information requirements.

Description element was used to describe the components of the technical solutions (for data warehouse solution and ICD codes synchronization).

Alternate- and Exceptional flow in the use cases were not specified.

Identification of critical changes was difficult in the use cases due to few available versions of use cases, and that the differences between the versions were comprehensive. Main flow of both of the use cases was totally removed and replaced with new and different steps compared to the previous ones. This type of changes can be related to refinement of use cases to reflect the final and actual implementation of use cases.

Description element was frequently changed in the use cases (5.4).The changes included addition of complementary specifications of the solutions (11.4.6).

The type of requirements of the use cases is considered as technical rather than user-dependent. The primary actor of the use cases was System CDR ETL (system user). However specification of the fields that should be accessible and representation of data in the data warehouse solution required user involvement.

For Data warehouse solution: Use cases contributed poorly to analyze and capture the information requirements (concerning definition of the data fields that should be accessible, the data sources and representation of data in the target database) of data warehouse solution. The Description element

was used to include the necessary specification of the technical solution of data warehouse solution. The mapping table in the domain models (related to delivery database) was used to capture the information requirements.

For Data synchronization solution: use cases were applied to analyze, capture and specify the solutions for data synchronizations use cases. However a new use case element, Invariant, was used to completely describe the technical solution of the function.

6.4 Results

This section provides a summary of the results presented in 6.2- 6.3 to answer the first question of the study:

RQ1: How was the use cases technique applied in different activities of the RE process in the CDR project and which challenges were encountered by use cases?

Use cases were initiated by high-level and short descriptions of the functionalities needed in the new solution of the CDR. Use cases in the initial stage served as placeholders for the features to be developed. As such, they provided an overview of the features and were used as inputs to prioritize and plan development tasks and activities.

Use cases were elaborated and implemented in the development iterations by the responsible developer(s). The analysis of the requirements and elaboration of use cases were conducted iteratively during the iterations. The main activities of the analysis were to understand the essence of the requirements, to gather related facts and data and capture the details concerning use cases. Use cases were eventually completed with the captured details during their development.

Development of use cases continued until their implementations were considered as completed according to the scope of the features or functions agreed between the solution team and stakeholders[22]. The final stage of the use cases lifecycle was thereafter initiated by being reviewed and refined so that they represented the actual implementation and the technical solution of the features.

Use cases were developed differently depending on the type of requirements they described and the developer(s) who implemented them. That the Data capture use cases specified by the same developer had Exceptional flow while other use cases specified by other did not, was an example of dependency of use cases specifications to developers. Usage of Invariant element by one of the developers is another example.

Three categories of use cases were identified among the studied use cases: 1) use cases with limited user interaction, 2) use cases that involved user interaction and 3) use cases with no user interaction.

[22] At the end of iterations, the relevant stakeholders related to the implemented functions were informed about the implemented functions through demos. In this way, the stakeholders could confirm that the right functions were developed, in addition the stakeholders could try out the new functions and give feed backs about necessary changes. When the stakeholders approved the functions and functions were tested by the solution team, the implementation was considered as completed.

The type of requirements of the first category of use cases was more technical rather than user-dependent. The use cases of the first category served as drafts or logs of issues, questions, notes, solution suggestions and to-do lists under the development of the functions described by these use cases. Use cases were utilized to analyze the requirements in conjunction with other techniques (documents reviews and sequence diagram). Further, capturing some of the details needed by the use cases required contribution from other artifacts such as the state diagram and user stories.

The second category of use cases was more user-dependent. Use cases of this category were changed fewer than the first category (compared by the number of available versions of these use cases). Use cases contributed poorly to analyze and capture the requirements of the features of this category. User stories, design prototypes and the state diagram were more beneficial to analyze, identify and capture the details of these requirements. Use cases by themselves were insufficient to specify functions of this category. The state diagram and interaction design were referred to in the use cases.

The third category of use cases had no user interaction. Use cases of this category were selected from Data capture, ETL –Data warehouse and ETL – Data synchronization areas. The type of requirements of these use cases was technical.

Use cases were utilized to analyze and capture the requirements of Data capture area. However, details of the coding statuses involved in these use cases were captured and specified by the state diagram. The complete specification of the data capture solution included two other documents (SSB Data contract and Mapping of SSB record to domain model) in addition to the use cases.

Use cases contributed poorly to analyze and capture the information requirements (concerning definition of the data fields that should be accessible, the data sources and representation of data in the target database) of data warehouse solution. The Description element was used to include the necessary specification of the technical solution of data warehouse solution. The mapping table in the domain models of the delivery database was used to capture the information requirements.

Use cases were applied to analyze, capture and specify the solution for data synchronizations function. However a new use case element, Invariant, was used to completely describe the technical solution of the function.

Further the changes history of the studied use cases were investigated to understand how use cases were applied and developed (elaborated) through different phases of RE process. The following results obtained based on the analyses of use cases changes:

- Type of changes in various elements of use cases and in different use cases showed that the determination of the content of the elements and the extent of specification of the components of the solution were challenging. These types of changes could mainly be seen in Description and Main flow elements.
- Further, that critical changes such as changes in requirements, scope of functions and in implementation were more reflected in Post-conditions and Main flow elements.

- The majority of use cases had no Exceptional flow (all of the studied use cases except for Data capture use cases which referred to Exceptional flow specified in the generic use case *UC Process SSB message*).
- Description element of all of the studied use cases was mainly used to 1) specify the technical solution of the functionalities related to use cases and 2) include parts of requirements that could not be categorized to any other elements. Specification of situations where pre-conditions were not true is such example.
- Further, the analyses of content of Description element revealed that specifications related to Alternate- and Exceptional flows and conditions of Alternate flows were placed in this element. These types of specifications did not completely match with the content of the elements (Alternate- and Exceptional flow) if specified, but had overlaps.

Although all of the use cases were structured based on the common use case template (11.1), the content of the use cases elements and when (by which changes and how often) use cases were updated, were decided by the individual developer. The variation of the specifications given in Description element of the use cases is an example of different contents. Another example to underline that the content of use cases was decided by developers is the Invariant element which was used by one of the developers to specify the invariable part of the solution in the ETL use cases. Still another example of different application of use case elements was seen in one of the use cases where pre- and post-conditions were removed and included in the main flow of the use case.

Usage of other techniques such as user stories, state diagram and design prototypes to analyze and capture the details of the requirements stressed that utilizing multiple techniques and requirements artifacts was necessary to achieve a complete and correct specification of the requirements. Further the state diagram and domain models are examples of artifacts used to document functionalities of the CDR in the system documentation in addition to use cases.

As depicted in Table 24, artifacts such as the state diagram, user stories and design prototypes contributed to analyze and capture the details of the requirements involved in several use cases.

User stories and design prototypes were mainly beneficial to the second category user cases while the state diagram was beneficial to capture the requirements necessary for several use cases in all three categories.

The techniques that were applied in addition to use cases to analyze, capture and specify functional requirements are summarized in Table 24.

Category	Use cases	Analyzing req.	Capturing req.	Specifying req./solution
1)Limited user interaction	Export to IRIS	Use case Documents review Sequence diagram	State diagram Use case	Use case State diagram
	Import from IRIS	Use case Documents review Sequence diagram	State diagram Use case	Use case State diagram
2)User interaction and design	Search for records	User stories	Design prototypes State diagram	Use case State diagram
	Req. for additional documents	User stories	Design prototypes	Use cases
3)No user interaction	Data capture use cases	Use case	Use case State diagram	Use case Domain models
	ETL- Data warehouse	Domain models	Domain models	Use case Domain models
	ETL- Data Synchronization	Use case	Use case	Use case (with Invariant element)

Table 24 The techniques applied in analyzing, capturing and specifying the functional requirements in CDR use cases

7 Other techniques and their benefits to use cases

To answer the second question of the study:

RQ2: how were other techniques than use cases beneficial to specify functional requirements? How were the limitations and challenges of use cases addressed by these techniques?

I identified the techniques that were applied during the different activities of the RE process. This was done by investigating the requirements artefacts that were developed during the development of the CDR (mainly the first five development iterations in this study). Further I tried to find out the following:

- How the techniques were applied
    - By studying the requirements artifacts developed by the techniques
    - Through interviews with the solution team and review of documents such as presentations that referred to or included the artefacts.
- In which phases or activities of the RE process the techniques were applied
    - By identifying the time that artefacts were developed
    - Knowledge of project activities through interviews and by my own observations
- The reasons for applying the techniques
    - Through interviews
    - By knowledge of the techniques and their benefits through literature

Further, the benefits of the various artefacts were set against the type of requirements. In addition the application of the techniques was aligned with the development of use cases to investigate how the other artefacts affected use cases.

Finally the capability of the techniques to analyze, capture and specify functional requirements were evaluated.

Presentations of the techniques and the results obtained by the above mentioned approach are provided in the next section (7.1).

7.1 Techniques

The techniques applied during the development of the CDR and in different RE activities are described in this section. The techniques are not grouped by the RE activities since the same techniques were applied during different RE activities. However the capability of techniques to analyze, capture and specify functional requirements are depicted in Table 25.

7.1.1 Documents review

Studying and inspection of documentations of the old (existing) solution such as system documents, user manuals and data contracts were mainly conducted by the project manager and system architect to understand the business domain and the capabilities of the existing solution in the preliminary analysis phase.

"I skimmed through the user manual of the existing system to gain an overview of the existing solution and its capabilities…..the majority of the processes (coding, processing, reporting) are the

same in the new solution but hopefully with enhancements" (from email exchange with the system architect)

"….I had read the user doc early on. It served to help identify and clarify user goals and gave a coarse impression of which kinds of UCs needed to be defined and addressed. It gave an impression of what functions must be supported as well as functions which could also be of value (perhaps in a new form) in the new system. A good example of the latter was the reports which could be generated: almost all were for QA purposes. These are now supported – plus many more -- via the data warehouse. Other examples were functions for code maintenance purposes" (from email exchange with the project manager)

The benefits of this technique that are identified i) through interviews with different project members and ii) reviews of documents of the existing (old) CDR solution (Table 5), are described as follows:

- Useful in identification of stakeholders and critical functionalities
- Useful in modeling existing workflows and processes
- Provided hints of issues and challenges in the existing system

Document archeology is used as a technique to search for underlying requirements and is recommended to use when the existing system is being modified or renewed in the new solution ((17), chapter five).

7.1.2 Interviews

Interviewing stakeholders in various forms were conducted by different members of the solution team during the development of the CDR. Extracting requirements and exchanging knowledge with stakeholders through short or long conversations in various meetings and other occasions are also considered as non-structured or open interviews.

My findings through i) interviews with the project manager, ii) reviews of the presentations and documentations made by the interaction designer in addition to iii)reviews of the user stories revealed that:

- One-to-one and more structured interviews concerning interaction design requirements were mainly conducted by the interaction designer. The users with central roles and responsibilities were interviewed. Interviews were beneficial in understanding the roles, tasks, processes, challenges and missing features in the existing system. In addition the visions and expectations of the new system were discussed. The knowledge gained through these interviews was the basis for developing user stories.
- Examples of the visions (told by the users in the interviews) that can be related to the search feature and better accessibility of data in the CDR are shown in Figure 36 (from presentation based on interviews with users made by the interaction designer)

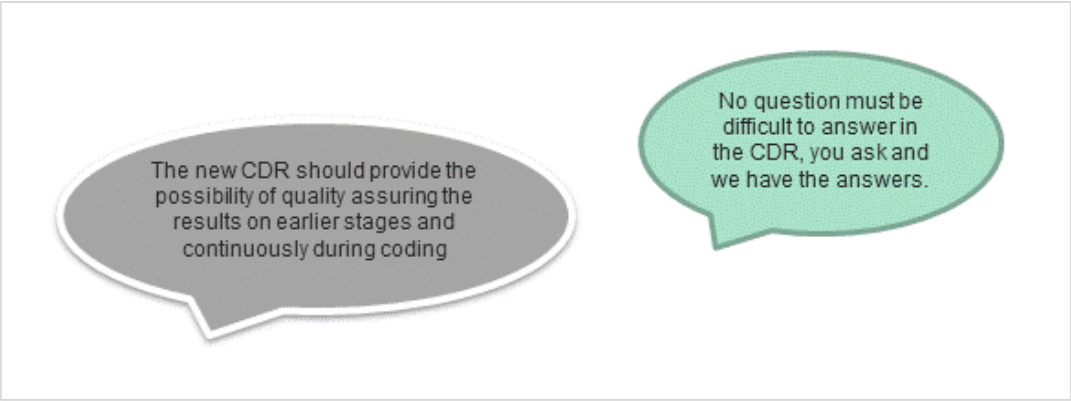

Figure 36 Users visions captured by interviews (from internal documents presented in Table 5)

Interviewing is mentioned as a technique to elicit and gather requirements that should be used in conjunction with other techniques. The approach of interviewing, skills of the interviewer (including domain knowledge, level of comprehension, writing skills), questions of the interview and knowledge of the interviewee and his or her ability to verbalize are mentioned as highly influential in the data and their quality provided by the interviews ((17), chapter 5).

7.1.3 High-level workflow

The workflow (Figure 37) was modeled to illustrate the main capability areas (such as Scanning, Punching, Data processing and QA, Reporting and Data delivery) and their related tasks (process death record, generate reports) in the order they are performed in the workflow.

The workflow was developed by the project manager based on the knowledge of the business domain and capabilities in the existing solution. The diagram was created in the preliminary analysis phase. The terms used in the diagram conformed to the terms used by the stakeholders.

The usage of this diagram in various presentations and the way it was used during the development of the CDR revealed the following benefits:

- In communication with the stakeholders to address the functional area(s) and tasks on focus.
- Used to present the overall development time schedule versus features that were planned to be implemented
- To confirm the work scope with the stakeholders

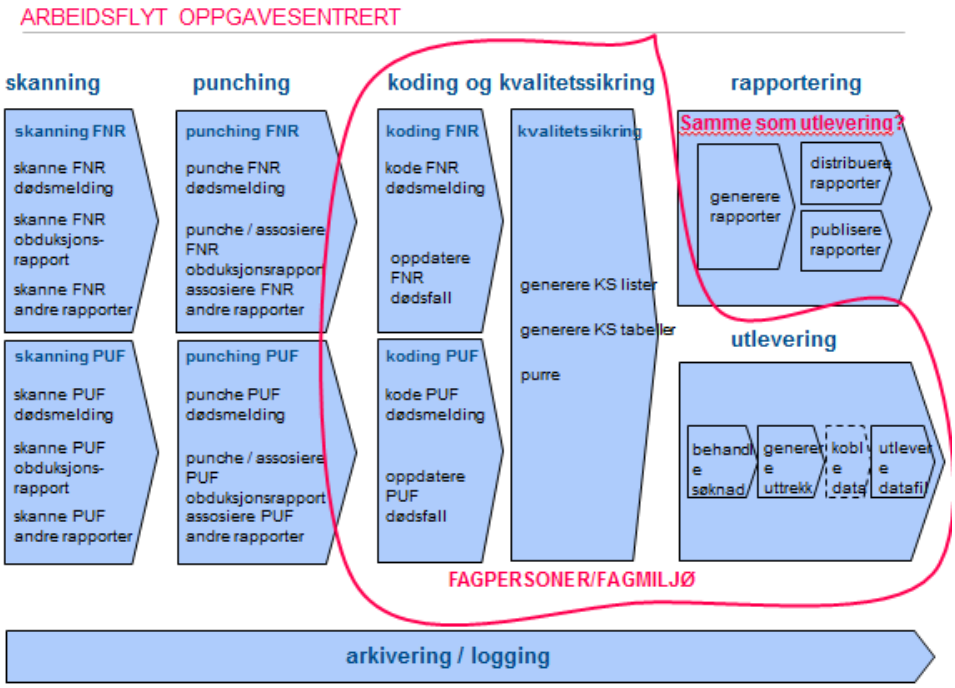

Figure 37 High-level workflow (from internal documents presented in Table 4)

The literature underlines the importance of modeling the current work as quickly as possible to ensure the correct understanding of the business between the parts. In addition the models allow the stakeholders to verify the representation and organization of the business in the new solution ((17), chapter 5).

7.1.4 Use case diagrams

Use case diagrams are described as the graphical table of contents for use cases. In addition use case diagrams provide a visualized representation of relationships between actors and use cases and among use cases (19).

Use case diagrams (4.6.2) are constructed by four elements: 1) the system boundary, 2) actors (primary, secondary or other stakeholders involved in the use cases), 3) use cases and 4) relationships between these elements. As such, development of use case diagrams necessitates the following activities which can also be considered as benefits of this technique:

- Identification of actors
- Identification of use cases and their relationships, such as *include* and *extend*
- Determination of system- and functional area boundaries
- Understanding the context in which use cases are involved
- Identification of external and internal factors influencing the system

The development unit at NIPH recommends developing and including use case diagrams as one of the necessary artifacts to provide a minimum documentation of the requirements. (Ref. technical notes available on wiki pages of the development unit)

My findings of application of use case diagrams through i) interviews with the system architect and developers who developed the use case diagrams, ii) studying use case diagrams and iii) their usage in discussions and presentations to the stakeholders conclude that:

- Use case diagrams were created in the preliminary phase and provided a high-level overview of the use cases (or features) and their relationships. In addition the boundaries of the solution and the functions of the system were discussed and communicated to the stakeholders through these diagrams.
- Use case diagrams were changed under the progress of the requirements analysis and development (elaboration). Use cases were removed or renamed or new use cases were created in these diagrams. Better understanding of the requirements through discussions with stakeholders was the main reason of the changes. Examples of such changes are depicted in 6.2.1, Figure 27, Figure 28 and Figure 29.
- The relationships between use cases were mainly changed during the requirements development. For instance when it was realized that several use cases needed to "decrypt the identity of the death person" in their scenarios, this use case was created as a separate use case and associated with the *Include* relationship.
- Use case diagrams were utilized to communicate the use cases that were planned to be implemented and their implementation priorities to the users and stakeholders. This was done by presentation of use case diagrams at the beginning of the development iterations to the stakeholders to inform them about the iteration activities and priorities (ref. presentations that includes use case diagrams)
- One use case diagram per functional area was developed in the CDR project. Example of a use case diagram is shown in Figure 38. The numbers (one and two) in the figure indicate the priorities where one is the highest.
- Use case diagrams were in addition used in discussions of the features and the functional area they belonged to and to discover missing use cases.

The limitations of use case diagrams are (19) (5) (1):

- Lacking well-defined semantics which can lead to different interpretations by stakeholders
- Controversy in definition of *include* and *extend* relationships
- Inability to represent the sequence of use cases execution
- Inability to depict the input, output data involved in the interactions

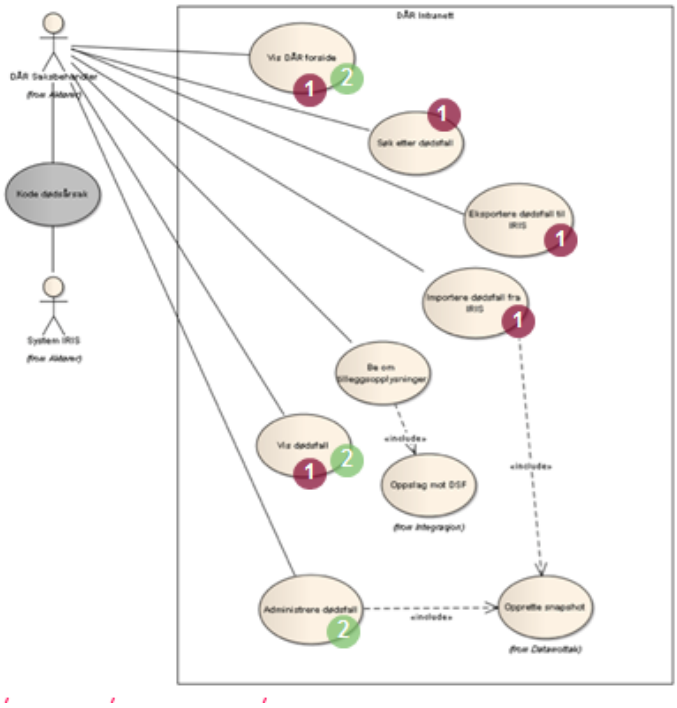

Figure 38 A use case diagram (from internal documents presented in Table 5)

7.1.5 Visualizations

Visualization is an effective technique to capture details rather than to write long and detailed texts. The benefits and superiority of visualized techniques in communication with the stakeholders, understanding and capturing requirements are commonly known and confirmed by several studies (1) (10).

Usage of videos and photos to capture moments in time is useful to show the current situation and the history and progression of the work over time. In addition, the videos or photos can be studied later, used and referred to in various documents. These techniques are particularly beneficial for distributed development (17), chapter 5.

Through i) studying of requirements artifacts, ii) status reports and iii) various presentations, the usage of visualization in the CDR development was as follows:

- During development of the CDR and in almost all the RE activities, visualization in form of diagramming and sketching on paper or whiteboard was used in user meetings, workshops and under discussions between the members of the solution team.
- In addition, photos of screen, paper and whiteboard were taken to document progressions, decisions, discussions, conclusions and agreements.
- The efficiency and time-saving character of photos in capturing the information was the most beneficial aspect of this technique.

