Digital Review
Final Project Report

Final report on the project of developing the Digital Review Application

Bachelorproject IN3405

SKION / DCOG

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
June 30, 2009

Exam committee consisting of ir. M. Tijsma, Dr. E.A. Hendriks, ir. B.R. Sodoyer

Preface

At the Technical University of Delft, students complete their Bachelor by doing a bachelor project. For this project, students can choose to perform this project externally, for example as an internship at a company.

This thesis, written by Joël van den Berg, Joep Weijers en Bob Westerveld, is the result of the development of a proof of concept application at SKION[1] from April to July 2009.

We like to thank Emile Hendriks, for being our supervisor for the Technical University of Delft and introducing us to SKION. We would also like to thank Martijn Tijsma for being our mentor during the project and also design the interface for us. If we needed anything, Martijn was there to arrange it for us.
We further like to thank Kars Meijboom and Henk-Jan van Wijk for providing support on how to connect to the SAP en the ProMISe database respectively. We also would like to thank Hanneke de Ridder for providing workspace and necessary tools. Finally we would like to thank Miriam Tros-Batist and De Witte Condor BSc, for providing support on SAP and contacting the HaGa hospital endlessly.

Joël van den Berg
Joep Weijers
Bob Westerveld

[1]Stichting Kinder Oncologie Nederland

Summary

SKION is a foundation that facilitates research of children with cancer. The research consists of investigating slides and other material gotten from the treating hospitals at SKION. The results of these investigations are saved into a database, while other information gathered by SKION through forms is stored in another database. Four times a year, Dutch oncologists from all around the country meet in real life to review several slides. This is a very inefficient way of working, since all participants have to come to a central location. As a solution, Martijn Tijsma introduced the possibilities of a Digital Review[3]. This requires an application that stores forms and combines data from the two databases and merges them on screen for all participants of the review to see. The goal of this project was to create a proof that such an application can be build (Proof of concept).

We analyzed the problem by asking SKION what functionality they would like to see in the application and what functionality we thought was necessary for the application. This information is gathered in the Requirements Analysis Document, appendix A. During the analysis of the requirements we found that we needed some way of making sure that the one responsible for signing the current CRF[2] was really the one signing the digital forms and that the forms couldn't be changed after signing. For this we came up with the Digital Signature, appendix F. By analyzing the requirements, we made a global outline of how the application could be designed and wrote this down in the Architectural Design Document, appendix B. Next we made the Implementation Plan, appendix C and Test Plan, appendix D, which describes how we wanted to implement and test our application during the project. Finally, we made a MoSCoW document, appendix E which shows the priorities during the project.

The result is a working application, yet, it does not fully meet the requirements defined in the RAD. The alpha test also revealed some problems, bugs, with the application. This is why we recommend further development of the application. Since the server used for the application was supplied by ourselves, we also recommend setting up a server with the application and a database on it. Furthermore, for a fluent continuation of the application, a Computer Scientist should be added to the workgroup. In conclusion, we did manage to make a proof of concept. We did not manage to implement all the goals defined in the MoSCoW document, but we did manage to test all the functionality to complete the 'Must haves' as stated in the document.

[2]CRF: Case Report Form

Contents

Chapter 1
Introduction

1.1 SKION

SKION[1] is founded in 2002. SKION has come from it's precursor SNWLK[2] which was founded in 1972. SKION facilitates classifying diagnosis and/or quality control of cancer at children, by providing a reference laboratory for the detection of leukemia and lymphomas. Secondly SKION has an in-house trial bureau which registers cancer at children and SKION also define protocols. SKION organizes Central Review meetings. At this meetings child oncologists and specialists from different hospitals around the Netherlands come together to inspect slides, reflect on them through discussion and give advise on the treatment they think is best. In order to organize these Central Review meetings, a lot of hard copy documents have to be send to the participating actors. During the review several more documents are filled in by the specialists. Afterward, the results are processed at the SKION central bureau in The Hague and result documents are send to pathologists and doctors.

1.2 Central Review to Digital Review

Because SKION wants to improve the effectiveness of reviewing patients and also want to improve the quality of the reviews, a workgroup has been formed to investigate whether it is possible to review patients using video conferencing and a digital application. The workgroup's work resulted in a project outline to implement a Digital Review concept.[3]

1.3 Digital Review application

Our contribution to the Digital Review concept is the development of an application which will contain the information available of patients which are to be reviewed. In this way, review participants are able to prepare the review more efficient, all the information will be kept in one central place, and information will be up-to-date at all times.

1.4 Readers management

First we will describe the problem and the analysis of this problem in chapter 2. A detailed version of the analysis can be found in Appendix A. Next we will give an overview of the design of the application, which is fully described in the Architectural Design Document (Appendix B), in chapter 3. The implementation of the application is described in chapter 4. The conclusion of the project can be found in chapter 5. In chapter 6, the problems we faced during the project are described. Chapter 7 gives an overview of what we thought about the process of the project. Finally we present our recommendations in chapter 8.

[1]In English: DCOG = Dutch Childhood Oncology Group
[2]Stichting Nederlandse Werkgroep Leukemie bij Kinderen

Chapter 2
Problem, Analysis and Goals

In this chapter, a brief overview will be given about the problem, the analysis and the goals of this project. For a full description, the Requirements Analysis Document (RAD), appendix A, can be read.