Figure 39 shows an example of such visualizations.

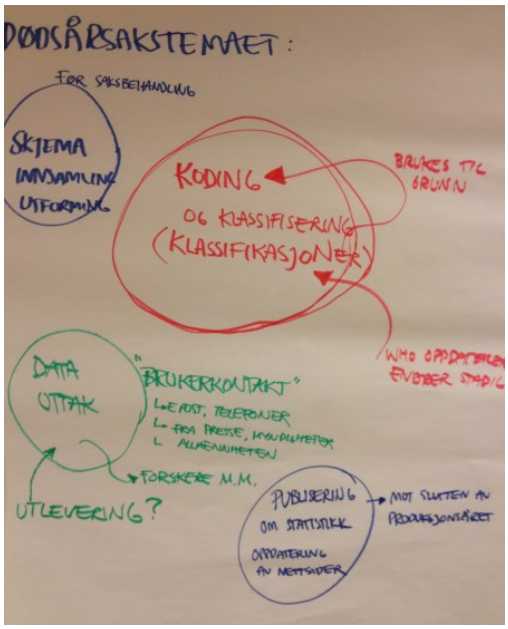

Figure 39 photo of whiteboard notes (from internal documents presented in Table 4)

7.1.6 Sequence diagrams

As implied by the name, sequence diagrams (2.6.5) are especially capable of capturing the sequence or time order of the events in the interactions between the objects of a system.

Sequence diagrams can be used both in the analysis and design phases of the development. During the design phase, architects and developers can use the diagram to understand and capture the system's object interactions, and to elaborate the overall system design (34).

Based on my understanding of the application of two sequence diagrams, one for interaction with IRIS (Figure 40) and one for modeling the overall sequences of tasks performed in the CDR workflow (11.11, Figure 49), the following can be reported:

- In the preliminary analysis phases, a high-level sequence diagram to illustrate the overall flow of the CDR tasks and how different objects (participants) interacted was developed (11.11). The diagram was beneficial to express the order in which the tasks were performed and the events by which the tasks were triggered. In addition the interactions between various participants (e.g. actors, systems, databases) were identified and verified by the stakeholders. This diagram was beneficial to understand and verify the overall flow of the interactions between the parts involved in the interactions.

- The other sequence diagram, shown in Figure 40, was used in the analysis of the requirements concerning interaction with IRIS. The details captured by the diagram were later incorporated in the scenarios of *UC Export to IRIS* and *UC Import from IRIS* use cases (5.3). The sequence diagram was beneficial in capturing actions and their order in these use cases. However, details concerning records statuses and their changes by different actions could neither be captured nor specified in this diagram.

Sequence diagrams are in addition decision free and lack information about how the events in the sequences are selected by the system or participants (34).

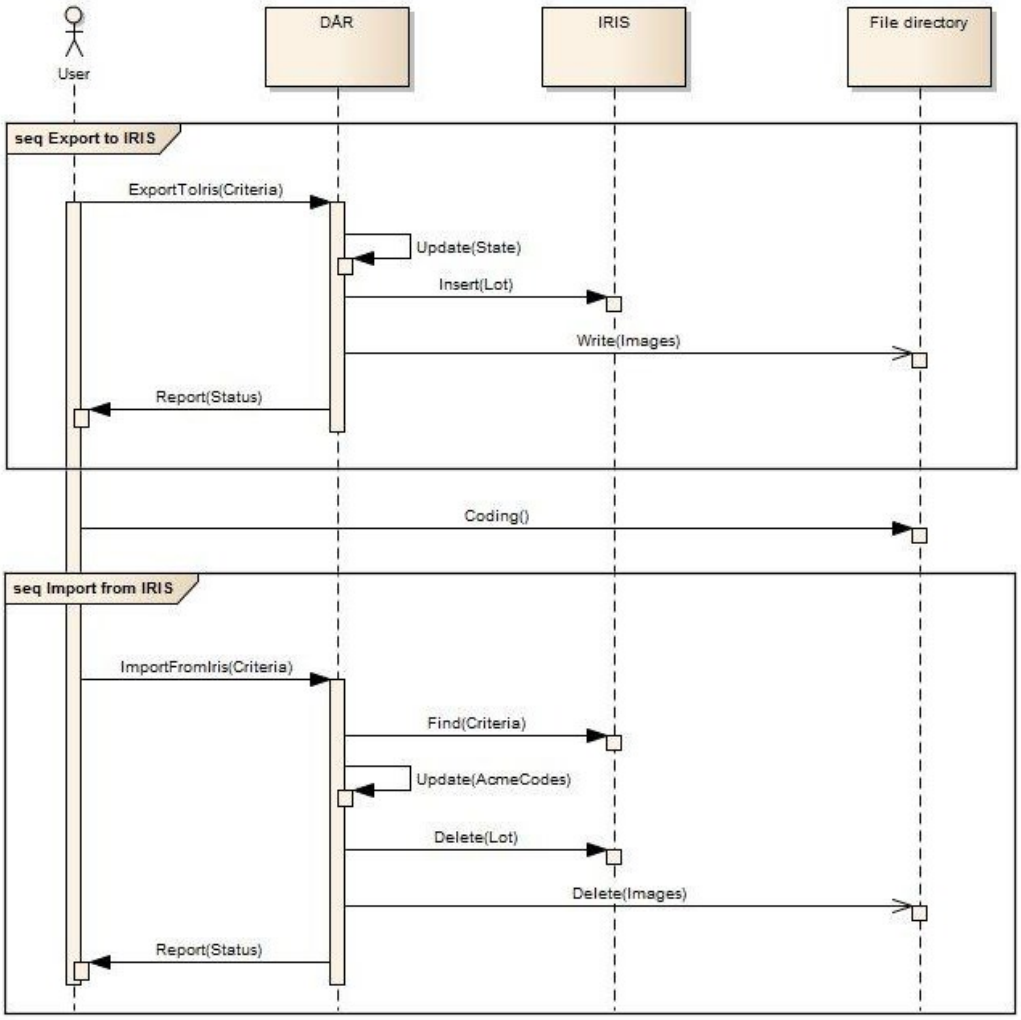

Figure 40 Sequence diagram for IRIS interaction (from internal documents presented in Table 4)

7.1.7 Direct communication

Based on my observations under the course of the thesis and through passive participation (by attending to some of the projects meetings), the following can be reported:

Direct communication was used both in the communication between the solution team and stakeholders and between the members of the solution team. The facts related to these communications are discussed below.

- Communication between the members of the solution team

  The number of the members of the solution team (3.6) and their close distance (either in the same office or in the offices next by) made the face-to-face communication possible. The fact that the members of the solution team shared experiences from previous projects and had a common language in the development domain were important to the efficiency of this type of communication.

Direct communication was used to bring up questions, under discussions, and making decisions where all the members or the relevant ones were present.

The obvious benefit of this type of communication was the elimination of the need for time-consuming documentation. However the risks of misunderstandings, forgetting the results of the discussions and decisions or informing the ones not present, underline that this type of communication should not be used uncontrolled.

- Communication with the users and stakeholders

  Direct contact with the users and stakeholders in various meetings, via phone and email was central in exchanging knowledge and capturing requirements and details under the course of the project. In addition the users were on-site one day a week in the last three months of the project to try the design and functionalities together with the solution team.

  The literature addresses that the effectiveness of such communication is strongly dependent on factors such as availability of the stakeholders, consensus among the stakeholders, trust between the stakeholders and solution team and skills of verbal communication (63).

7.1.8 User stories

Based on i) the interview concerning application of user stories with the project manager, ii) studying of the user stories and their relations to use cases, iii) history log of user stories status and iv) the definition of user stories (2.6.3) the following is found about the user stories in the CDR project:

- In the CDR project, user stories were developed by the interaction designer based on the interviews that she had conducted with different users and stakeholders. The main purposes of the interviews were to analyze and capture interaction design requirements. Reviews and prioritization of user stories were mainly conducted by the project manager.

  "User stories can be considered as examples as to why and in what way major functions need to be accessed" (from an interview with the project manager)

- The template of user stories was suggested by the project manager. Due to the time- and type specific tasks in the CDR, some new components were added to the traditional user story template.

  These additional components contributed to categorize the features and consequently to design a more user friendly and effective layout. For instance functions with the same time criteria were placed close to each other and in the right order desired by the users.

  One example of a user story structured according to the template is shown in Figure 41:

User story ID	Role	Before I start the daily work	Before I finish the day	Any time during the day	When I work with deaths in IRIS	When working with complex deaths	While we work with finishing the year	I wish to…	Because
5	Officer(user)				X	X		Have access to all related documents to a death in CDR or IRIS.	I want to be able to compare documents and manually decide which code (cause of death) is correct

Figure 41 Example of a user story. The additional components marked by red contributed to categorization of functions and a more user friendly design in web pages (from internal documents presented in Table 4)

- After gathering the user stories, they were reviewed by the project manager in order to be categorized into functional areas and to decide in which iteration they could be implemented. In addition the user stories that could be related to the use cases were identified. These user stories complemented use cases with extra details and verified the requirements that were already captured and specified by the use cases.

- Development of user stories demanded extensive user involvement and therefor increased the users' participation in analyzing and capturing the requirements. Further user stories were the basis for development of design prototypes.

- The user stories were neither changed, nor updated after they were gathered. The changes were done on the design prototypes. The lifetime of user stories continued until the design prototypes were initiated. Hence the application of user stories in the CDR did not comply with their application according to the agile development and practice (27) (28).

Some of the advantages of user stories mentioned by the solution team are as the following:

- User stories contributed to align and clarify functional areas.
- Knowledge gained through use cases could be supported and verified by the user stories.
- User stories eased delineation of functions and identification of features in the same categories.
- User stories were beneficial to use cases with extra details. In some cases the user stories described special scenarios in the flows (main flow or alternate flow) of the use cases.
- User stories facilitated the understanding of the features to the developers.
- User stories contributed with details in developing the state diagram.

7.1.9 Design prototypes

Prototyping is said to be an effective technique for capturing requirements. By giving a real enough impression of the solution, how it will look like and work, the prototypes help stakeholders to confirm the requirements, suggest improvements and bring up new requirements that might

otherwise be missed or discovered after the product was in use. Prototypes are particularly beneficial in the following situations:

- The product did not exist before and it was difficult to visualize.
- The stakeholders have no experience with either the product or the proposed technology.
- The stakeholders have been using the same system for some time and are used to the way they have been working.
- The stakeholders have difficulties in articulating their requirements.
- The Solution team has difficulties to understand the requirements.
- The feasibility of a requirement is in doubt.

The interactive and realistic simulation provided by prototypes is beneficial to encourage users to explore the solution and try the usability of the features. In contrary prototypes may tempt stakeholders to highly concentrate on the appearance of the solution and forget about the functional requirements and improvements (17), chapter 5.

Through i) the interview concerning the development of user stories and prototypes with the project manager who was involved in these processes, ii) reviews of the presentations made by interaction designer and iii) study of versions of prototypes and their evolution during the time, the following findings can be reported:

- Design prototypes were used to simulate the web-based interface of the CDR. Development of the prototypes started after the interaction designer had interviewed the users and gained enough domain knowledge and understanding of the requirements. User stories were also created before prototyping started (ref. to the dates of creations and confirmations in interviews)

- The prototypes were created and updated in the designer tool, Wireframes, by the interaction designer. The prototypes were made available on the intranet and to the project members. Example of a prototype under construction with comments (the red and green texts for notes and questions respectively) is shown in Figure 42. The approach of developing the design requirements was user-participatory in which the prototypes were developed in close collaboration with the users (ref. to presentations of interaction designer)

- Through the development of interaction design requirements the prototypes were used in communication with the users. Prototypes were frequently updated based on the results of discussions and feedbacks received from the users. When necessary, the developers were contacted by the interaction designer or project manager to ensure the possibility of a technical solution or to suggest alternative solutions (ref. sketches of prototypes, interview with the project manager)

- After the users tried and approved prototypes, they were forwarded to the developers who implemented the related use cases to these prototypes. In some cases there were meetings with the developers for orientations about the choices, functions related to page elements

like buttons (active, inactive), drop-downs, and filters. (ref. interview with the project manager)

- Design prototypes were beneficial to concretize the solution for the users so that they could experience the new layout and features of the system. In addition, recognizing missing features and functionalities were facilitated.(from interview with the system architect)

- As a communication channel, prototypes contributed to easier exchange of information between the solution team and users. To developers, prototypes provided the necessary information and specification to implement related use cases.

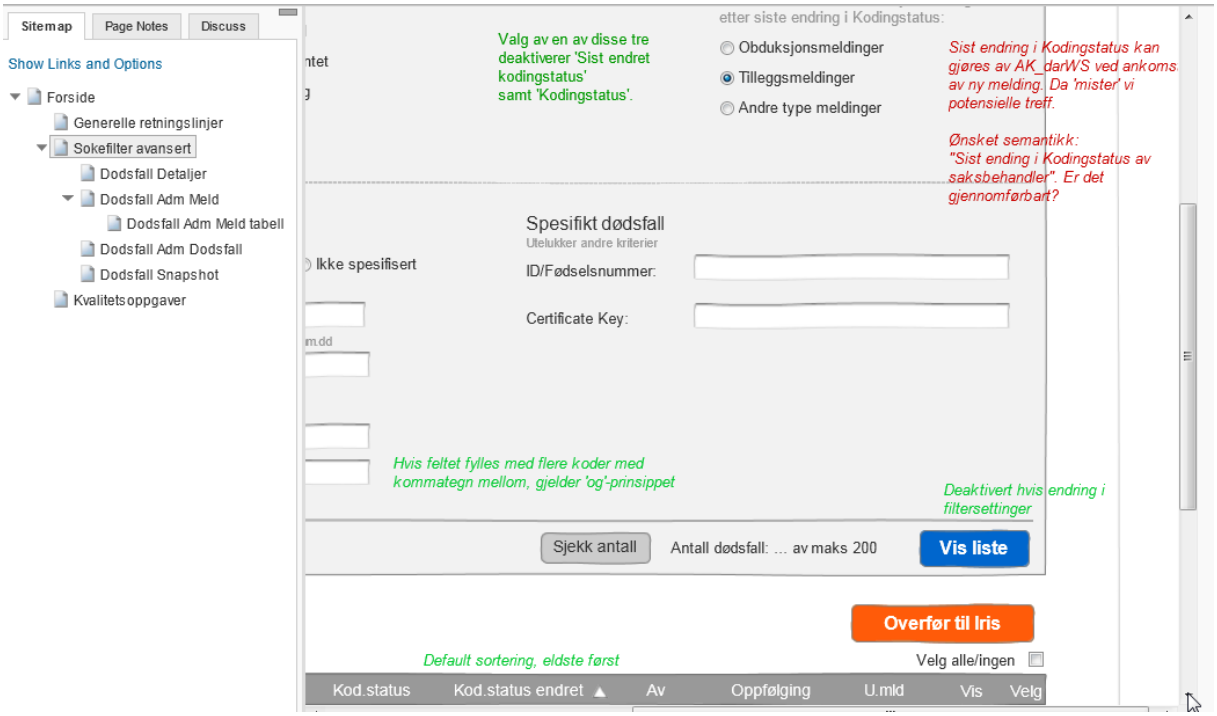

Figure 42 Example of a design prototype (from internal documents presented in Table 4)

7.1.10 State machine diagrams

A state machine diagram (2.6.4) is a graphical representation of the sequence of states of an object, the events that cause a transition from one state to another, and the actions that result from a change in state.

State machine diagrams are especially capable to identify and capture the unique states of an object and the transitions of the object between the states through its life cycle.

State machines are said to be useful for developing real-time or event-driven systems because they represent the dynamic behavior of the system. In addition state machines can be used during all phases of the RE process. State machines can be beneficial in the following situations: (64)

- To model a use case scenario.

- During analysis and design, to model event-driven objects that reacts to events outside an object's context.
- During analysis and design, several state machine diagrams can be developed to show different aspects of the same state machine and its behavior.

The process of development of the state diagrams in the CDR which is outlined below is based on i) the interview concerning the state diagram for death records in the CDR project with the project manager, ii) studying the state diagram and its connections to user stories and use cases in the CDR, iii) the presentation with the scenario walkthroughs (used to capture the details of the states):

- The state machine diagram was used to model the unique states of a death record and the valid transitions between the states through the record's process flow.

- The user story that particularly triggered the need for the state diagram was about the capability of having an updated overview of records in the work pool at any time (6.3.2). Being able to categorize and count the records according to their unique process-statuses - named as coding status - were highly beneficial to users to plan and prioritize which records to work with in their available time (from the interview with the project manager).

- The identification of the distinct coding statues and agreements of the terminology and semantic of the statues were an exhausting process that required a close collaboration between the users and solution team.

- The statuses (or states) were identified through scenario walkthroughs that described the various flows of processing a record and the statues under the different stages of the process (ref. presentation of scenario walkthroughs, Table 5). Afterwards the state diagram was created and elaborated according to the agreed statuses and valid transitions.

- Power point presentations, whiteboard sketches and notes (documented by taking photos) in addition to the domain model and IRIS specific statuses were the artifacts that contributed in the development of the state diagram.

The details obtained from the state diagram were utilized in the development of the functions such as search for records, export to/import from IRIS, administration of records and in message processing (data capture). The details were incorporated in different elements of the use cases such as Description, Main- and Alternate flows.

The state diagram was beneficial in capturing requirements across several use cases. Further the state diagram was capable to model the history of the statuses of a record.

The state diagram is provided in 11.10. The four lanes depict the main categorization of the statuses: 1) Not-finished 2) Under coding 3) On wait and 4) Finished.

In addition to coding statuses, records can be assigned (manually or automatically) in follow-up queues such as Births register, Cancer registry and Medical consultant. The complete list of statuses and follow-ups are shown in Figure 43.

A clip of the state diagram is shown in Figure 44. In order to distinguish the different paths, different colors are used on the transitions arrows. The states and arrows in black depict the main path of the state transition. The states with need of manual processing (coding) are marked by the text on the transitions arrows.

The definition of colors and additional notes are provided in the state diagram (the yellow notes and the legend on Figure 44).

The developed state machine diagram is referred to in the system documentation for the CDR solution and will be maintained in case of future changes.

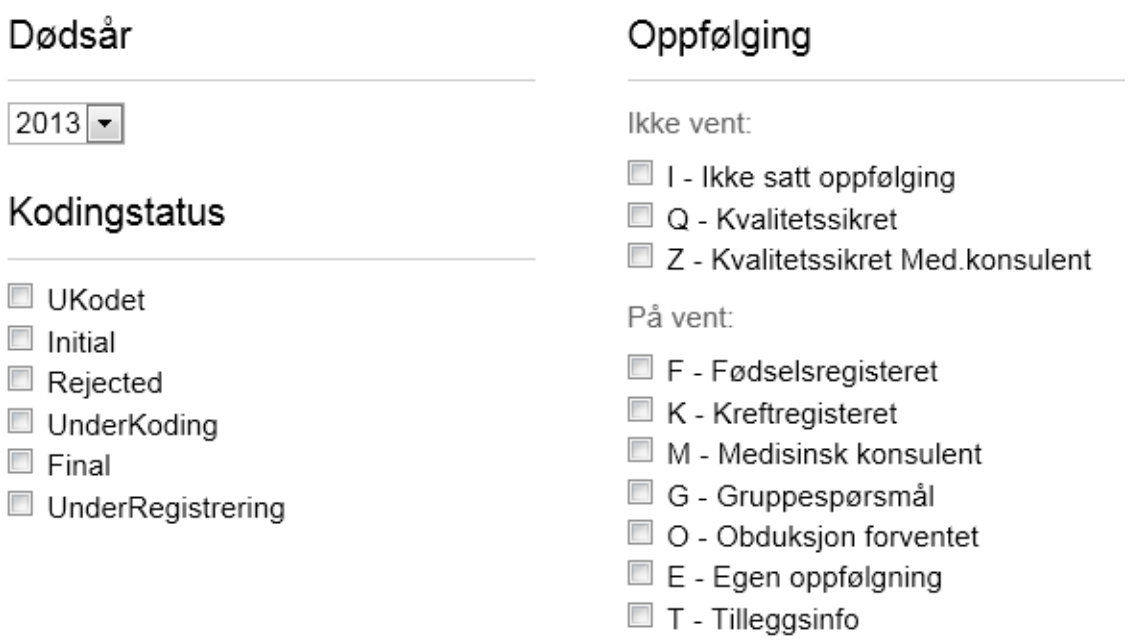

Figure 43 Coding statues and follow-up queues (screenshot from the CDR TEST application)

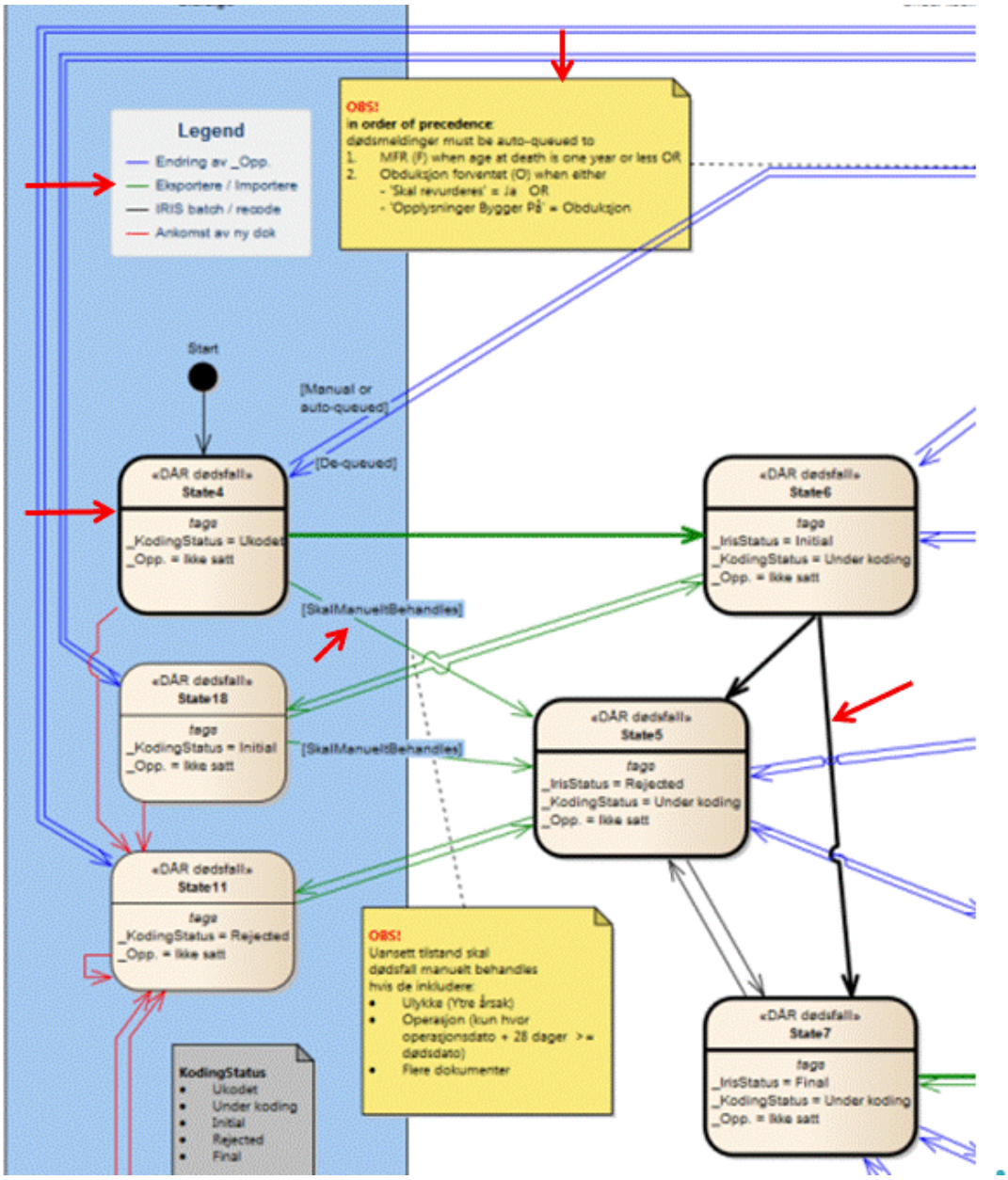

Figure 44 Clip of the CDR state diagram (11.10) (internal documents presented in Table 4)

7.1.11 Other techniques

Usage of Word documents and tabular formats to document the following is considered as another technique to gather, structure and specify the requirements:

- Data contracts for data exchange (incl. specification of respective fields in the external systems and CDR and the translations rules)
- Domain models for description of data fields and their valid values (e.g. the delivery database domain model)
- Information and details related to the requirements written in emails or on sticky notes or stored in the memories of the solution team and other stakeholders were not always

documented and specified to requirements artifacts. However these techniques were used to obtain and store dynamic information under the course of the project.

7.2 Results

Use cases are considered as the main artifacts to represent the functional requirements of the CDR solution. However achieving a complete and correct specification of functional requirements was not possible without application of other techniques (than use cases) and their artifacts.

Various techniques were utilized to contribute to different purposes and in different phases of the RE process.

The requirements artifacts that were developed during the CDR development (mainly the first five development iterations) were studied and investigated to understand how the techniques were applied and what benefits gained through their resulted artifacts. In addition the RE activities under which the techniques were applied were identified. Further the cases where the other techniques contributed to use cases were investigated. The results of the investigations are reported in the following:

- Visualization and direct communication were applied under the course of the project. Visualization in form of diagramming and sketching on paper or whiteboard was used in user meetings, workshops and under discussions between the members of the solution team. In addition, photos of screen, paper and whiteboard were taken to document progressions, decisions, discussions, conclusions and agreements.

- Direct communication was used both in the communication between the solution team and stakeholders, and between the members of the solution team. Direct contact with the users and stakeholders in various meetings, via phone and email was central in exchanging knowledge and capturing requirements and details under the course of the project.

Application of other techniques in the order their artifacts were developed is provided as follows:

- The documents of the existing (old) system were reviewed in the preliminary analysis phase to understand the business domain and needed capabilities. The reviews were beneficial to understand existing workflows and processes and to provide hints of issues and challenges in the existing system.

- Further in the same stage, the high-level workflow was developed to communicate the functional areas and tasks on focus to the stakeholders. As the workflow was task oriented, the scope of the solution and needed functionality were discussed by using the diagram in various meetings with different purposes (e.g. in presentation of time-and activity plans).

- Later in the preliminary analysis phase, the first use case diagrams were developed. The diagrams provided an overview of use cases (needed functionalities). Use cases were also initiated at this stage (6.2.1). Use case diagrams were changed by the use cases (e.g. use cases that were removed or created) and their relationships through the development iterations. Use case diagrams were utilized to communicate the features (or use cases)

planned to be implemented and their implementation priorities to the stakeholders. One use case diagram per functional area was developed.

- Further the high-level sequence diagram (11.11)to model the overall flow of the interactions in the order they were performed and the objects involved in the interactions (e.g. actors, external systems, databases) was developed. The diagram was beneficial to understand and verify the overall flow of interactions with the stakeholders.

- The activities of the development iterations started by analysis of the use cases that were prioritized to be implemented. The techniques and requirements artifacts applied in the development iterations were more specific and related to certain purposes. For instance, the main purpose of the interviews conducted by the interaction designer in the first and second iterations was to understand, analyze and capture the requirements of interaction design.

- The interviews were in addition beneficial in understanding the roles, tasks, processes, challenges and missing features in the existing system. Users' visions and expectations of the new system were also uncovered through these interviews.

- Further the knowledge gained through the interviews was the basis for developing the user stories.

- The template used for creation of user stories had elements to specify the time- and type of the tasks where applied. As such the functions could be easier categorized and a more user-friendly design gained. The user stories were the basis for development of design prototypes. In addition the user stories contributed with extra details to the related use cases and verified the requirements that were already captured by the use cases. The lifetime of user stories continued until the design prototypes were initiated.

- The design prototypes were used to simulate the web-based interface of the CDR. The prototypes were available on the intranet and to the project members. The prototypes were beneficial to concretize the solution, how it would look like and work. As such, recognizing missing features and desired changes were facilitated by the prototypes. In addition the communication between the solution team and users became easier through the prototypes. To developers, prototypes provided the necessary details to achieve the complete implementation of related use cases.

- A sequence diagram to analyze the requirements of IRIS integration was also developed. The details captured by the diagram were later incorporated in the scenarios of the *UC Export to IRIS* and *UC Import from IRIS* use cases. The sequence diagram was beneficial in capturing actions and their order in these use cases. However, details concerning records statuses and their changes by different actions could neither be captured nor specified by this diagram.

- The state machine diagram was developed to capture and specify the unique states (coding statuses) of a record through the record's process flow in the CDR. It was a user story that triggered the need for specification of coding statuses. Categorization of records according to

the coding statuses was highly beneficial to users to plan and prioritize the records to work with in their available time. The identification of the distinct coding statues and agreements of the terminology and semantic of the statues were an exhausting process that required a close collaboration between the users and solution team.

- The statuses (or states) were identified through scenario walkthroughs that described the various flows of processing a record and the statues under the different stages of the process.

- The details obtained from the state diagram were utilized in the development of functions such as search for records, export to/import from IRIS, administration of records and in message processing (Data capture). As such the state diagram was beneficial to capture the requirements across various use cases and functions.

- Usage of Word documents and tabular formats to document domain models and data contracts for data exchange between the CDR and external systems are considered as another technique to gather, structure and specify the requirements.

The techniques (7.1) and the RE activities in which the techniques were applied are summarized in Table 25.

Techniques	Preliminary analysis	Requirements development (elaboration)			
		Req. Analyzing	Req. Capturing	Req. specifying	Solution specifying
Document reviews	X	X			
Use case diagrams	X	X	X		
High-level sequence diagram	X	X			
High-level workflow	X	X			
Visualizations	X	X	X	X	
Interviews	X	X	X		
Direct communication	X	X	X		
Use cases		X	X	X	X
User stories		X	X	X	
Design prototypes		X	X	X	
State diagram		X	X	X	X
Others				X	X

Table 25 Requirements techniques applied during the development of the CDR and their usage in the different activities

8 Discussion

In this chapter, based on the findings presented in chapter five, six and seven and characteristics of the CDR project in chapter three, the problem domain of the study and research questions (1.2) are discussed.

Section 8.1 provides a summary of the common issues of use cases reported by literature. These issues are discussed in the context of the CDR project.

In section 8.2 specific challenges and limitations encountered by use cases in the CDR project are discussed.

In section 8.3 the approach to analyze the content of the use cases is discussed.

In section 8.4 the facts related to the CDR project which might have affected the requirements process and application of the techniques are discussed.

8.1 Use cases challenges

This section describes the most reported issues of use cases in literature. These issues are discussed in the context of the CDR project.

8.1.1 Use cases misuse

Use cases are often misused in that they are used to include and describe other aspects of the requirements than the behavior of the system in interaction with the primary actor. A common example of such misuses is to combine use cases with user interface elements by which use cases are transformed to user manuals rather than to be the requirements documentation (19) (65) (1).

Although inclusion of User Interface (UI) details may be beneficial to add details to use cases, they make use cases longer and harder to maintain. In addition use cases are generally specified by developers or requirements analysts (or engineers) who usually are not expert on UI development (26).

The issue of use cases misuse indicates that there is a need for other solutions to specify UI design requirements related to use cases. Task models are suggested as a complementary technique to be used in addition to use cases to deal with this issue (26).

In the CDR development, the UI requirements related to *UC Export records to IRIS* and *UC Import Records from IRIS* were specified in Description element during the development of use cases (Figure 31). In addition to these UI specifications specified by the developers, the design requirements of the functions were developed by the interaction designer and through design prototypes which users were highly involved in their specifications. The UI requirements related to *UC Search for records* were specified and developed only through design prototypes.

The UI specifications were removed and substituted with reference to interaction design of the CDR intranet (which referred to the current implementation of the interaction design).

The reason to include UI specifications in Export and Import functions were that:

These functionalities constituted the core functions of the CDR; classification of causes of deaths by IRIS. The development of these use cases started in the early phases of the project when the design of the intranet was not completely specified and implemented. In addition these use cases involved limited user interaction and design requirements. Therefore specification of UI in these use cases with respect to the circumstances is understandable.

8.1.2 Use cases relationships

Understanding and modeling the relationships between use cases is challenging. The main purpose of the relationships is to minimize the redundancies between use cases.

There are three types of relations between use cases (26):

1. The *include* relation is the most commonly used relation to specify sub-use cases. If use case B is included in use case A, use case B is invoked by use case A in the step defined in use case A. This relationship should be used to factor out similar behaviors across several use cases.

   Example of such a relation is found in *UC Import records from IRIS* which includes UC *Create snapshot* use case (Figure 21). The second step in the main flow of Import use case refers to creation of snapshots of the records but does not specifically mention the use case name. The relation is neither specified in other elements of Import use case.
   Another example is found in *UC Export records to IRIS* which according to the use case diagram (Figure 21) includes *UC Decrypt identity*. However no reference to this use case could be found in Export use case. The same applies for *UC Request for additional documents*.

2. The *extend* relation indicates that the extending (base) use case invokes the extended use case if the corresponding extension point has been reached within the base use case and the extension condition is fulfilled. After completion of the extension use case, the base use case continues.

   Example of such a relation is found in *UC Search for records* which is extended by *UC Search with identity* (Figure 21). This relation is mentioned in Description element of Search use case however there is no reference to the extended use case in the Main flow. The conditions that should be satisfied to invoke the extended use case are neither specified.

3. The inheritance relation is used to model generalization and specification between use cases. This relation is said to be the least used and specified relation and that the usage of this relation leads to more complicated specifications in practice. The UML definition only states that “A generalization from use case C to use case D indicates that C is a specialization of D”. One possible interpretation of this may be as follows: The inherited use case serves as a template for all its inheriting use cases. In this vein, the inheriting use case replaces one or more of the courses of action of the inherited use case.

   An example of such a relation was found in Data capture use cases where the six use cases were specialized use cases of the generic use case *UC Process SSB Message*. The generic use case includes the common parts such as Description and Exceptional flow to which the specialized use cases referred. As such the usage of this relation has reduced the redundancies.

Literature discusses that the usage of relationships can improve and ease the communication and comprehension of use cases. However lack of well-defined and agreed semantics of use case concepts and relationships may cause ambiguity and misinterpretation. Thus it is recommended to put effort on writing use cases rather than to organize use cases in relationships (26) (19).

8.1.3 Use cases in requirements management

Based on the results of an evaluation of techniques for documenting user requirements, use cases are incapable or poorly capable to support the following aspects of requirements management (1):

- Relationship and type of relationships between requirements
  Requirements of a product or a solution are normally related to each other. Understanding the relationships between the requirements is critical to plan releases, manage changes and reuse. In addition the type of relationship should be identified to manage the consequences of changes.
- Represent types of requirements
  Use cases are mainly used to model functional requirements and are not capable of representing other types of requirements such as non-functional.
- Priority between requirements
  Prioritizing requirements is an important activity in RE. The purpose is to provide an indication of the order in which requirements should be developed.
- Traceability between requirements
  A requirement is traceable if its origin can be identified and if it can be referred to in future development. Traceability of a requirement is especially important when the software product enters the operation and maintenance phase.

Use cases were not utilized to address the abovementioned aspects of the requirements management in the CDR development. Categorization of use cases by functional areas (Figure 19) and use case diagrams for each functional were mainly used to provide an overview of the requirements and their relations. Use cases related to core functionalities of the CDR such as data capture and IRIS integration for classification of causes of deaths were prioritized.

Non-functional requirements such as security, performance, testability and reliability were documented based on a specific template (template for application's quality found in wiki pages of development unit) to document different aspects of the application's quality.

- Verifiability
  A requirement is verifiable if there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement (1).

Due to the lack of semantics, use cases are difficult to be analyzed. In addition rules that can guide inspection of use cases are not well studied and developed (5). The textual character of use cases and impreciseness and ambiguities that follow in use cases text, make the verification of completeness and correctness of use cases to a difficult task (10) (1).

In the CDR project, use cases were reviewed by the developers and project manager to be controlled 1) for correctness in that they matched and specified the actual implementation and were in accordance with the requirements and 2) for completeness meaning that they sufficiently specified the technical solution of the functional requirements. These reviews were conducted manually.

8.1.4 Use cases elements

The following challenges related to use cases elements are reported (66):

- Pre-conditions: use cases normally do not state a solution for when pre-conditions are not satisfied.

  This issue can be seen in *UC Fill Delivery Database* and *UC Synchronize ICD Codes*, where no solution for unsatisfied pre-conditions is described.

- Post-conditions: a user-centric description of post-conditions does not include the post-conditions that are not a part of the user goal.

  This issue was not observed in CDR use cases. In addition several of the studied use cases had no human primary actor to be user-centric.

- Main- and Alternate flow: definition of the conditions under which the alternate flows should be run is usually not included in use cases. The same applies for extension points and conditions that should be satisfied to invoke extended use cases.

  A variant of this issue was observed in the CDR use cases, discussed in 8.2.2.3.

- Exceptions: Information about how to detect exceptions and what to do in each case are usually not sufficiently identified and specified.

  Exceptional flows were almost not specified in the CDR use cases, discussed in 8.2.2.3.

8.1.5 Other issues

Other commonly reported issues of use cases are:

- Use cases focus on the system-user interactions. As such the context of the problem and existing constraints beyond the system boundaries are not captured by use cases (67) (68) (69).

  This issue was observed in specification of the search feature, discussed in 8.2.1.1.

- Use cases are less suitable to specify requirements concerning data warehouse, batch processes or time-triggered functions, hardware products with embedded control software and complex mathematical algorithms. These types of requirements can normally not be captured through user-system interactions. Use cases are neither capable of capturing complex business rules and government regulations (70).

This issue was observed in the data warehouse use case which is discussed in 8.2.1.2.

- Finding a balanced and suitable level of details in use cases is challenging. Too-detailed use cases are lengthy and difficult to follow and understand and too-high-level use cases are often not valuable (67).

  This issue, which is discussed in detail in 8.2.2, was observed and identified by the changes analysis of use cases (chapter five).

- Use cases are generally not useful to facilitate the communication of requirements with stakeholders. Due to use cases elements which mostly address the development aspects of the requirements, discussion of use cases content with non-technical stakeholders is not beneficial (1).

  This could be observed in the application of use cases in the CDR project, discussed in 8.4.1.

8.2 Use cases challenges in the CDR development

In this section, use cases challenges with respect to the types of requirements (8.2.1), use cases elements (8.2.2) and the use case template (8.2.3) in the CDR development are discussed.

8.2.1 Type of requirements

According to the results outlined in chapter six, use cases were poorly capable of analyzing and capturing requirements concerning the search function and data warehouse solution. In addition use cases could not provide a complete specification of these solutions. The limitations and challenges of use cases are discussed in the next sections.

8.2.1.1 Search function

Use cases were less capable of analyzing and capturing the requirements of the search feature. This can be related to the type of the requirement which mainly concerned user interaction and design.

The challenging part of the search function was to identify the search criteria. The requirements of the search originated from a user story (6.3.2) that described the need of overview of the existing records categorized by their statuses and follow-up queues in the work pool. Dealing with this requirement made it clear that there was a need for identification of the unique coding statuses and follow-up queues to be used for the search criteria.

User stories in this relation not only facilitated capturing the details of the search function but also addressed the essential need of identification of coding statuses.

Knowledge of coding statuses was further used in the functions that caused changes in the coding statuses of the records. Import from- and export to- IRIS and Data capture were as such. This complies with the challenge reported by literature and that use cases are not suitable to capture the context of the problem and the requirements that exist across different features (8.1.5).

Further use cases were incapable to specify and document the coding statuses. The state machine diagram was developed to completely identify and specify the unique coding statuses and valid transitions between the statuses.

Design prototypes were beneficial in capturing the requirements concerning the simple and advanced searches and the arrangement of their criteria. In addition the display of the result fields, their order and which features to be available for the results were understood and captured by the design prototypes.

8.2.1.2 Data warehouse

Specification of information requirements (6.3.3.2) is critical in the development of data warehouse solutions. Use cases were poorly capable of capturing and specifying these requirements. Use cases are structured to describe user-system interactions and therefor don not have the right constructs to capture other types of requirements.

Information requirements of the data warehouse solution of the CDR were specified in separate Word documents and in tabular formats.

Usage of the new element, Invariant, in two ETL use cases (6.3.3.2) was also an attempt to adjust use cases to gather and specify the components of the ETL solutions. This indicates that the same use case template may not be suitable to capture and specify various types of requirements.

Different use case templates such as fully dressed and casual suggested by Cockburn (19) are examples of templates used for different purposes.

8.2.2 Use case elements

Description and Main flow elements of use cases were changed more frequently than the other elements of use cases while Exceptional flow and Alternate flow were lest specified (chapter five). These elements are discussed in the following sections.

8.2.2.1 Description element

The Description element generally contained various specifications (e.g. IRIS integration specification) and descriptions of the technical solutions specified by use cases (e.g. search function). The specifications that could not be categorized to any other elements of use cases were also placed under this element (e.g. specification of cases where pre-condition was not true in *UC Import records from IRIS*).

The length of Description varied from three to four lines (e.g. *UC Synchronize ICD codes*) to three pages (*UC Export records to IRIS*).

Further, information related to Alternate flows (e.g. the rules for pre-rejections which can be considered as the conditions of alternate flows in *UC Export records to IRIS*) and Exceptional flows (description of the second-time export of records in *UC Export records to IRIS*) were found in Description element (11.3.2).

Usage of Description in including the specification of the technical solution can be considered as beneficial to provide a complete specification of the solution. However redundancy of information

specified in other elements of use cases may cause confusion and inconsistency in use cases in case of mismatches between the specifications.

The level of detail and what specification to contain, was dependent on the type of requirements and the individual developer's decision. As such the uniformity that use cases provide is limited to the elements and high-level structure defined by the use case template and not in the content and structure of the individual elements.

8.2.2.2 Main flow

The changes in the Main flow element of the majority of the studied use cases (Data capture, Data processing (except for search)) were caused by changes in implementation of functions, changes in scope of the functions or changes in the requirements (6.3 , 6.4).

8.2.2.3 Least specified elements

The majority of use cases had no Exceptional flow. Data capture use cases were the only use cases which referred to the Exceptional flow specified in the generic use case *UC Process SSB message*.

The immediate conclusion is that developers were agreed on non-necessity of specification of this element in the use cases as it was handled in the implementation. Another reason could be to avoid lengthy and complex use cases.

Alternate flow was the second least specified element in all of the studied use cases.

In some of the use cases such as *UC search for records* and *UC Request for additional documents* use cases, Alternate flow was not found due to the type of function. In *UC Export records to IRIS* use case, the criteria for Alternate flow were specified but there was no connection between these criteria and Main flow and in which step these criteria would be checked and how the alternate flow was handled. In *UC Import records from IRIS* use case, the criteria of Alternate flow were specified in Description (the records that should not be imported back).

In Data capture use cases, Alternate flow was removed due to the changes in requirements (handling duplicate messages).

8.2.3 Use case template

The same use case template was applied to construct all of the use cases.

In some of the ETL use cases, the need for description of the invariable part of the solution, led to the usage of a new element (Invariant), created by the responsible developer. This addresses the need of adjustment of the template for different types of requirements.

Lengthy and unstructured narrations in the Description element might also be an indication that different templates could suit different types of requirements better. In addition to provide more uniformity in the content of use cases elements, a more detailed guideline on which specification to contain in this element could be helpful.

8.3 Analysis of the use cases complexity level

In an attempt to analyze the use cases by their content and estimate their complexity, I tried to investigate whether the complexity level of use cases might have affected how the use cases

technique was applied. In addition the relation of the complexity to challenges of use cases and the choice of other techniques were examined.

The complexity of a use case was estimated by considering the total number of steps in the Main- and Alternate flows as specified in the use cases. The complexity of a use case is said to be dependent on the complexity of the functionality and its logic described by the use case. The approach to estimate the complexity was inspired by the estimation conducted in (10). The results of the estimations varied from 2 to 26, the lower and higher limits respectively. Estimations with values lower than 14 were considered as middle. The results of the estimations are summarized in Table 26.

Functional area	Complexity level
Data capture	4-9 (low-middle)
Data processing	3-26 (low-high)
ETL	2-12 (low-middle)

Table 26 Estimations of the complexity level of use cases

Considering the results of the estimations, I investigated the content of Main- and Alternate flows and how they were structured to examine the validity of the results. In addition I asked the solution team about the use case with the highest estimated complexity level (*UC Register Death Record*) and whether the implementation of the use case was affected by the complexity. The answers did not comply with my results and developers did not consider the use case as complex. However the estimations in some other use cases, e.g. from Data capture area were realistic and correct.

After closer investigation of the content and structures of Main- and Alternate flows and the analysis approach in (10), I realized that this type of estimation may not be appropriate for the studies where the uniformity in application of use cases and in the structure of use cases element (particularly Main- and Alternate flow) were not evaluated beforehand. Therefore the results of the estimations are not used to argue for challenges or use of other techniques in this study.

8.4 Facts related to the CDR project

Some of the facts related to the circumstances of the project that might have impacts on the RE process and application of various techniques are discussed in the following sections.

8.4.1 General application of use cases

Use cases – the textual descriptions - were not utilized in communication with the stakeholders. Specification of elements of use cases was neither discussed with the users (or stakeholders). Use case diagrams however were used to communicate 1) the functional areas, 2) use cases that were planned to be implemented and 3) their implementation priorities to the stakeholders. In addition the boundaries of the functions (or features) were illustrated and discussed on these diagrams.

Use cases were specified and implemented by the same developer. This fact has certainly affected the frequency of use cases updates and the content of the elements of use cases. Use cases might have been updated more often if different developers were involved in the development or implementation.

Further the fact that use cases were specified by developers was related to the type of the requirements (technical use cases, category one and three, 6.3.1 and 6.3.3). In the use cases that

higher user interaction was involved (category two, 6.3.2) the requirements were captured by user stories and design prototypes by the interaction designer. Requirements that touched several use cases (the coding statuses of the records) were captured and specified by the state diagram by the project manager.

8.4.2 Number of primary users

The few numbers of primary users (four) was advantageous in communication with the users and that the responsibility areas and contact persons were limited to these users. In addition the conflicts and disagreements were limited.

Despite the advantages, being dependent on these few users who could contribute to the requirements tasks was considered as a risk. The available time of these users was also limited which in some cases resulted in waiting times for the solution team.

8.4.3 Stakeholders conflict

There were little (no) conflicts amongst the different stakeholders concerning CDR requirements since the majority of the stakeholders (named in section 3.7) were not involved in the requirements process. This condition was due to the fact that existing contracts and solutions were not object of changes in the new CDR.

8.4.4 Constraints

The available time for implementing the new CDR and transferring the registry administration and employees to the NIPH was limited and could not be extended (see section 3.2). The time constraint had impacts on priorities and decisions on the scope of the solution as well as the time used to specify requirements. According to the solution team, the choice of the techniques was not affected by the time pressure. However the amount of time to specify requirements influenced the application of techniques.

8.4.5 Previous knowledge and experiences

Experiences from development of other health registers (such as Vaccination register) conducted by the solution team and the general knowledge of requirements of health register were beneficial to more effective identifying of needed features in the CDR. In addition the ability to suggest possible solutions through existing solutions developed for similar requirements was increased.

The positive effect of previous experiences can be seen in Figure 26 where the functional areas and many of the use cases were identified in the early stages of the project.

"…. (Previous experiences) *was of very great value, especially with respect to increasing efficiency of the team's work. However, it may have kept us from thinking outside-the-box in certain areas."* (from email exchange with the project manager)

"…. (Previous experiences) *had a big role, especially concerning non-functional requirements but also for understanding the requirements related to standards, codes, data delivery, data warehouse, statistics, and interaction with the national population register. However, whether these experiences had only positive effects can be discussed"* (from email exchange with the system architect)

9 Validity

The validity of a study denotes the trustworthiness of the results, to what extent the results are true and not biased by the researchers' subjective point of view. Although the validity of the analysis and results cannot be assessed before these phases are completed, the validity should be considered in all of the phases of a case study (e.g. in data collection and research design) in order to maintain the validity of the study (49).

The four commonly used evaluation aspects and the recommended case study tactics are shown in Figure 45 (51). The four validity evaluations and how they are addressed in the study are discussed in the following sections.

9.1 Construct validity

This aspect of validity concerns the correctness of operational measures for the concepts being studied (51) and should be mainly considered during data collection.

The recommended tactics to this validity (Figure 45) are followed in the study by collecting data from various sources such as document reviews, interviews, literature reviews and passive participation or observations.

A part of data analysis in this study refers to changes in the use cases during the first five development iterations (chapter five). The changes history of use cases was recorded by manually saving snapshots or using the available versions of use cases in the source code management system. The number of snapshots and versions varied in different use cases. Some of the updates may have not been captured by the manual snapshots dependent on the time the snapshots were taken. The number of available versions of use cases was dependent on how often the developers had updated the use cases in the source code management. However these had only effect on the number of changes and not on the changes.

Interpretation of changes and what changes were related to (e.g. distinguishing between corrections, precisions, addition of details) were also challenging. In addition determining if the changes were related to changes in the requirements was not straightforward and required several reviews of the use cases and changes in addition to review of the interviews with developers.

To address these difficulties, the draft of the thesis was sent to the interviewees (who were also members of the solution team) to be reviewed and confirmed for the validity of the information that referred to interviews, documents and facts of the project.

9.2 Internal validity

This aspect of validity is more important for explanatory or studies that concern casual relations between the studied objects (51).

This study concerns to explore and describe application of different techniques for specifying functional requirements with particular focus on use cases and challenges encountered by using them. However to maintain the internal validity, I have tried to explain the chain of evidences (by following the RE activities) and collecting data through the development of the project to derive the results.

9.3 External validity

This aspect of validity is concerned with the extent to which it is possible to generalize the findings, and to what extent the findings are of relevance for other cases (49).

Application of use cases and other techniques to specify requirements varies in different projects and organizations dependent on the project's circumstances such as size and type of the project (small, distributed, stakeholders), type of the product (new, legacy, complex) (19) and factors such as budget, available resources and time and legal constraints (e.g. how detailed and complete the specification should be). In addition the development process has impacts on the requirements process, approaches and techniques to specify requirements. As such generalization of application of use cases and how they are developed is difficult.

The findings of this study can contribute to more effective requirement processes and selection of techniques (with regard to the type of requirements) to development projects with the same type of requirements. The challenges identified in application of use cases are also general and can be helpful to other projects that apply use cases.

Usage of other techniques and how they were beneficial to specify requirements are quite general and commonly accepted and can contribute to other projects to select alternative techniques to specify requirements.

The cases study tactic to address this validity concerns use of theory in single-case studies (Figure 45). As there are few studies conducted in evaluation of the use cases technique for specifying requirements, this tactic could not be applied. However by extensive literature searches under the course of the thesis, I tried to find similar studies, systematic literature reviews, best practices and guidelines to support the findings of the study.

9.4 Reliability

This aspect concerns to what extent the data and analysis are dependent on the specific researchers and if the same results can be derived if the study is conducted by another researcher and with the same collected data (49).

The database of this study contains the following:

- Projects documents that are studied and referred to (Table 4, Table 5)
- Transcripts of all the interviews

- Snapshots and versions of use cases
- Other requirement artifacts that are referred to
- Description of the analysis approach
- Literatures that are referred to

As such I consider the results of the analyses as reproducible by another researcher. However the results of the analyses of use cases changes can differ dependent on how the changes are interpreted and defined.

TESTS	Case Study Tactic	Phase of research in which tactic occurs
Construct validity	♦ use multiple sources of evidence ♦ establish chain of evidence ♦ have key informants review draft case study report	data collection data collection composition
Internal validity	♦ do pattern matching ♦ do explanation building ♦ address rival explanations ♦ use logic models	data analysis data analysis data analysis data analysis
External validity	♦ use theory in single-case studies ♦ use replication logic in multiple-case studies	research design research design
Reliability	♦ use case study protocol ♦ develop case study database	data collection data collection

Figure 45 Case study tactics for four design tests (51)

9.5 Other factors

Other factors that might constitute threats to validity are described in the following:

9.5.1 Literature

Even though literature search and review of relevant studies were among the major activities under the course of the thesis, important articles might be overseen. This can be related to the usage of various terms in the field of requirements engineering and the overlaps between the terms. Analyzing, eliciting, gathering, capturing, describing, specifying and documenting are used interchangeably in literature although they may refer to different activities and techniques.

9.5.2 Case of the study

The main reason to choose the CDR project was the availability and accessibility to the projects members and data. This might constitute a threat since other projects could have provided other types of requirements artifacts that could answer the research questions in another (better) way. In addition the threats mentioned in 4.5 apply to the case of study.

9.5.3 Document reviews

Important documents might be overseen. The analysis of documents was based on my understanding and interpretations which could be biased (also mentioned in 4.5).

9.5.4 Interviews

I might have missed important points during the interviews or misunderstand the interviewees. However by the confirmations received from the interviewees (through reviewing the thesis) this threat is reduced.

9.5.5 Participation

I could have participated more in the project activities, however the meeting's agenda and the details discussed were usually beyond the details I needed and the available time for doing the thesis.

10 Conclusions

Understanding the application of use cases and challenges encountered by their application were the main concerns of this study.

To understand use cases application in practice, a case study on investigation of use cases in the development project for modernization of the Cause of Death Registry (CDR) was conducted. The study lasted for just over one year where the application of use cases through different activities of the RE process was followed. In addition the evolution of use cases through their lifecycles was studied. As addressed by literature, very few empirical real-life studies concerning use cases and their issues were conducted.

The results of the investigation answered the two research questions of the study:

RQ1: How was the use cases technique applied in different activities of the RE process in the CDR project and which challenges were encountered by use cases?

RQ2: How were techniques other than use cases beneficial to specify functional requirements? How were the limitations and challenges of use cases addressed by these techniques?

Further, use cases and other applied techniques were evaluated in their capability of analysing, capturing and specifying the requirements.

Investigation of use cases and their lifecycle showed that use cases had different forms and were useful for different purposes:

- Before and during the initial implementations, the main purpose of use cases was to describe the requirements, missing information, issues of the development, facts and constraints to consider and possible solutions. Use cases were used as drafts by the developers to gather and store the different aspects of the requirements at this stage.
- During the implementation, use cases served as logs of implementation and technical notes with focus on the technical solution, issues faced through the implementation and reminders of to-dos.
- By the end of implementation, use cases were reviewed and refined to contain descriptions of the technical solutions for functional requirements as implemented in the final solution.

Study and analysis of use cases changes history showed the following patterns in use cases application and development:

- Use cases were specified by developers and for development purposes. Use cases were not used to communicate requirements with the stakeholders.
- All of the use cases were structured based on the same use case template.
- The majority of use cases had no Exceptional flow. Use cases that had Exceptional flow were specified by the same developer.
- Use cases that were specified by the same developer, were more similar in the use cases elements that were specified and extent of specifications and details.
- Changes in requirements, scope of functions and implementation were the main reasons for changes in Post-conditions and Main flow elements of use cases.

- Description element of all of the studied use cases was mainly used to 1) specify the technical solution of the functionalities and 2) include parts of requirements that could not be categorized to any other elements. Further, in few use cases, specifications related to other elements such as Alternate flows and conditions that invoked Alternate flows were included in Description element.
- The level of detail and specifications of the use cases elements, varied with the type of requirements and the individual developer's decision. This was apparently observed in the variation of the specifications and extent of details given in Description element. Further the new use case element, Invariant, used by one of the developers showed that the content and structure of use cases were dependent on the developers.

Based on the use cases application mentioned above, the main challenges encountered by use cases in specifying functional requirements were identified:

- Determination of the content and extent of specifications of use cases elements were challenging. This challenge was mainly observed in Description and Main flow elements.

  In literature this challenge is addressed through finding the proper level of details with respect to among the others, the type of the project, stakeholders and final product. In addition the purpose of the requirements specification (e.g. as a contract between the customers and solution team, for estimations of time, resources and activities or for development activities) are mentioned as important factors to consider when specifying requirements.

- Not all of the necessary specifications of the requirements and technical solutions could be categorized in the use cases elements defined in the CDR use case template. Usage of Description element to describe various components of the solution, and situations where pre-conditions of use cases were not true, are such examples. Usage of the new use case element, Invariant, by one of the developers is another example.

- Use cases did not state their relationships with other use cases. In one case, the *extend* relation was mentioned in Description element but there was no reference to the extended use case in the main flow. The conditions to invoke the extended use case were neither specified in the use case.

To address the challenges mentioned above, I suggest:

- To discuss and agree the necessary and practical level of detail between the stakeholders involved in specifying requirements. The sufficient level of specification to developers and future maintenance should be considered as important factors in the decision of the level of detail. Further, the circumstances of the project such as size of the project, available time and resources, skills of RE techniques and the solution team should be taken to account when deciding techniques and level of detail in requirements specification.
- To use different use case templates with adjusted elements with respect to the type of requirements. Application of other recommended techniques and best practices to specify particular types of requirements can also be considered.

- Not use Description element to specify other elements of use cases. Redundancy of information leads to inconsistency in requirements specifications and difficulties in their maintenance. The parts of requirements that cannot be categorized in use cases elements (e.g. the situations where pre-conditions are not true) should be addressed in guidelines for how and where to be specified.
- To describe specifically how to refer to related use cases (e.g. extended use cases, included use cases) and the conditions that invoke related use cases in the use cases guideline. Another solution to address the invisibility of related use cases could be to include use cases diagrams as a part of the functional description in the system documentation.

In spite of these challenges identified through the CDR project, the project's circumstances such as 1) few numbers of developers who specified and developed the same use cases, 2) few numbers of stakeholders, 3) no conflict in requirements of different stakeholders, 4) the usage of use cases to development activities and for developers and 5) the time pressure which forced the project to this way of work[23] made these challenges non-problematic in this project.

Not having these conditions, e.g. in larger projects with distributed stakeholders, with a larger solution team, and conflicts in requirements and decisions, could turn these challenges to become problematic.

Moreover, the fact that decisions were made by few stakeholders in the CDR project, reduced the need for registration of every detail in use cases and their frequency of update. In this way, the time was saved and lengthy and complex use cases were avoided. These benefits however can make some difficulties and confusions in understanding of use cases, design choices and decisions for people outside the project and without access to internal knowledge.

Investigation of other requirements artifacts than use cases showed that various techniques were applied to different purposes and in different stages of the RE process. The state diagram, for instance, was particularly beneficial to capture and specify the requirements across different features while user stories and design prototypes were mainly used to capture and specify interaction design requirements. The development of the state diagram took place after the user stories flagged the need for the state diagram.

Diagram models such as use case-, sequence-and state diagrams, in addition to general visualizations were used to communicate requirements with the stakeholders. The preference of visualized artifacts underlines that they were easier to be used to discuss and capture details of requirements with stakeholders.

The main lessons learned from this study regarding to application of use cases and other techniques are:

- Utilizing multiple techniques with respect to their capabilities and type of requirements is necessary to achieve a complete and correct specification of the requirements.

[23] By working closely in a little team and having dynamic information and details in memory

- More detailed guidelines on specification of use cases elements and the requirements that cannot be categorized to use cases elements can be beneficial to consistency and completeness of use cases. Different use case templates adjusted to the type of requirements can be developed and used. Further discussions and agreements on the level of detail, mandatory use cases elements, or the minimum specification of the elements to be sufficient to future maintenance and the system documentation could be beneficial to address the challenges concerning the content of the elements and to streamline the application of use cases.
- Categorization of functional requirements with respect to their types could contribute to the choice of proper techniques. In addition, application of recommended techniques for certain types of requirements such as data warehouse could be considered.

The results of this study can be used to create a high-level guideline for deciding on relevant techniques to be used on different types of requirements and requirements activities.

However, the development of more detailed guidelines would require a deeper understanding of the relationships between the various techniques and how they can contribute to each other, necessitating further investigations and empirical knowledge gained from real-life projects.

11 Appendix

11.1 Use case template in the CDR project

Id	[En unik identifikator for use caset. Et tips er å prefikse med ”UC” og bruke use case navnet i Pascal Casing, slik som for eksempel ”UCRegistrereNyKunde”, istedenfor tilfeldige løpenummer.]
Navn	[Bruk aktive verb og fokuser på målet til brukeren, slik som ”Registrere ny kunde”, istedenfor ”Kunderegistrering”.]
Trigger	[Hendelsen som starter use caset. Ofte initiert av brukerinteraksjon, men kan også initieres av et sett av betingelser som blir oppfylt.]
Beskrivelse	[Beskrivelse av use caset. Her er det også naturlig å beskrive forretningsregler og valideringsregler, så fremt disse er relatert kun til det aktuelle use caset. Lag et eget dokument for forretningsregler hvis disse går på tvers av flere use cases og referer til dokumentet fra use case beskrivelsen.]
Pre-conditions	[Betingelser som må være oppfylt for at use caset skal kunne utføres. Unngå bruk av generelle eller opplagte betingelser, slik som ”Systemet må være oppe”, ”Databasekoblingen må være operativ”.]
Post-conditions	[Betingelser som er oppfylt når use caset avsluttes. Typisk forandring i tilstand, som for eksempel at det er blitt lagret en eller flere rader i databasen.]
Main flow	[En liste av hendelser som utføres i en normal gjennomgang av use caset. Fokusere på aksjonene til aktøren som utfører use caset, og hvordan systemet responderer til disse aksjonene.]
Alternate flow	[En liste med av varierende/alternative hendelser til den normale flyten.]
Exceptional flow	[En liste med avvikende hendelsen til den normale flyten. Fokuser på avvikende hendelser som er viktig at systemet kjenner til og kan håndtere. Typisk hvordan håndtere feilsituasjoner.]
Issues/Notes	[Ustrukturert tekst som ikke passer under noen av de andre attributtene. Brukes typisk som et todo/kladdefelt i analysefasen.]

11.2 Example of an interview guide

Example of an interview guide (created by the author) used in interviews

Duration: 1 hour

Preparation: background information related to the interview questions are studied by the interviewer, the meeting location for the interview is booked, the interviewee is informed about the purpose of the interview and possible topics for discussions.

Interview guide

- Brief presentation of the master thesis topic and problem domain, how it is planned to conduct the case study and collect data, the purpose of the interview and what data we intend to collect over the interview
- Presentation of interviewee, academic background and field of work, experiences, the role in the project
- Description of the RE activities, techniques and background for the choice of the techniques (is it dependent of the development process? Any previous experiences?)
- Techniques for specifying requirements
    - Templates, guidelines in specifying and documenting requirements
    - How to track changes in the requirements documents? Versions? Changes? Tools?
    - How/who keep track of discussions and changes that are suggested in the meetings with users?
    - Review meetings? How often?
    - Quality check of use cases
    - Type of requirements (functional / non-functional)
    - Any thoughts about non-functional requirements and their impact when defining functional RE?
    - Test plans
- Project background
    - Issues with the existing system and motivations for the new system
    - Some of the main goals and advantages of the new system (usability, quality, cost reduction, maintainability,…)? Plans t to measure/assess achievement goals through the course of the project?
    - Introduction of stakeholders, is there any priority on user groups? Any major Differences?
    - Constraints (restriction on how the system should be designed, technology, software, hardware, time, budget,etc)
    - Project issues (conditions under which the project will be carried out like lacking resources, lacking knowledge, etc)
    - Challenges
    - Risks

11.3 Examples of CDR use cases

11.3.1 UC Import records from IRIS (from 01.09.2013)

<table>
<tr><td>Id</td><td>UCImportereDodsfallFraIris</td></tr>
<tr><td>Navn</td><td>Importere dødsfall fra Iris</td></tr>
<tr><td>Trigger</td><td>DÅR saksbehandler velger å importere en lot av dødsfall fra Iris.</td></tr>
<tr><td>Beskrivelse</td><td>
Brukstilfellet omfatter tilbakeføring (import) av et dødsfall fra Iris, klassifikasjonssystemet for dødsfall. En lot fra Iris (består av to tabeller, MedCod og Ident) tilsvarer et dødsfall i Dår koblet med ID-feltet CertificateKey. Når en lot importeres oppdateres dødsfallet med data og status fra Iris, og loten slettes fra Iris.

Følgende dødsfall skal ikke tilbakeføres fra Iris:
• Dødsfall som er ukodet (initial) i Iris, men har fått oppfølging «K» eller «Z»
• Dødsfall som er rejected i Iris, men har fått oppfølging «K» eller «Z»

GUI

For valg for tilbakeføring av ett eller flere dødsfall se interaksjonsdesign.

<table>
<tr><th>Iris</th><th>Dår</th><th>Regel</th></tr>
<tr><td colspan="3">xxxxIdent</td></tr>
<tr><td>Ident.CertificateKey</td><td>Dodsfall.Uuid</td><td>Parses fra base64 til Guid og matches med dødsfall i Dår</td></tr>
<tr><td>Ident.MannerOfDeath</td><td>Dodsfall.YtreArsak</td><td>Settes til null dersom MannerOfDeath = 1.</td></tr>
<tr><td>Ident.AcmeCodes</td><td>Dodsfall.Klassifisering.AcmeCodes</td><td></td></tr>
<tr><td>Ident.ErnCodes</td><td>Dodsfall.Klassifisering.ErnCodes</td><td></td></tr>
<tr><td>Ident.SelectedCodes</td><td>Dodsfall.Klassifisering.SelectedCodes

Dodsfall.Klassifisering.Arsak[x].

VarighetTekstStandardisert

Dodsfall.Klassifisering.Arsak[x].

Arsaksdiagnose.Diagnose

Dodsfall.Klassifisering.Arsak[x].

Arsaksdiagnose.indeks</td><td>Feltet fra Iris er formatert slik at vi kan plukke ut diagnosene med tilhørende linje og indeks.</td></tr>
<tr><td>Ident.CodingVersion</td><td>Dodsfall.Klassifisering.SystemVersjon</td><td></td></tr>
<tr><td>Ident.UCCode</td><td>Dodsfall.Klassifisering.
UnderliggendeDiagnose.Kode</td><td>Opprettes i Dår dersom den ikke</td></tr>
</table>
</td></tr>
</table>

			finnes
	Ident.SubstitutedCodes	Dodsfall.Klassifisering.SubstitutedCodes	
	Ident.Status	Dodsfall.Saksbehandling.Kodingstatus	
	xxxxMedcod		
	MedCod.CertificateKey	Dodsfall.Uuid	Parses fra base64 til Guid og matches med dødsfall i Dår
	MedCod.LineNb	Dodsfall.Klassifisering.Arsak[x]	Brukes til å mappe en enkelt MedCod i Iris til Arsak i Dår. 0 = ArsakA, 1 = ArsakB, 2 = ArsakC, 3 = ArsakD, 4 = ArsakE, 5 = Arsak Medvirkende
	MedCod.TextLine	Dodsfall.Klassifisering.Arsak[x].Tekst	
	MedCod.CodeLine	Dodsfall.Klassifisering.Arsak[x].CodeLine	
	MedCod.IntervalLine	Dodsfall.Klassifisering.Arsak[x].VarighetTekst	

Pre-conditions	Dødsfall må eksistere i Iris
Post-conditions	Dødsfall er slettet fra Iris og har fått oppdatering i Dår.
Main flow	1. Saksbehandler henter Lot-liste (liste over dødsfall basert på brukernavnet til saksbehandler fra Iris). 2. Snapshot tas av hvert dødsfall som skal tilbakeføres 3. For hver Lot i listen hentes korresponderende dødsfall opp fra databasen: a. Hele eksisterende Klassifisering på eksisterende dødsfall overskrives med ny klassifisering som tilbakeføres fra Iris. b. Endringer gjort i Iris angående operasjon tilbakeføres. c. Endringer gjort i Iris angående Ulykke tilbakeføres. d. Status på dødsfallet i Dår settes til «Initial», «rejected» eller «Final» om underliggende dødsårsak er satt. e. Lot slettes 4. Behandlingsresultat med antall tilbakeførte dødsfall returneres til saksbehandler.
Alternate flow	
Exceptional flow	
Issues/Notes	

11.3.2 UC Export records to IRIS (from 01.09.2013)

<table>
<tr><td>Id</td><td>UCEksportereDodsfallTilIris</td></tr>
<tr><td>Navn</td><td>Eksportere dødsfall til Iris</td></tr>
<tr><td>Trigger</td><td>DÅR saksbehandler velger å eksportere en lot av dødsfall til Iris.</td></tr>
<tr><td>Beskrivelse</td><td>
Brukstilfellet omfatter å overføre en lot av dødsfall fra Dår til Iris, hvor den skal kodes (klassifiseres). En lot tilhører saksbehandleren som oppretter loten.

En lot består av to tabeller i Iris, Ident og MedCod. Ident-tabellen inneholder informasjon om avdøde og generell informasjon om dødsfallet. MedCod-tabellen inneholder hver enkelt diagnose på et dødsfall. Den kan ha flere linjer som peker på samme dødsfall, og de bindes med ID-feltet CertificateKey.

Følgende dødsfall skal ikke overføres til Iris (filtreres bort):
• Duplikatoverføring. Samme dødsfall skal ikke kunne eksistere i Iris samtidig.
• Dødsfall som tidligere et klassifisert i eldre ICD kodeverk enn hva Iris benytter.
• Dødsfall som ikke har dødsmelding.

GUI
For detaljer rundt valg/søkekriterier se interaksjonsdesign for Dår Intranett.

Integrasjon
Data fra DÅR blir overført til Iris ved å skrive til følgende tabeller i Iris xxxxIdent, xxxxMedcod, der xxxx representerer brukerinitialene til saksbehandler. Ident representerer alle attributter knyttet til et dødsfall, bortsett fra Medcod som inneholder en liste med diagnosetekster (Årsak). Følgende data overføres fra DÅR til Iris:
(Mappingen under gjelder bare ved førstegangs overføring)

<table>
<tr><th>DÅR</th><th>Iris</th><th>Regel</th></tr>
<tr><td colspan="3">xxxxIdent</td></tr>
<tr><td>Dodsfall</td><td></td><td></td></tr>
<tr><td>Dodsfall.Uuid</td><td>Ident.CertificateKey</td><td>Uuid konverteres til Base64 før overføring.</td></tr>
<tr><td>Dodsfall.Tidspunkt</td><td>Ident.DateDeath</td><td></td></tr>
<tr><td>Dodsfall.Avdode</td><td></td><td></td></tr>
<tr><td>Dodsfall.Avdode.Fodselsdato</td><td>Ident.DateBirth</td><td></td></tr>
<tr><td>Dodsfall.Avdode.Kjonn.Kode</td><td>Ident.Sex</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Dodsfall.Operasjon</td><td></td><td></td></tr>
<tr><td>Dodsfall.Operasjon</td><td>Ident.ReasonSurgery
Ident .Status = Rejected</td><td>Hvis ikke null, samt at alle andre attributter for Dodsfall.Operasjon er null skal følgende tekst overføres «Operasjon utført»</td></tr>
<tr><td>TidspunktTekst</td><td>Ident.ReasonSurgery
Ident .Status = Rejected</td><td>Hvis utfylt. Legges først i Ident.ReasonSurgery strengen.</td></tr>
<tr><td>Tekst</td><td>Ident.ReasonSurgery
Ident .Status = Rejected</td><td>Hvis utfylt.</td></tr>
</table>
</td></tr>
</table>

	Tidspunkt	Ident.DateOfSurgery Ident .Status = Rejected	Hvis utfylt.
	Tidspunkt	Ident.RecentSurgery Ident .Status = Rejected	Hvis utfylt. Settes til «1»: Hvis <= 28 dager fra dødsdato Settes til «0»: Hvis > 28 dager fra dødsdato, eller blankt
	Dodsfall.YtreArsak		
	Dodsfall.YtreArsak	Ident.MannerOfDeath = 2 Ident.Status = Rejected	Dersom YtreArsak-entiteten ikke er null er dødsfallet en YtreArsak. Da blir manner of death 2 (ihht iris-dokumentasjon). Settes som rejected i IRIS. Se issue 1.
	Tidspunkt	Ident.DateOfInjury	Hvis utfylt
	TidspunktTekst	Ident.ExternalFreeText	Dersom dato ikke lar seg parse i Dår appendes tidspunktet også til fritekstfeltet. «YtreArsak tidspunkttekst:» + Tidspunkttekst
	Tekst	Ident.ExternalFreeText	Appender teksten til fritekstfeltet i Iris. Ingen egen felt for YtreArsaktekst «YtreArsak tekst:» + Tekst
	YtreArsakSted	Ident.PlaceOfOccurance	Koder fra WHO – mappes direkte.
	YtreArsakAktivitet	Ident.ActivityCode	Koder fra WHO – mappes direkte.
	~~Klassifisering.Skadediagnose.Kode~~	~~Ident.MainInjury~~	~~Dersom overføring ikke skjer for første gang~~
	SpesielleOmstendigheterA		
	Kode Tekst	Ident.MannerOfDeath Ident.FreeText	Dersom 1 settes MoD til 1 (Homicide). Dersom 2 settes MoD til 4 (Suicide). Dersom 5 settes MoD til 6 (Not Filled In). Dersom 9 settes MoD til 6. Dersom 4 settes MoD Til 3 (Pending Investigation) Dersom 3 appendes tekst til fritekstfeltet i

			Iris: «SpesOmsA: Misbruk av Narkotika»
	SpesielleOmstendigheterB		
	Kode Tekst	Ident.FreeText	Ingen av kodene i SpesOmsB kan mappes til Iris. Hvis denne er satt appendes tekst til fritekstfeltet: «SpesOmsB: (Tekst fra Dår).
	Dodsfall-Innkommendemelding-Dokument	Ident.CerImage	Alle dokumenter relatert til et dødsfall skal sammenslås til et bilde og lagres på..\DÅR\IrisTemp\ [Dodsfall-Uuid]
	Dodsfall.Saksbehandling		
	Registreringsstatus		Registreringsstatus settes til «UnderKoding»
	xxxxMedCod		
	Dodsfall.Klassifisering.Arsak1A		
	Arsak1A.Tekst	MedCod.TextLine	
	Arsak1A.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	Hver linje i MedCod kobles til Ident (dødsfall) med en Certificate Key
		MedCod.CodeOnly = 0 (eller 1)	Settes til 0 dersom Dodsfall.Saksbehandling. Registreringsstatus = «Ukodet», ellers 1. Hvis 1 vil Iris åpne MedCod-linjen i koderedigeringsmodus
		MedCod.LineNb = 0	LineNb bestemmer hvilken årsak det er snakk om i Iris. 0 mapper her til «Årsak 1A»
	Arsak1A.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	Overføring av diagnoser skjer ikke ved førstegangsoverføring. For hver diagnosekode på en årsak i Dår appendes disse til tekstlinjen «CodeLine» i MedCod, separert med «, ».
	Index		Index i Arsaksdiagnose tas vare på for å huske rekkefølgen på diagnosekodene slik de var i Iris. Når vi overfører andre gang husker vi

			denne rekkefølgen og de overføres i riktig orden.
	Dodsfall.Klassifisering.Arsak1B		
	Arsak1B.Tekst	MedCod.TextLine	
	Arsak1B.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	
		MedCod.CodeOnly = 0 (eller 1)	
		MedCod.LineNb = 1	
	Arsak1B.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	
	Index		
	Dodsfall.Klassifisering.Arsak1C		
	Arsak1C.Tekst	MedCod.TextLine	
	Arsak1C.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	
		MedCod.CodeOnly = 0 (eller 1)	
		MedCod.LineNb = 2	
	Arsak1C.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	
	Index		
	Dodsfall.Klassifisering.Arsak1D		
	Arsak1D.Tekst	MedCod.TextLine	
	Arsak1D.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	
		MedCod.CodeOnly = 0 (eller 1)	
		MedCod.LineNb = 3	
	Arsak1D.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	
	Index		
	Dodsfall.Klassifisering.Arsak1E		
	Arsak1E.Tekst	MedCod.TextLine	
	Arsak1E.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	
		MedCod.CodeOnly = 0 (eller 1)	
		MedCod.LineNb = 4	
	Arsak1E.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	
	Index		
	Dodsfall.Klassifisering.Arsak2		
	Arsak2.Tekst	MedCod.TextLine	
	Arsak2.VarighetTekst	MedCod.IntervalLine	
	Dodsfall.Uuid	MedCod.CertificateKey	
		MedCod.CodeOnly = 0 (eller 1)	
		MedCod.LineNb = 5	
	Arsak2.Arsaksdiagnose		
	Diagnose.Kode	MedCod.CodeLine	
	Index		

	Ved 2. gangsoverføring: Må da mappe diagnoser, og sette diagnosene i Iris til «Code Only»
Pre-conditions	
Post-conditions	Lot er skrevet til Iris sine tabeller: xxxxIdent, xxxxMedcod. Alle dødsfall i DÅR som er inkludert i lot har fått oppdatert status om at de er overført til Iris.
Main flow	5. Saksbehandler angir hvilke dødsfall (*DodsfallQuery*) som skal inkluderes i Iris-lot. 6. Saksbehandler sjekker antall dødsfall som skal overføres til Iris i henhold til resultatet av *DodsfallQuery*. 7. *DodsfallQuery* benyttes til oppslag i Dår for å hente et utvalg dødsfall. 8. Systemet sjekker om saksbehandler allerede har tilhørende lot-tabeller i Iris og oppretter disse automatisk dersom de ikke eksisterer. 9. DÅR eksporterer dødsfall til Iris ved å skrive til tabellene Ident og Medcod. a. Dødsfall eksporteres ved at listen dødsfall som spesifiseres i punkt 1 transformeres til en liste over lots. 10. DÅR markerer dødsfallene som er eksportert for å unngå duplikatoverføring. a. Verdien Dodsfall.Saksbehandling.Registreringsstatus settes til «UnderKoding» 11. DÅR saksbehandler mottar en behandlingsmelding som sier hvor mange dødsfall som er overført.
Alternate flow	Dødsfall skal pre-rejectes (settes som Reject -> Coder i Iris) dersom følgende kriterier er oppfylt: • Dødsfallet har mer enn ett tilhørende dokument • Dødsfallet har en ytre årsak • Operasjon er foretatt mindre enn 24 timer før døden inntraff • Hvis Spesielle Omstendigheter gruppe A er noe annet enn «Ukjent» (9).
Exceptional flow	
Issues/Notes	Vi må fange opp at andre meldinger (f.eks. obduksjon, tilleggsmeldinger) har kommet inn etter at et dødsfall er kodet. Det må ergo være en funksjon i GUI hvor man kan overføre dødsfall som har fått flere meldinger etter at den er blitt kodet. (Kan sjekkes ved å sammenligne klassifiseringsdato mot siste meldingsdato). 1. Sjekk om dagens løsning bruker «Reject», «MainInjury» (sannsynligvis) eller «CoderReject» ved pre-rejection f.eks. ved YtreArsak. Eventuelt Reject -> Coder. a. Kun ved å sette «CoderReject» unngår vi batch-prosessering. Denne verdien i databasen vises i Iris gui som en checkbox med «do not recode in batch processing» 2. Ikke mulig å redigere dato for YtreArsak i Iris. Det vil si at dersom vår «TidspunktTekst» i YtreArsak er en lesbar dato som ikke følger formatteringsreglene kan saksbehandlere ikke oppdatere datoen i Iris. 3. Iris skiller kun mellom "Natural" og "External Cuse" i dødsårsak. a. Vanskelig å skille YtreArsak fra andre "Spesielle omsetninger" i Dår. b. Kun ved MoD Natural er "External Cause"-feltet uredigerbart. c. I Dår er det kun ved «YtreArsak» (MoD «Accident») vi har mulighet til å sette ActivityCode og «PlaceOfOccurence». Hvis da «External Cause» er f.eks. selvmord er det vanskelig å tilbakeføre verdiene. d. Hva skal vi gjøre dersom Spesoms og MoD «Accident» er motsigende?

11.3.3 UC Search for records (from 01.09.2013)

Id	UCSøkEtterDødsfall
Navn	Søk etter dødsfall
Trigger	DÅR saksbehandler søker etter dødsfall basert på gitte kriterier.
Beskrivelse	Brukstilfellet omfatter søk etter et eller flere dødsfall i DÅR. Brukstilfeller er primært knyttet til å identifisere dødsfall som skal eksporteres til Iris (*UCEksportereDodsfallTilIris*), men benyttes også for å identifisere dødsfall som trenger annen saksbehandling. Søket skal skille mellom ofte brukte- og avanserte søkekriterier. Avanserte søkekriterier skal per default være skjult for saksbehandler, men med mulighet for vis/skjul. Søket skal benytte «AND» operator mellom kriteriene (*DodsfallQuery)* som er angitt. Alle søkekriterier som gir unike treff (f.eks. *Ident, Uuid*) skal ignorere andre søkekriterier. Det skal være mulig å velge en «årgang» søket skal gjelde for. Med «årgang» menes da året dødsfallet skjedde (*Dodsfall.Tidspunkt*). Default årgang skal være konfigurerbar. Det skal implementeres en begrensning mht. antall treff i søkeresultatet. Hver enkelt rad i søkeresultatet skal kunne markeres for overføring til Iris. Videre skal det være mulig å sortere søkeresultatet på alle kolonner. Selve søkeresultatet som skal sorteres - det vil si at det ikke skal eksekveres et nytt søk ved sortering. Enkelte søkekriterier skal kunne sendes med som parameter til søkesiden. Søket vil da automatisk fylle ut gjeldende søkekriterier og eksekvere søket. Typisk vil dette være parametriserte hyperlinker fra andre sider i løsning eller fra andre applikasjoner, som f.eks. datavarehuset. Se også *UCSokMedIdent* som er en utvidelse av dette brukstilfellet.
Pre-conditions	
Post-conditions	
Main flow	1. Saksbehandler angir hvilke dødsfall (*DodsfallQuery*) som det skal søkes etter. 2. Søkekriteriet (*DodsfallQuery)* benyttes til å søke i DÅR for å hente et utvalg dødsfall. 3. Applikasjonen logger alle søk som leverer data dekryptert. 4. DÅR saksbehandler mottar søkeresultatet.
Alternate flow	
Exceptional flow	
Issues/Notes	For detaljer rundet felter og GUI refereres det til Interaksjonsdesign, Domenemodell, UML diagrammer og faktisk implementasjon.

11.3.4 UC Request for additional documents (from 20.08.2013)

Id	UCBeOmTilleggsopplysninger
Navn	Be om tilleggsopplysninger
Trigger	Saksbehandler vurderer at en dødsmelding er for dårlig eller mangelfullt utfylt.
Beskrivelse	Brukstilfellet understøtter deler av prosessen med å generere et brev til et sykehus eller en kommunelege med forespørsel om utfyllende opplysninger knyttet til et dødsfall. Saksbehandler henter dødsfallet hvor det skal bes om tilleggsopplysninger. Saksbehandler må i samme skjermbildet ha tilgang til å åpne scannede dokumenter knyttet til dødsfallet, ettersom opplysninger om behandlende og utfyllende lege kun er tilgjengelig i de scannede dokumentene. Brukstilfellet understøtter ikke å generere et komplett dokument, men kan i tabellform vise relevant opplysninger om *Avdøde* og *Dødsfall*. Det vil være en manuell prosess knyttet til å opprette og adressere dokumentet. Systemet skal understøtte å laste opp dokumentet (via UCAdministrereDødsfall), samt velge *Opplysningstype* som er etterspurt for tilleggsmeldinger.
Pre-conditions	Dødsfallet er registrert i DÅR.
Post-conditions	En tabell viser relevant opplysninger om *Avdøde* og *Dødsfall*.
Main flow	1. Saksbehandler henter dødsfallet hvor det skal bes om tilleggsopplysninger. 2. Saksbehandler trykker «Vis tabell» 3. Systemet setter sammen opplysningene og genererer en tabell med opplysninger om *Avdøde* og *Dødsfall*.
Alternate flow	
Exceptional flow	
Issues/Notes	

11.3.5 UC Fill delivery database (from 07.02.2014)

Id	UCFyllUtleveringsbase
Navn	Kopiere data til DÅR-utleveringsdatabase
Trigger	En tidsplan (SQL Server Agent jobb) setter i gang ETL-pakken én gang i uken.
Beskrivelse	Persondata fra DÅR-database(DÅR-DB) og helsedata fra DÅR-datavarehus (DÅR-DVH) skal overføres til DÅR-utleveringsdatabase (DÅR-UTV-DB). **Utleveringsdatabasen** DÅR-UTV-DB er et Oracle-miljø som administreres av FHI avdelingen i Bergen. Helsedata skal ikke inneholde fødselsnummer. Utleveringsdatabasen er delt opp i to skjemaer(brukere): Helsedata(*DAR_LEVENDE_HELSE*) og persondata (*DAR_LEVENDE_PERSON*). **Kryptering** Fødselsnummer skal ikke lagres i utleveringsdatabase eller på utleveringstjener ukryptert, men skal først krypteres før lagring. Det er lagt til rette for både kryptering og dekryptering i Oracle-miljøet med egne funksjoner. **DÅR-databasen** Fødselsnummer er kryptert i DÅR-DB, men denne krypteringen «deles» ikke med utleveringsdatabasen. **Utvalg og lastemetode** I utgangspunktet er det meningen at hver overføring skal inneholde alle personer og alle dødsfall i DÅR-DB og DÅR-DVH. I første omgang vil dataene kopieres med en fullast, hvis

dette er for langsomt vil det bli vurdert om deltalast er nødvendig.

Felter
Definert i domenemodellen. Merk at det er feltet P_UUID som knytter en rad i helsedataene til korrekt person i persondataene.

Stykkevis overføring
Både persondata og helsedata skal overføres stykkevis i batcher. Antall batcher skal kunne settes av brukeren med SSIS-parameteren *DefaultAntallBatcher*. Hvis parameteren er satt til 1 vil flyten overføre alle dataene i én omgang. Parameteren må være større enn 0. Batchene nummereres fra 0 til n-1, hvor n = antall batcher.

Innenfor en batch finner vi fødselsnumre som tilfredsstiller uttrykket:
Fødselsnummer mod #batcher = batchindeks, batchindeks = 0 ... #batcher

Alle batcher i en overføring vil normalt foregå innenfor én SSIS-kjøring, men hvis feil oppstår kan overføringen strekkes seg over flere SSIS-kjøringer.

Ved å kjøre stykkevis løftes færre rader data fra kilden og på den måten reduseres også minnepresset. Dette gir en jevnere overføring med økt ytelse og mindre minne og derfor også diskbruk.

Frys
Når lastingen starter lastes data fra DÅR-DB og DÅR-DVH til mellomtabeller i forkammeret (*MellomtabellHelse* og *MellomtabellPerson*). På denne måten vil alle batcher laste data fra samme datasett, uavhengig av hva som skjer av oppdateringer i DÅR-DVH eller DÅR-DB etter utleveringsjobben har startet eller om utleveringsjobben strekker seg over flere SSIS-kjøringer. Frysing skjer når batch med batchindeks 0 overføres.

Robust overføring
Hvis en utleveringskjøring stopper opp på grunn driftsforstyrrelse skal kjøringen ha mulighet til å fortsette fra batchen som ble avbrutt ved forrige kjøring.

Brukeren skal kunne overkjøre dette ved å sette SSIS-parameteren *FortsettSisteJobb* til FALSE. Denne er normalt satt til TRUE.

Jobbtabell
Når en ny jobb starter vil den enten:

- Legge til en ny rad:
  - Tabellen er tom
  - *FortsettSisteJobb* er FALSE
  - Forrige kjøring ble fullført og er markert som SUKSESS
- Oppdatere siste rad (den med høyest løpenummer):
  - Forrige kjøring ble avbrutt før status ble satt
  - Forrige kjøring feilet
- Ingen endring: Overføring starter ikke fordi parametere har ulovlig verdi eller fordi en annen jobb allerede kjører.

Når en rad blir lagt inn vil AntallKjoringer bli satt til 1. For hver gang den starter en omkjøring vil denne kolonnen bli inkrementert med én.

Når en ny legges til blir OriginalAntallBatcher satt lik DefaultAntallBatcher. Hvis en jobb må omkjøres brukes OriginalAntallBatcher til å bestemme antall batcher og ikke DefaultAntallBatcher.

<table>
<tr><td></td><td>
Hver overføringsjobb har en rad i jobbtabellen. Hver jobb har en status:

KJØRER: Utleveringsjobben kjører.
OMKJØRER: Utleveringsjobben har startet på en omkjøring fordi den feilet i forrige kjøring.
SUKSESS: Alle batcher er overført og ingen feil oppstod
FEILET: Siste batch som skulle overføres feilet.
UDEFINERT: Jobb feilet før status ble satt til KJØRER.

SisteBatchIndex viser hvilke batch som den enten arbeider med å overføre (Status er OMKJØRER eller KJØRER), batch som feilet (status er FEILET) eller den er ferdig med å overføre (status er SUKSESS). Kolonnene kan ha verdiene: NULL, 0, 1, …, OriginalAntallBatcher

Kolonner som heter AntallPersonerXXXX er tellekolonner som forteller hvor mange rader som er hentet fra kilden eller levert til Bergen.

Parametere
<table>
<tr><th>Navn</th><th>Scope</th><th>Beskrivelse</th></tr>
<tr><td>DefaultAntallBatcher</td><td>Pakke</td><td>Antall batcher kjøringen skal deles opp i. Benyttes så sant det ikke er en omkjøring. Da vil verdien hentes fra jobbtabellen via feltet OriginalAntallBatcher. Lovlige verdier 1,…</td></tr>
<tr><td>FortsettSisteJobb</td><td>Pakke</td><td>Normal kjøring vil denne verdien være satt til TRUE. Hvis den derimot er satt til FALSE, vil jobben ikke sjekke i jobbtabellen om forrige jobb feilet eller ikke, men istedenfor tvinge igjennom en ny kjøring med en ny rad i jobbtabellen.</td></tr>
<tr><td>Sone</td><td>Prosjekt</td><td>To lovlige verdier TEST eller PRODUKSJON</td></tr>
</table>
</td></tr>
<tr><td>Pre-conditions</td><td>DÅR-DVH og DÅR-DB må være synkronisert, slik at hvert dødsfall i DÅR-DVH kan knyttes til en avdød person i DÅR-DB.

Ikke et like strengt krav at hver avdødde person i DÅR-DB kan knyttes til et dødsfall i DÅR-DVH, men hvis avdødde legges til DÅR-DB etter at forkammeret til DÅR-DVH er lastet men før utleveringen har startet, vil disse personene bli inkludert i LEVENDE_PERSON men ikke i DAR_LEVENDE_HELSE.</td></tr>
<tr><td>Post-conditions</td><td>Etter at jobben er fullført skal DÅR-utleveringsdatabase ha en eksakt kopi av DÅR-datavarehus/database for de feltene definert i domenemodellen som kopieres over.</td></tr>
<tr><td>Main flow</td><td>

Legge til ny rad i jobbtabellen
Sett status til «KJØRER»
Tømme mellomtabellene
Hent helsedata fra db-utsnitt i DÅR-DVH og fyll mellomtabellen for helsedata
Hent helsedata fra db-utsnitt i DÅR-DVH og fyll mellomtabellen for helsedata
Hent persondata fra krypto-skjemaet i DÅR-DB og fyll mellomtabellen for Persondata. Samtidig last oppslagstabellen for P_UUID, slik at det blir mulig å finne P_UUID basert på Dødsfall UUID
Overføre helsedata fra mellomtabell for helsedata til helsedatadataskjemaet i DÅR-utleveringsdatabase. P_UUID skapes ved slå opp i P_UUID oppslagstabellen.
Hent ut data fra mellomtabellen for persondata og dekryptere fødselsnummer i
Overføre men ikke lagre ukryptert fødselsnummer
Kryptere fødselsnummer med funksjoner som er installert i DÅR-utleveringsdatabase.

</td></tr>
</table>

	11. Lagre kryptert kryptert fødselsnummer persondataskjemaet i DÅR-utleveringsdatabase. 12. Overfør kodeverkene 13. Skriv til notfikasjonsloggen
Alternate flow	
Exceptional flow	
Issues/Notes	**Nettverksproblemer** Det viser seg at nettverket over til Bergen er ustabilt og at kjøringen brytes før den er ferdig. For å unngå dette problemet er robusthet bakt inn i designet. **Drivere** Standard Oracle-driver som følger med SSIS viser seg å ha svært dårlig ytelse. Det er også mulighet for at den har en minnelekkasje. Driveren fra Attunity har vesentlig bedre ytelse og bruker mindre minne men har noen ulemper også: - Fler avhengigheter i utrullingen - De har kun source/target-objekter og mangler bla. SQL-task og «OLE Db Cmd» - Krever «Sql Server Entreprise»-lisens **Minnepress og ytelse** Uten egne tiltak for kjøring medførte lastingen at alt minne på serveren ble oppbrukt og at den startet med swapping/trashing. Dette medførte at jobben nesten stoppet opp. Dette blir unngått med to tiltak: - Redusere minnebruket - Overføre data i mindre bolker(batcher) som hver bruker mindre minne.

11.4 Total overview of use cases changes

Rules applied in the comparisons:

- The versions that only differed in minor syntactic changes (e.g. changes in the spell of the words, adding comma, period, etc.) are not included in the comparisons overview.

11.4.1 UC Export records to IRIS

UC Export records to IRIS				
1	**versions comparisons**	**UC element**	**Changes**	**Functional changes**
	V0-V1 (22.01.2013 - (02.01.2013)	Issues / notes	How to capture new messages related to existing records after a record have been processed (coded) in IRIS. Need for a function that can export records that are already coded but have received new messages was registered.	
2	V1-V2 (02.01.2013-	Description	16 new rows (fields) added in the IRIS integration specification table	Y

	(14.02.2013)	Main flow	1. Input parameters to select records added (step 1.a) 2. Technical description of Dodsfallquery used in the query of records (step 1.b) added 3. Description of “lot” in IRIS added (step 2.a) 4. Update rule for Coding status added(step 3.a)	4.Y
		Issues / notes	New issues on IRIS integration added, questions about different possible solutions added	
3	V2-V3 (14.02.2013- (28.02.2013)	Description	1- Additional description of the IRIS component (lot) 2- GUI specification modified **IRIS integration** 3- One change in the specification of a translation rule 4- 12 new rows (fields) in specification of the IRIS integration table where 4 were complete specification and 8 without translation rules 5- Update rule for Coding status added (the same was done in the post-condition)	3.Y 4.Y 5.Y
		pre-conditions	Removed	
		post-conditions	Update rule for coding status added	Y
		Main flow	Information on input parameters removed.	
		Alternate flow	To-do notes added	
		Exceptional flow	To-do notes deleted	
		Issues / notes	New notes added, some of the old removed	

4	v3-v4 (28.02.2013- (18.03.2013)	Description	1. GUI specification a. Change in the name of the field b. Specification of a new field 2. IRIS integration a. Change of Ulykke til Ytrearsak b. Change of one IRIS field c. Addition of 2 new rows (complete specification) d. One translation rule changes e. CDR and IRIS Fields for Arsak1A.Arsaksdiagnose to Arsak1E.Arsaksdiagnose added (addition of 5 new rows) 3. Technical solution for second times processing of a record added	 2b.Y 2c.Y 2d.Y 2e.Y
		Post-conditions	Update rule for Coding status removed	
		Main flow	1. Pre-condition included in step 4(check for existence of lot tables in IRIS before export can be executed) 2. Description of “lot” removed	1.Y
		Alternate flow	To-do about the second time processing of a record removed	
		Issues / notes	1. A new questions added 2. Ulykke was changes to Ytrearsak.	
5	V4-V5 (18.03.2013- (16.04.2013)	Description	Description of GUI specification modified	
		Alternate flow	Five criteria for alternative flow added	Y
		Issues / notes	Technical info on IRIS integration added	
6	V5-V6 (16.04.2013- (15.08.2013)	Description	1. New condition for Exceptional flow added (the records that should not be transferred to IRIS) 2. GUI specification changed 3. IRIS integration: two changes in translation rules	1.Y 3.Y
7	v6-v7 (15.08.2013- (01.09.2013)	Description	GUI specification removed	

11.4.2 UC Import records from IRIS

UC Import records from IRIS				
	Versions comparisons	**UC element**	**Changes**	**Functional changes**
1	V0-V1 22.01.2013 -14.02.2013	Description	Headline of the table of the specification of IRIS integration added	
		Main flow	overall description of main scenario added (in one sentence)	
2	V1-V2 14.02.2013 -28.02.2013	Description	GUI specification on criteria for choosing records added	Y(?)
		Main flow	Steps of the Main flow(2, 3a-3e and 4) added	Y
		Issues / notes	issues and questions about the coding statuses, follow-up queues and IRIS integration added	
3	V2-V3 28.02.2013 -18.03.2013	Description	a syntax change	
4	V3-V4 18.03.2013-16.04.2013	Description	GUI specification changed	Y(?)
5	V4-V5 16.04.2013-18.08.2013	Description	1. GUI specification modified 2. 14 new rows in the IRIS integration specification table added	2.Y
6	V5-V6 18.08.2013-01.09.2013	Description	1. Exception cases for records that cannot be imported back added 2. Description of *lot* added 3. GUI specification substituted with a reference to interaction design	1.Y
		Pre-conditions	added	Y
		Post-conditions	added	Y
		Main flow	1. Step 1: Status Final was removed from the selection criteria in IRIS for records to be imported 2. Step 3d: coding statuses were changed to initial,	1.Y 2.Y

			rejected and final by import if the cause of death was defined.	
		Issues / notes	Removed	

11.4.3 UC Search for records

	UC Search for records			
	Versions comparisons	**UC element**	**Changes**	**Functional changes**
1	V0-V1 18.03.2013- 15.08.2013	Description	1. Description of the initial purpose of the search function in relation to export and import functions added 2. Division of Simple and advanced search described 3. Limitation of the number of the results discussed 4. Possibility to export to IRIS for search results described 5. Sort on all of the columns described 6. Possible criteria and precedence of Ident over other criteria described	Y(?)
		Issues	References to GUI, interaction design, domain model and implementation added	
2	V1-V2 15.08.2013- 01.09.2013	Description	1. Reference to extended use case added 2. Requirement of logging by decryption of sensitive info removed	1.Y 2. the requirements were moved and specified in the extended use case
		Post-conditions	Removed (decryption of log removed)	Same as above

11.4.4 UC Request for additional documents

UC Request for additional documents				
	Versions comparisons	**UC element**	**Changes**	**Functional changes**
1	V0-V1 22.01.2013- 14.02.2013	Description	1. Complementary info about the process of letter generation added. 2. Fields to fill out by requesting additional info specified.	

			3. Template (reference to predefined templates that are found) added. 4. Business rules described. 5. Description of generation of the additional document added.	
		Pre-conditions	Added	Y
		Post-conditions	Added	Y
		Main flow	Steps (3) of the flow added	Y
		Issues	1. Solution suggestion added 2. Questions added 3. Link to info for integration with RESH register specified.	
		Description	Note about a new version of the use case which was described in UC Administrate Death records added.	
2	V1-V2 14.02.2013- 01.08.2013	Description	1. the note from previous version removed 2. specification of fields to fill out removed 3. Look up in RESH and kommunelege listen are excluded from the functionality, Template, business rules, document generation Description removed	The changes are reflected in the Main flow and Post-conditions
3	V2-V3 01.08.2013- 20.08.2013	Post-conditions	Conditions are changed according to the changes in the functionality (instead of a letter or doc, a table view was provided)	Y, The changes reflect changes in the scope of the function
		Main flow	2 steps modified 2 steps removed according to the changes in the function (and requirements)	Y
		Issues/ Notes	Removed	

11.4.5 Data capture use cases

	Use cases of Data capture	**Versions comparisons**	**UC element**	**Changes**	**Functional changes**
1	UC Process SSB Other Documents V0: 14.06.2013 V1: 28.08.2013	V0-V1	Post-conditions	Modified (2 possible conditions are either a record created or a record updated with a new message)	Y

	V2: 02.09.2013		Main flow	Step 1.a added (ref. to Exceptional flow if validation fails) 7 Steps removed due to the changes in handling update of existing records(the changes agree with the changes in the Post-condition)	Y
			Alternate flow	Removed (due to the changes in the Main flow and Post-conditions)	
			Exceptional flow	substituted with the reference to *UC Process SSB melding*	
		V1-V2	Main flow	Two new steps added (2.b og 2.b.1) 2.b.If statement for coding status = underoding and 2.b.1 If yes it should be registered in the notification log	Y
2	UC Process SSB Death Abroad V0: 14.06.2013 V1:28.08.2013 V2:02.09.2013 V3:14.10.2013	V0-V1	Description	substituted with the reference to UC Process Death Message	
			Main flow	Removed (substituted with the reference to UC Process Death Message)	
			Alternate flow	Removed (substituted with the reference to ucprocessDødsmelding)	
			Exceptional flow	Removed (substituted with the reference to UCProcessSSBmelding)	
		V1-V2	Description	The reference to UC Process Death Message removed	
			Main flow	All steps (8) added	Y
			Post-conditions	Modified (2 possible conditions are either a record created or a record updated with a new message)	Y
			Alternate flow	Removed (due to the changes in the Main flow and Post-conditions)	
		V2-V3	Main flow	Two new steps added: 2.iiii.Check for coding status = underCoding and 2.iii.1. If yes, the update should be registered in the notification log	Y

3	UC Process SSB Death Message V0: 28.06.2013 V1:28.08.2013 V2:02.09.2013 V3:06.11.2013	V0-V1	Description	Substitution with references to *UC Process SSB Message* and domain model	
			Main flow	2 steps changed (4.1 and 4.2) due to the changes on record updates when new messages related to existing records received.	Y
			Issues	New issues about duplicate messages and update when new messages received, rules of update suggested	
		V1-V2	Main flow	Steps were modified (6 new steps added) due to the changes in update rules and which fields to update. Specification of fields that should be updated followed the Main flow.	Y
			Alternate flow	Removed due to the changes in the Main flow	
			Exceptional flow	Removed (Substituted with reference to *UC Process SSB Message*)	
			Issues	Removed	
		V2-V3	Main flow	Two new steps added (2.iv og 2.iv.1) 2.iv.Check for coding status = underCoding and 2.iv.1. If yes, update should be registered in the notification log	Y
4	UC Process SSB Death message From Police V0: 24.06.2013 V1: 28.08.2013	V0-V1	Description	Substituted with references to *UC Process Death Message*	
			Main flow	Substituted with reference to *UC Process Death Message*	
			Alternate flow	Substituted with reference to *UC Process Death Message*	
			Exceptional flow	Removed (Substitution with reference to *UC Process Death Message*)	
		no further versions available			

5	UC Process SSB Message (the generic use case) V0: 24.06.2013 V1: 29.08.2013 V2: 06.11.2013	V0-V1	Description	1.Mapping info about dodskommune2 attribute deleted 2.Refernce to domain model for specification of fields related to ulykke and obduksjon added	
			Alternate flow	2 new steps added to check duplicate messages related to a record	Y
			Issues	Removed (the issues were related to the Alternate flow)	
		V1-V2	Alternate flow	Removed (because the six special use cases got their own specific Alternate flow)	
6	UC Process SSB Autopsy Message V0: 28.06.2013 V1:28.08.2013 V2:02.09.2013 V3:06.11.2013	V0-V1	Description	Substituted with references to UC Process SSB Message	
			Main flow	Step 4.i changed, 4.i.1 removed due to changes in handling update of existing records when new messages received	Y
			Issues	Issues on record update added and rules for update suggested	
		V1-V2	Post-conditions	Modified (2 possible conditions are either a record created or a record updated with a new message)	Y
			Main flow	Step 1.a added, Steps 4.b to 4.1 removed due to changes in update of existing records when new messages received	Y
			Alternate flow	Removed due to the changes in the Main flow and Post-condition	
			Exceptional flow	Removed (Substitutied with the reference to *UC Process SSB Message*)	
			Issues	Removed (issues were related to update rules)	
		V2-V3	Main flow	Two new steps added: 2.ii.Check for coding status= underCoding and 2.ii.1. if yes, update should be registered in the notification log	Y
7	UC Process SSB Additional info V0: 14.06.2013 V1: 28.08.2013 V2: 02.09.2013 V3: 14.10.2013	V0-V1	Description	Substituted with references to *UC Process SSB Message*	
		V1-V2	Post-conditions	Modified (2 possible conditions are either a record created or a record updated with a new message)	Y

			Main flow	Step 1.a added, Steps 4.b to 4.1 removed due to changes in update of existing records when new messages received	Y
			Alternate flow	Removed due to the changes in the Main flow and Post-conditions	
			Exceptional flow	Removed (Substitutied with reference to *UC Process SSB Message*)	
		V2-V3	Main flow	Two new steps added: 2.ii.Check for coding status = underCoding and 2.ii.1. If yes, update should be registered in the notification log	Y

11.4.6 ETL use cases

11.4.6.1 UC Synchronize ICD Codes

UC Synchronize ICD Codes				
	Versions comparisons	**UC element**	**Changes**	**Functional changes**
1	V0-V1 (14.08.2013 - 02.01.2014)	Invariant	1. Description of the target tables added 2. Description of the source files added 3. Handling different ICD codes versions added	Y(?)
		Description	1. Additional Description on upload of ICD codes added 2. Description of the comparison of reference data in the CDR with the ICD codes added	
		Main flow	All the previous steps removed and 13 new steps added	Y(?)
		Exceptional flow	Removed	
		Post-conditions	The tables that will be update (filled with data) are specified. (Tabellene IcdKodeverkType, IcdKapittel, IcdBlokk, IcdKode og IcdKodeDar er tømt og deretter fylt opp med data)	Y(?)

As only two versions of this use case were available for comparison, it was uncertain if the changes were related to functional changes.

11.4.6.2 UC Fill Delivery Database

ETL UC Fill Delivery Database				
	Versions comparisons	**UC element**	**Changes**	**Functional changes**
1	V0-V1 (31.07.2013 - 26.11.2013)	Description	Databases names précised Additional Description of batch transfer, freeze, robust transfer, job table, status of jobs, parameters, solution package added	Y(?)
		Pre-condition	Name of databases précised Additional Description about the synchronization of databases in the data warehouse solution added	Y(?)
		Issues /notes	Technical issues such as network problems, efficiency, Oracle drivers, memory and efficiency added	
		Main flow	Removed	Y(?)
		Post-condition	Removed	Y(?)
2	V1-V2 (26.11.2013-29.11.2013)	Issues /notes	Syntax correction	
3	V2-V3 (29.11.2013-09.01.2014)	Issues /notes	The resolved issues on differences between test and production environment removed	
4	V3-V4 (09.01.2014-07.02.2014)	Main flow	The steps (7) added	Y(?)
		Post-condition	Reference to domain model added	
		Description	Syntax changes	
		Trigger	Specified (as a job running once a week)	

11.5 Detailed Requirements Engineering process in CDR

Activity	Input	Output	Roles	Techniques	Frequency
Preliminary Analysis & Knowledge gain	• Business problem • Goals • Documents - User manual and documentation for existing systems - Existing work flows and processes - Existing data contracts with external stakeholders - Standards (such as ICD codes) - Regulations, laws (such as CDR regulation and personal privacy law)	• Domain knowledge • Identification of stakeholders, their roles and responsibilities • Analysis artifacts - Initial domain model - High-level modeling diagrams such as workflow diagram • Initial scope of the solution • Initial use case diagrams • Decision of priorities • Time- and activity schedules	• System architect • Project manager • Developers • Stakeholders • Interaction designer	• Documents reviews • User meetings / presentations by users and other stakeholders • Workshops • High-level diagram modeling	Continuously
Requirements analysis and development START (the activities are conducted iteratively in the development iterations)					
Investigation of available data related to the requirements	• Requirements planned to be developed in the iteration (according to the priorities) • Output from the analysis activities where applicable	• Understanding requirements • Use case diagrams • High-level modeling diagrams such as high-level sequence diagram • Considerations of alternative solutions • Identification of issues	• Interaction designer • Developers • System architect • Project manager • Involved stakeholders	• Interviews • Direct contact with the involved stakeholders • Document reviews • Use case diagrams • Visualizations • High-level diagram modeling • Reusing solutions of similar requirements	Continued until the requirements are ready to be developed
Capturing the details of requirements	• Output from the activity above	• User stories • Use cases • Paper notes, whiteboard sketches, photos of notes and whiteboard • Modeling diagrams (state machine diagram, sequence diagram)	• Interaction designer • developers • Involved stakeholders	• Modeling diagrams • User involvement • Use cases • User stories • Development and test activities	Continued until the necessary details of the requirements are captured

Activity	Input	Output	Roles	Techniques	Frequency
Prototyping (design)	• Uer stories (updated)	• Design prototypes	• Interaction designer	Design tool (Wireframes)	Running until the design was completed and approved
Technical review (design)	• Design prototypes	• Design prototypes (updated)	• Project manager	• Domain knowledge • Document reviews • Requirements artifacts such as notes, diagrams, etc.	Running until the prototypes was approved
Users review (design)	• Design prototypes	• Feedback from users	• Users • Project manager	• Discussions • Demos	Running until the design was approved
Final design approval	• Design prototypes	Status of the related UC updated (by project manager)	• Users • Project manager	• Discussions	Running until the design was approved
Requirements elaboration and specification	• Details of the requirements • Discovered errors, feedback from users • Outputs from the activities above	• Use cases • Modeling diagram (state machine diagram)	• Developers • System architect • Involved stakeholders	• Use cases technique • User involvement • Modeling diagrams • Development and test of the solution	Continued until the requirements specification is complete
Requirements analysis and development END					
Users review	• Features	• Feedbacks • Decisions on changes	• Developer • Users • System architect	• Demo presentation • Discussions	Continuously until functionality is approved
Approval (user acceptance)	• Features	• Final approval	• Users • Project manager	• Features tryout	Done for each functionality
Use cases review and Quality assurance	• Implemented feature and use cases specification	• Final use cases (reviewed and updated)	• Project manager	• Review routines	Done for each use case

11.6 Technological platform in the NIPH

The following technologies are utilized in the NIPH development platform:

- .Net , C#, ASP.NET
- HTML5, Ajax
- XML Web Services
- Windows Communication Foundation
- ebXML, XML, XML Schema
- WIF (Windows Identity Foundation)
- SQL Server
- Analysis Services
- Reporting Services
- SQL Server Integration Services (SSIS)
- Entity Framework
- Enterprise Library
- Internet Information server
- Microsoft Message Queue
- Active Directory Federation Services
- Network Load Balancing

Development tools:

- Visual Studio 2010
- Business Intelligence Development Studio
- Team Foundation Server
- Enterprise Architect
- XML Spy
- Red Gate (Databaseverktøy)

11.7 The old CDR solution technology platform

- AMD machines with LINUX
- CITRIX
- Oracle Developer Forms 6i /Report 6i
- Oracle encryption tools for database
- Stored procedures (pl/sql) in the database, shell scripts
- PGP Encryption tool for TIF files
- SAS Foundation v. 9.2
- IRIS

11.8 Changes in design prototypes

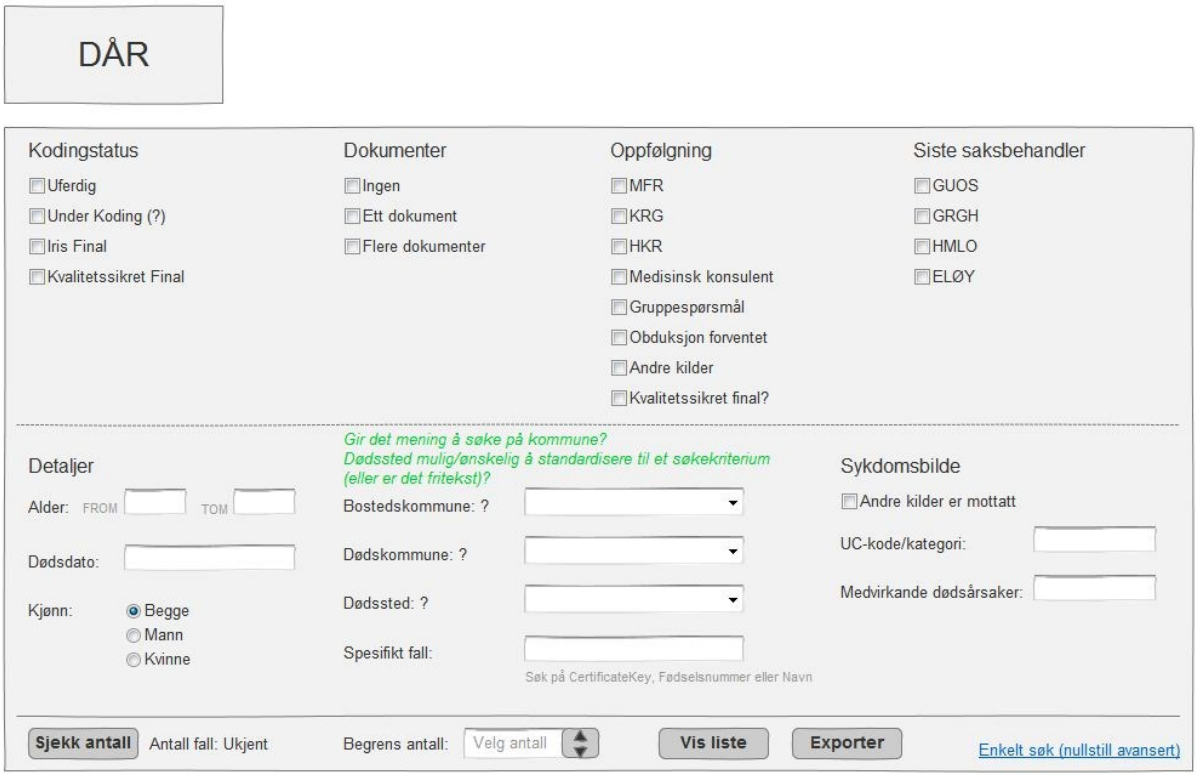

Figure 46 design prototype for advanced search from 01.03.2013 (from internal documents presented in Table 4)

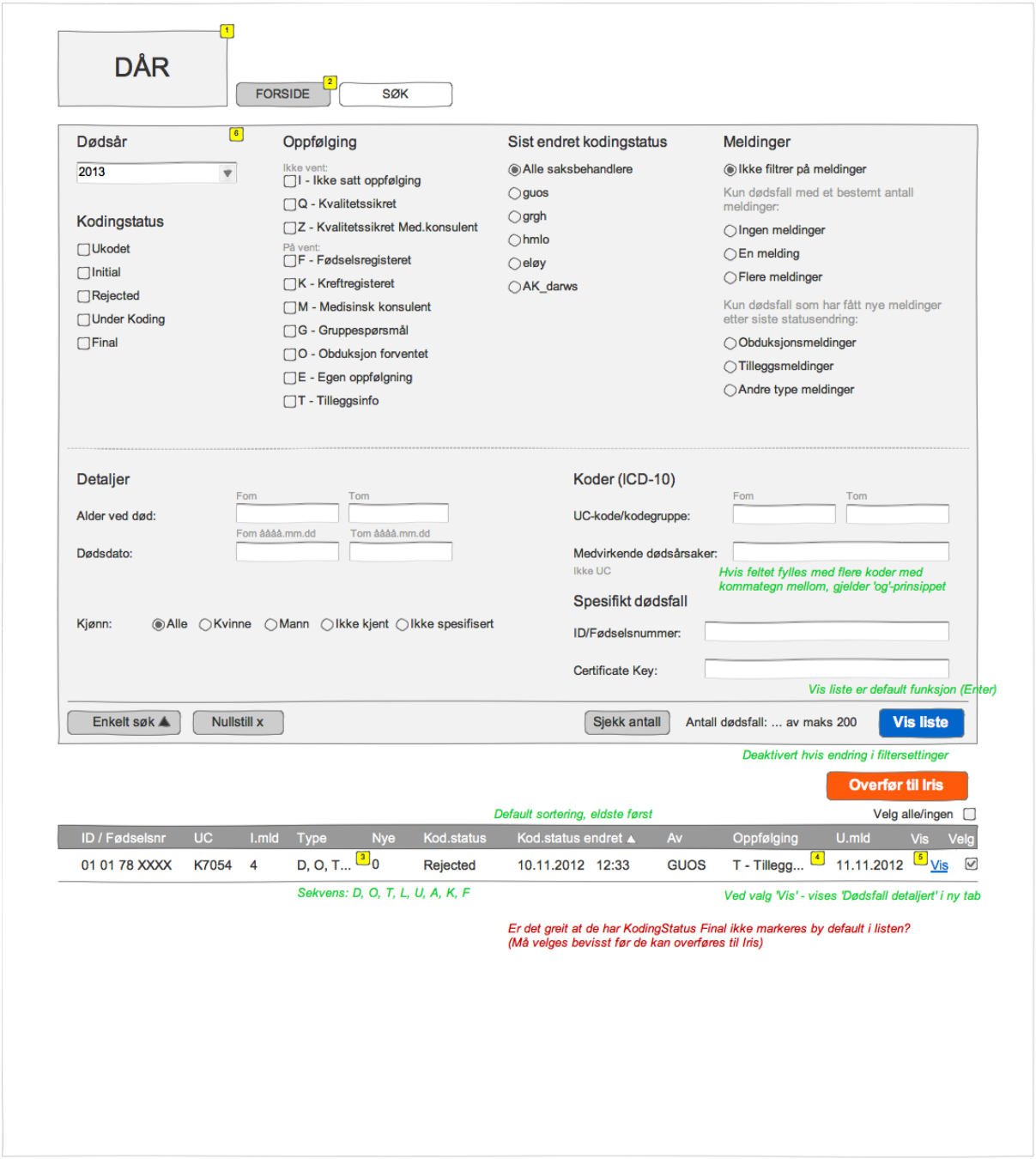

Figure 47 Design prototype for advanced search from 26.04.2013 (from internal documents presented in Table 4)

11.9 Three versions of Import records from IRIS use case under development

11.9.1 UC Import records from IRIS Version 14.02.2013

<table>
<tr><td>Id</td><td>UCImportereDodsfallFraIris</td></tr>
<tr><td>Navn</td><td>Importere dødsfall fra Iris</td></tr>
<tr><td>Trigger</td><td>DÅR saksbehandler velger å importere en lot av dødsfall til Iris.</td></tr>
<tr><td>Beskrivelse</td><td>Snapshot
Overskrive
Slett fra IRIS
Status – ligger ikke lengre i IRIS
Mapping på tilbakeføring til domene

<table>
<tr><th>Iris</th><th>Dår</th><th>Regel</th></tr>
<tr><td colspan="3">xxxxIdent</td></tr>
<tr><td>Dodsfall.Uuid</td><td>Ident.CertificateKey</td><td></td></tr>
</table>
</td></tr>
<tr><td>Pre-conditions</td><td></td></tr>
<tr><td>Post-conditions</td><td></td></tr>
<tr><td>Main flow</td><td>Saksbehandler henter Lot (liste over dødsfall basert på brukernavnet til saksbehandler og status «Final» fra Iris.</td></tr>
<tr><td>Alternate flow</td><td></td></tr>
<tr><td>Exceptional flow</td><td></td></tr>
<tr><td>Issues/Notes</td><td></td></tr>
</table>

11.9.2 UC Import records from IRIS Version 28.02.2013

<table>
<tr><td>Id</td><td>UCImportereDodsfallFraIris</td></tr>
<tr><td>Navn</td><td>Importere dødsfall fra Iris</td></tr>
<tr><td>Trigger</td><td>DÅR saksbehandler velger å importere en lot av dødsfall til Iris.</td></tr>
<tr><td>Beskrivelse</td><td>Snapshot
Overskrive
Slett fra IRIS
Status – ligger ikke lengre i IRIS

GUI
Saksbehandler har følgende valg for å bestemme hva som skal tilbakeføres:

<table>
<tr><th>Kriterier</th><th>Default verdi</th><th>Kommentar</th></tr>
<tr><td>Lot-navn</td><td>Brukernavn til saksbehandler</td><td>Bruker tilbakefører alltid fra egen lot</td></tr>
<tr><td>Kø</td><td>Blankt</td><td>Dødsfall i Iris kan ha blitt lagt i en kø, man kan velge å tilbakeføre kun dødsfall fra en gitt kø</td></tr>
<tr><td>Fødselsnummer</td><td>Blankt</td><td>Kan tilbakeføre et enkelt dødsfall basert på fnr.</td></tr>
</table>

<table>
<tr><th>Iris</th><th>Dår</th><th>Regel</th></tr>
<tr><td colspan="3">xxxxIdent</td></tr>
</table>
</td></tr>
</table>

	Dodsfall.Uuid	Ident.CertificateKey	
Pre-conditions			
Post-conditions			
Main flow	5. Saksbehandler henter Lot-liste (liste over dødsfall basert på brukernavnet til saksbehandler og status «Final» fra Iris). 6. Snapshot tas av hvert dødsfall som skal tilbakeføres 7. For hver Lot i listen hentes korresponderende dødsfall opp fra databasen: a. Hele eksisterende Klassifisering på eksisterende dødsfall overskrives med ny klassifisering som tilbakeføres fra Iris. b. Endringer gjort i Iris angående operasjon tilbakeføres. c. Endringer gjort i Iris angående Ulykke tilbakeføres. d. Status på dødsfallet i Dår settes til «Tilbakeført» eller «Tilbakeført_Final» om underliggende dødsårsak er satt. e. Lot slettes 8. Behandlingsresultat med antall tilbakeførte dødsfall returneres til saksbehandler.		
Alternate flow			
Exceptional flow			
Issues/Notes	- Kø: Mappes fra Iris til kø i Dødsfall saksbehandling. Burde ha samme Iris-konfig som i produksjon. - Statuser på dødsfall: Tilbakeført, Tilbakeført_Final (Sistnevnte dersom det eksisterer en Underlying Cause) - ActivityCode og PlaceOfOccurance: Dersom ulykke (Manner of death 2) mappes disse i Dår ved tilbakeføring. ISSUE: Dersom AC og PoO er satt ved en annen Manner of death (f.eks. mord/selvmord) kan det ikke tilbakeføres da Sted/Aktivitet er i Ulykke-entitet i Dår. - Tilbakeføring av FreeText. Dette feltet fylles ved overføring til Iris når verdier er vanskelige å mappe. - ExternalFreeText? Feltet under ActivityCode og PlaceOfOccurance? Det er et datafelt i Iris, men vises ikke i Gui.		
	-		

11.9.3 UC Import records from IRIS Version 18.08.2013

Id	UCImportereDodsfallFraIris
Navn	Importere dødsfall fra Iris
Trigger	DÅR saksbehandler velger å importere en lot av dødsfall fra Iris.
Beskrivelse	Brukstilfellet omfatter tilbakeføring (import) av et dødsfall fra Iris, klassifikasjonssystemet for dødsfall. Snapshot Overskrive Slett fra IRIS Status – ligger ikke lengre i IRIS **GUI** Saksbehandler har følgende valg for å bestemme hva som skal tilbakeføres:

Kriterier	Default verdi	Kommentar

Lot-navn	Brukernavn til saksbehandler	Tekstboks. Pre-utfylt med navnet på innlogget saksbehandler.
Tilbakeføringskriterier	«Alle med kodingstatus final»	Radiobuttonliste med valgene «Alle med kodingstatus final», «Alle unntatt Egen oppfølging» og «Alle».

Iris	Dår	Regel
xxxxIdent		
Ident.CertificateKey	Dodsfall.Uuid	Parses fra base64 til Guid og matches med dødsfall i Dår
Ident.MannerOfDeath	Dodsfall.YtreArsak	Settes til null dersom MannerOfDeath = 1.
Ident.AcmeCodes	Dodsfall.Klassifisering.AcmeCodes	
Ident.ErnCodes	Dodsfall.Klassifisering.ErnCodes	
Ident.SelectedCodes	Dodsfall.Klassifisering.SelectedCodes Dodsfall.Klassifisering.Arsak[x]. VarighetTekstStandardisert Dodsfall.Klassifisering.Arsak[x]. Arsaksdiagnose.Diagnose Dodsfall.Klassifisering.Arsak[x]. Arsaksdiagnose.indeks	Feltet fra Iris er formatert slik at vi kan plukke ut diagnosene med tilhørende linje og indeks.
Ident.CodingVersion	Dodsfall.Klassifisering.SystemVersjon	
Ident.UCCode	Dodsfall.Klassifisering. UnderliggendeDiagnose.Kode	Opprettes i Dår dersom den ikke finnes
Ident.SubstitutedCodes	Dodsfall.Klassifisering.SubstitutedCodes	
Ident.Status	Dodsfall.Saksbehandling.Kodingstatus	
xxxxMedcod		
MedCod.CertificateKey	Dodsfall.Uuid	Parses fra base64 til Guid og matches med dødsfall i Dår
MedCod.LineNb	Dodsfall.Klassifisering.Arsak[x]	Brukes til å mappe en enkelt MedCod i Iris til Arsak i Dår. 0 = ArsakA, 1 = ArsakB, 2 = ArsakC, 3 = ArsakD, 4 = ArsakE, 5 = Arsak Medvirkende
MedCod.TextLine	Dodsfall.Klassifisering.Arsak[x].Tekst	
MedCod.CodeLine	Dodsfall.Klassifisering.Arsak[x].CodeLine	

	MedCod.IntervalLine | Dodsfall.Klassifisering.Arsak[x].VarighetTekst |
Pre-conditions	
Post-conditions	
Main flow	1. Saksbehandler henter Lot-liste (liste over dødsfall basert på brukernavnet til saksbehandler og status «Final» fra Iris). 2. Snapshot tas av hvert dødsfall som skal tilbakeføres 3. For hver Lot i listen hentes korresponderende dødsfall opp fra databasen: a. Hele eksisterende Klassifisering på eksisterende dødsfall overskrives med ny klassifisering som tilbakeføres fra Iris. b. Endringer gjort i Iris angående operasjon tilbakeføres. c. Endringer gjort i Iris angående Ulykke tilbakeføres. d. Status på dødsfallet i Dår settes til «Tilbakeført» eller «Tilbakeført_Final» om underliggende dødsårsak er satt. e. Lot slettes 4. Behandlingsresultat med antall tilbakeførte dødsfall returneres til saksbehandler.
Alternate flow	
Exceptional flow	
Issues/Notes	- Kø: Mappes fra Iris til kø i Dødsfall saksbehandling. Burde ha samme Iris-konfig som i produksjon. - Statuser på dødsfall: Tilbakeført, Tilbakeført_Final (Sistnevnte dersom det eksisterer en Underlying Cause) - ActivityCode og PlaceOfOccurance: Dersom ulykke (Manner of death 2) mappes disse i Dår ved tilbakeføring. ISSUE: Dersom AC og PoO er satt ved en annen Manner of death (f.eks. mord/selvmord) kan det ikke tilbakeføres da Sted/Aktivitet er i Ulykke-entitet i Dår. - Tilbakeføring av FreeText. Dette feltet fylles ved overføring til Iris når verdier er vanskelige å mappe. - ExternalFreeText? Feltet under ActivityCode og PlaceOfOccurance? Det er et datafelt i Iris, men vises ikke i Gui.
	-

11.10 The state diagram

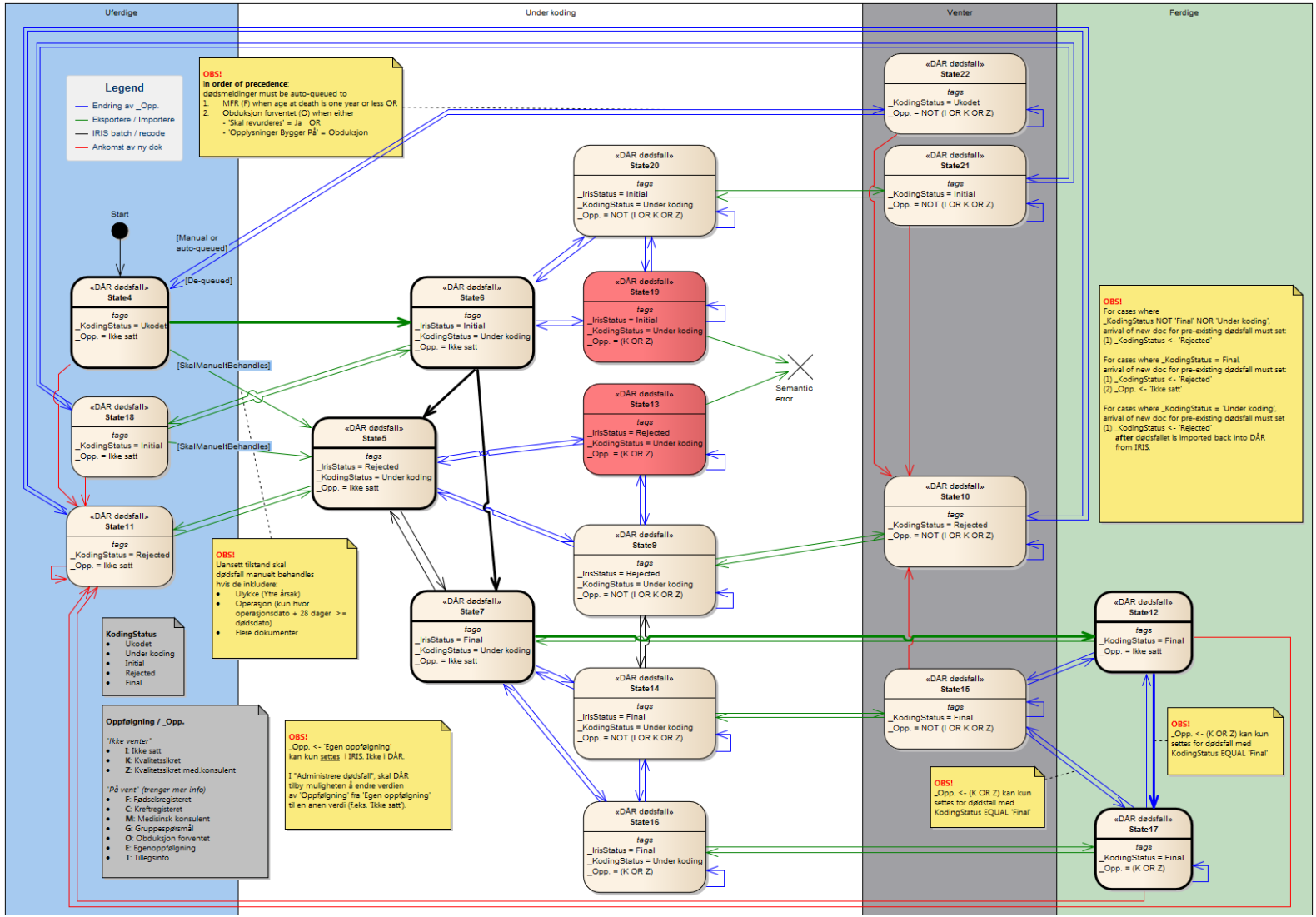

Figure 48 The CDR state diagram (from internal documents presented in Table 4)

11.11 The high-level sequence diagram

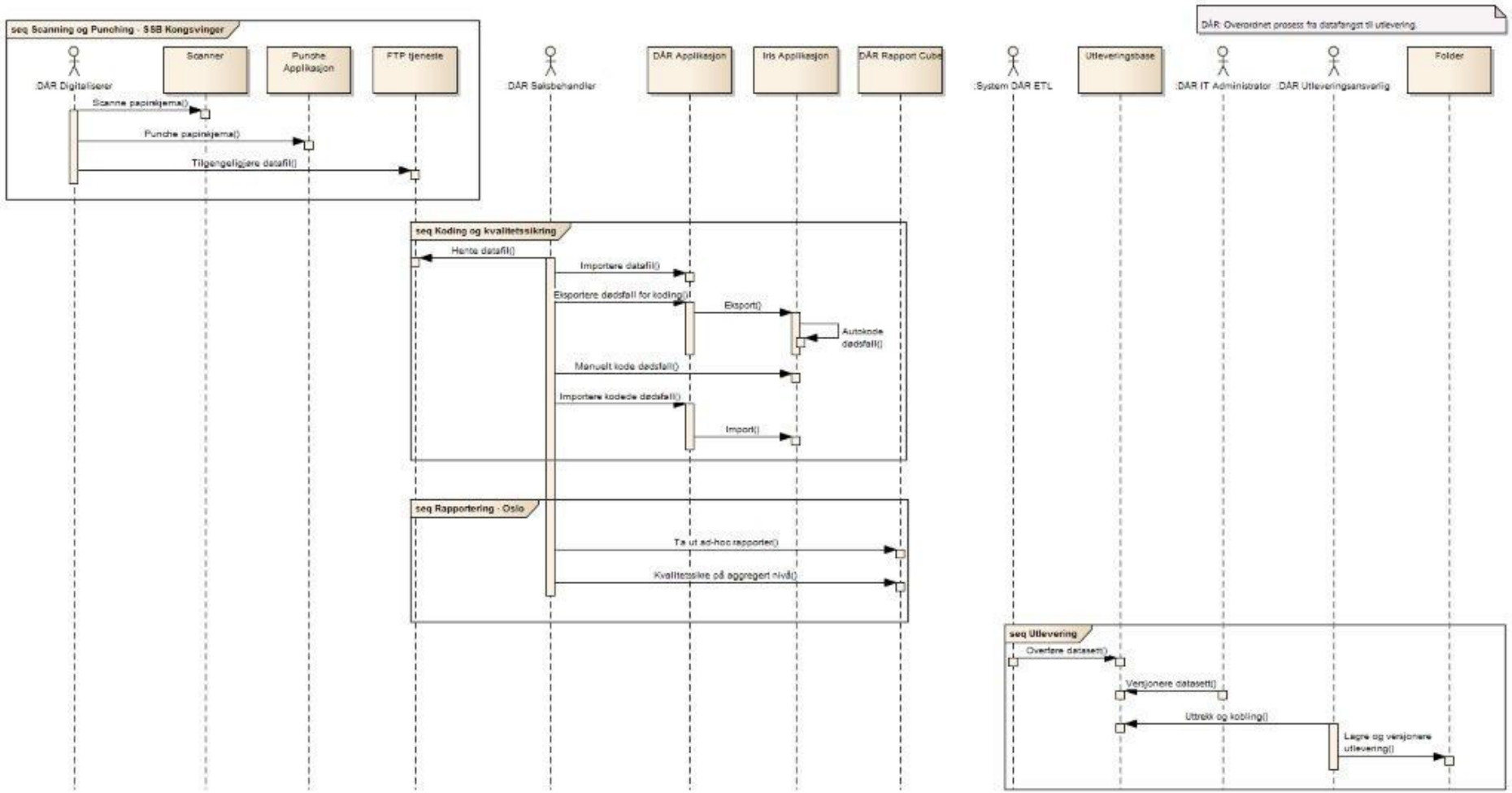

Figure 49 The high-level sequence diagram of the CDR workflow (from internal documents presented in Table 4)

12 References

1. **M. dos Santos Soares, D.Souza Cioquetta.** Analysis of Techniques for Documenting User Requirements. *Computational Science and Its Applications – ICCSA 2012.* Brazil : Springer Berlin Heidelberg, 2012.
2. **I. Sommerville.** *Software Engineering, 9th edn.* s.l. : Addison-Wesley, Essex, 2010.
3. **The Standish Group.** *CHAOS Chronicles v3.0. Tech. rep., The Standish Group.* 2003.
4. Chaos report. [Online] [Cited: 09 10, 2013.] http://www.projectsmart.co.uk/docs/chaos-report.pdf.
5. **A. Amaechi, S. Counsell.** Use cases in requirements capture - Trends and open issues. *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference, pp.137,142, 25-28.* 2012.
6. **N. Condori-Fernanz, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, O. Pastor.** A systematic mapping study on empirical evaluation of software requirements specification techniques. *IEEE, pp. 502-505.* 2009.
7. Helseregister loven. [Online] [Cited: 10 04, 2013.] http://www.regjeringen.no/nb/dep/hod/dok/regpubl/otprp/2006-2007/otprp-nr-52-2006-2007-/5/1/4.html?id=463694.
8. Statics Norway. [Online] [Cited: 03 10, 2014.] http://www.ssb.no/.
9. **N. Maiden, S. Robertson.** Developing Use Case and Scenarios in the Requirements Process. *ICSE'2005, pp. 561-570.* 2005.
10. **Narasimha Bolloju, Sherry X.Y. Sun.** Benefits of supplementing use case narratives with activity diagrams—An exploratory study. *Journal of Systems and Software, Volume 85, Issue 9, Pages 2182-2191, ISSN 0164-1212.* 2012.
11. Activity diagram. [Online] [Cited: 01 14, 2014.] http://en.wikipedia.org/wiki/Activity_diagram.
12. Dødsårskasregisteret. [Online] http://www.fhi.no/eway/default.aspx?pid=240&trg=Main_6664&Main_6664=6898:0:25,7838:1:0:0:::0:0.
13. Dødsårsaksregisteret (ved SSB). [Online] [Cited: 03 20, 2014.] http://www.ssb.no/en/dodsarsak/.
14. IEEE. [Online] [Cited: 03 19, 2014.] http://www.ieee.org/index.html?WT.mc_id=hpf_logo.
15. **Requirements Engineering qualifications board.** *REQB Certified Professional for Requirements Engineering version 1.3.* 2011.
16. IEEE standard. [Online] [Cited: 02 19, 2014.] http://www.computer.org/portal/documents/82129/160549/IEEE+29148-2011.pdf.
17. **Suzanne Robertson, James C. Robertson.** *Mastering the Requirements Process.* s.l. : Addison-Wesley, 2012.
18. **I. Jacobson.** *Object-Oriented Software Engi-neering - a use case driven approach.* s.l. : Addison-Wesley, 1992.
19. **A.Cockburn.** *Writing effective use case.* s.l. : Addison-Wesley, 2001.
20. **D. Kulak, E. Guiney.** *Use Cases: Requirements in Context.* s.l. : Addison-Wesley Professional, 2004.
21. **F. Armour, G. Miller.** *Advanced Use Case Modeling.* s.l. : Addison-Wesley, 2001. 978-0201615920.
22. **D. Sinnig, H. Javahery.** Mastering use cases: capturing functional requirements for interactive applications. *EICS '10 Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems p.373-374.* 2010.

23. **B. Bernardez, A. Duran, M. Genero.** Metrics for Use Cases, chapter in Metrics for Software Conceptual Models, pp.59-98. 2005.
24. **L. L. Constantine, L. A. D. Lockwood.** *Software for Use: A practical guide to the Models and Methods of Usage-Centered Design.* s.l. : Addison-Wesley, New York, 1999. 0201924781.
25. UML Use Case Diagrams: Tips and FAQ. [Online] [Cited: 02 13, 14.] https://www.andrew.cmu.edu/course/90-754/umlucdfaq.html.
26. **D. Sinnig, R. Frédéric, C.Patrice.** Use cases in practice: a survey. Proc. CUSEC 5, 2005.
27. **Behrens, Dean Leffingwell with Pete.** *A user story primer.* s.l. : Leffingwell, LLC., 2009.
28. **M.Cohn.** *User Stories Applied: For Agile Software Development.* s.l. : Addison-Wesley Professional, 2004.
29. Agile development. [Online] [Cited: 02 10, 14.] http://en.wikipedia.org/wiki/Agile_software_development.
30. State machine diagram. [Online] [Cited: 01 14, 2014.] http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm.
31. State machine diagram. [Online] [Cited: 01 15, 14.] http://www.sparxsystems.com/resources/uml2_tutorial/uml2_statediagram.html.
32. state machine diagram. [Online] [Cited: 01 12, 14.] http://www.uml-diagrams.org/state-machine-diagrams.html.
33. Sequence diagarm. [Online] [Cited: 01 23, 14.] http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/.
34. Sequence diagram. [Online] [Cited: 02 24, 14.] http://www.ibm.com/developerworks/rational/library/3101.html.
35. seqTut. [Online] [Cited: 01 12, 14.] http://csis.pace.edu/~marchese/CS389/L9/Sequence%20Diagram%20Tutorial.pdf.
36. **M. Fowler.** *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition.* s.l. : Addison-Wesley Professional, 2003.
37. CDR regulations. [Online] [Cited: 02 10, 2014.] http://lovdata.no/dokument/SF/forskrift/2001-12-21-1476.
38. Sentrale helseregistre. [Online] [Cited: 11 01, 2013.] http://www.fhi.no/helseregistre/om-helseregistre/sentrale-helseregistre.
39. Modernisering av sentrale helseregistre. [Online] [Cited: 02 12, 2014.] http://www.fhi.no/eway/default.aspx?pid=239&trg=Content_6465&Main_6157=6261:0:25,8347&Content_6465=6178:89211::0:6268:4:::0:0.
40. ICD codes. [Online] [Cited: 02 11, 2014.] http://www.who.int/classifications/icd/en/.
41. NIPH (fhi.no). [Online] [Cited: 10 04, 2013.] http://www.fhi.no/eway/?pid=240.
42. Helseregistere i FHI. [Online] [Cited: 12 02, 2013.] http://www.fhi.no/helseregistre/om-helseregistre.
43. Avdeling for IT og e-helse (FHI). [Online] [Cited: 03 01, 2014.] http://www.fhi.no/eway/default.aspx?pid=239&trg=List_6212&Main_6157=6261:0:25,8353&MainContent_6261=6464:0:25,8888&List_6212=6218:0:25,8889:1:0:0:::0:0.
44. WHO (World Health Organization). [Online] [Cited: 02 05, 2014.] http://www.who.int/en/.
45. EuroStat. [Online] [Cited: 01 14, 2014.] http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home/.
46. TFS. [Online] http://en.wikipedia.org/wiki/Team_Foundation_Server.
47. Enterprise Architect. [Online] [Cited: 01 21, 2014.] http://en.wikipedia.org/wiki/Enterprise_architecture.

48. Sharepoint. [Online] [Cited: 01 21, 2014.]
http://en.wikipedia.org/wiki/Microsoft_SharePoint.
49. **P. Runeson, M. Höst.** Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering 14:131–164.* 2009.
50. **Zelkowitz, M.V. and D. Wallace.** *Experimental Validation in Software Engineering. Information and Software Technology, 39(11): p. 735-743.* 1997.
51. **Yin, R.K.** *Case Study Research Design and Methods.* s.l. : Sage Publications, 2003.
52. **HK. Klein, MD. Myers.** A set of principles for conducting and evaluating interpretative field studies in information systems. *MIS Q 23(1):67–88 doi:10.2307/249410.* 1999.
53. **Wohlin, C., M. Höst, and K. Henningson.** *Empirical Research Methods in Software Engineering.* 2003.
54. **Yin, R. K.** *Case study research: Design and methods.* s.l. : 4th edition. Thousand Oaks, CA: Sage., 2009.
55. Open office community forum. [Online] [Cited: 08 10, 2013.]
https://forum.openoffice.org/en/forum/viewtopic.php?f=6&t=66021.
56. Use cases blog. [Online] [Cited: 08 10, 2013.] http://alistair.cockburn.us/Use+cases.
57. En god kravspec - grunnmuren i et vellykket webprosjekt. [Online] [Cited: 03 05, 2013.]
http://blogg.digitroll.no/bloggindex.aspx?pageId=961&articleId=3297&news=1.
58. **S. Easterbrook, J. Singer, M. Storey, D. Damian.** Selecting empirical methods for software engineering research. Guide to advanced empirical software engineering. s.l. : Springer London, 285-311., 2008.
59. Ethnography. [Online] [Cited: 03 01, 2014.]
http://alarcos.esi.uclm.es/ease2012/Resources/documents/Sharp-Abstract.pdf.
60. Exam Diff Pro. [Online] [Cited: 01 010, 2014.]
http://www.prestosoft.com/edp_examdiffpro.asp.
61. IRIS. [Online] [Cited: 11 18, 2013.]
http://www.dimdi.de/static/en/klassi/koop/irisinstitute/about-iris/index.htm.
62. ETL. [Online] [Cited: 02 05, 2014.] http://en.wikipedia.org/wiki/Extract,_transform,_load.
63. **N. Boulila, A. Hoffmann, A. Herrmann.** Using Storytelling to record requirements: Elements for an effective requirements elicitation approach. *Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games (MERE), 2011 Fourth International Workshop on, pp.9,16, 30-30.* 2011.
64. State machine diagram. [Online] [Cited: 02 03, 14.]
http://publib.boulder.ibm.com/infocenter/rsdvhelp/v6r0m1/index.jsp?topic=%2Fcom.ibm.xtools.modeler.doc%2Ftopics%2Fcstated.html.
65. **S. Lilly.** Use case Pitfalls: Top 10 Problems from Real Projects Using Use cases. *Proceedings of TOOLS USA '99, IEEE Computer Society.* 1999.
66. Why use cases are not enough. [Online] [Cited: 01 15, 14.]
http://www.processimpact.com/more_about_reqs_book/chapter_11.pdf.
67. What is wrong with use cases. [Online] [Cited: 02 06, 14.]
http://www.jacksonworkbench.co.uk/stevefergspages/papers/ferg--whats_wrong_with_use_cases.html.
68. Use Cases: the Pros and Cons. [Online] [Cited: 01 05, 14.]
http://www.ksc.com/article7.htm.
69. Use Cases: Good & Bad. [Online] [Cited: 01 07, 14.]
http://alinement.net/component/content/article/35.
70. When use cases are not enough. [Online] [Cited: 01 15, 14.]
http://www.processimpact.com/more_about_reqs_book/chapter_11.pdf.

71. Regjerningen.no. [Online] http://www.regjeringen.no/nb/dep/hod/dok/regpubl/otprp/2006-2007/otprp-nr-52-2006-2007-/5/1/4.html?id=463694.
72. ACME. [Online] [Cited: 11 18, 2013.] http://en.wikipedia.org/wiki/Mortality_Medical_Data_System#ACME.

List of figures

List of tables