2.1 The Problem

The problem of SKION was that the current manner of doing a review was to troublesome for the people involved. Letters had to be sent to datamanagers so they could fill it in and sent the letter back to SKION. Futhermore, the people involved in the review had to be present at a central location and the paperwork had to be printed out and handed over to these people. These problems needed a solution, so Martijn Tijsma was asked to do research on a so called Digital Review. During this Digital Review, the people involved no longer need to be at a central location, but rather at their local hospitals and they would communicate through web conferencing. This new approach however required some software that would make necessary data present for each of the ones involved during the review.

2.2 Analysis

For this solution to work however, an analysis had to be done on the information required and the people involved. In the old process, data was gathered by hand from two different databases and then merged into a single sheet. Next to this data, some forms had to be filled in by datamanagers and pathologists which were also used during the review but also inserted into a database. One could easily see that these several processes could easily be joined into a system. This system however had some criteria. For one, it had to be accessed from different locations and had to transfer patient data. This required some extra measures on security defined by the NEN[1]. Also, it had to be up and running at all times, so that the forms could always be accessed and filled in. Furthermore, the system needed to give different rights to different people, because, for example, a datamanager isn't allowed to see anything but the pre-CRF of his own patients.

2.3 Important requirements

As can be read in the RAD document, requirements for the program were gathered. Some of the most important requirements are the functional requirements and the non-functional requirements, which are listed below.

Functional requirements

- Patient information from both the SAP and ProMISe database need to be shown on the same screen.
- Patient information must be filled in through the system and saved in the ProMISe database.

[1]NEderlandse Norm, the institution for setting norms and standards

- People involved need to be notified on the progress of a cenral review.
- Central reviews need to be organised through the system.
- Distinguish different types of user with different privileges.
- Authentication is required to access the system.
- Several types of user need to be able to sign forms using a digital signature.
- There has to be the possibility of adding new types of review to the system.
- Information that could not be processed by the application need to be stored.

Non-functional requirements

- Documentation of the application needs to be extensive, because the software is a fundament for future development of this project.
- The performance of the system is required to be working as smooth as an average web page. Even when the load on the application is high.
- Several users need to be able to access the same data at the same time.
- The application has to be available at all times.

For a more detailed view on what requirements are needed, we refer to the RAD, appendix A.

2.4 Goals

The goal of this project is to make a system that proves that a system, meeting all the requirements defined during the analysis, can be build and can be used in the way that SKION intended.

Chapter 3
Evaluation of the design phase

This chapter will briefly describe what we did during the design phase of the project and how we used it to take advantage of it in the next phases.

3.1 Documentation

A big part of the design phase was writing documentation. This was actually everything we needed to implement the application. Making the documents was troublesome and required some time, but they were good use for us when trying to understand and define the problem. This, in the end, made the implementation go more smoothly.

3.2 Model, View, Controller

We designed the system by the Model, View, Controller (MVC) approach [1]. This basically means that all the objects that would contain data are in the model part of the system. When the classes in the View package needs something from these models, these classes ask the controller to do something with the models so they get the information they require. This approach produces a well organised system, which is easy to maintain and alter and thus easy to expand with more functionality.

3.3 Modules

We needed to make some parts of the application extendable. For example, SKION mentioned that in the future, other kinds of research, also making use of a certain review procedure, consider using the application. These reviews all follow a different pattern on how the review is set up, and the parties that are involved during a review.

3.4 Test Driven Development

As stated in the RAD, we implemented the application using Test Driven Development, which means that for every functionality in the system, we wrote a test function. After writing the testcase we would implement the functionality, test it with the test function and repeat this till all the test functions passed. To automate these tests, we used PHPUnit, an application to unit test a program. A unit test is the test of a very small part of the system, for example a class. The result is a well tested and reliable system. Unfortunately we could not test everything with this approach. For example the classes in the View package were tested by clicking through the application. Another part was the integration testing where all the classes were put together. This also had to be done by hand. These latter two testing methods are also common when programming without unit testing, but unit testing increases the traceability of small errors like typo's or misplaced operators.

3.5 The new database

Our program requires a database to store the users in the system and some more system-specific entries. We had two options, to store this info in either SAP or ProMISe or to set up our own database. So we set up a new MySQL database. We installed this database on the same computer that hosted the application, so the application would have immediate access to this database. This way, we could also easily adapt the database to the applications needs, for example the storage of user data, privileges and coupling patient information. This separates the patient and the system data, leaving the maintenance of the existing databases to the ones responsible for it, and giving us, the programmers, full freedom to work and adapt a database specifically for the program's needs.

3.6 Planning

Before the project started, we made a global planning which gave a good outline for the project. The first half of the project, we planned to get familiar with the problem by gathering requirements. When we were finished gathering requirements, we presented them to SKION during a midterm evaluation. This way we could verify them and make some changes to them. Next we planned the implementation phase in compliance with the milestones stated in our implementation plan, and also briefly listed here.

- Database connectivity
- Authentication
- Patient overview
- Patient detailed view
- Review specific functionality
- User interface
- Email
- Logging
- Documentation

SKION could use this to see how far we were with the implementation of the system and it gave us a good guide on what had to be finished on what time. The planning as followed can be found in the appendix G. Unfortunately, near the end, our planning had to be extended a bit. The alpha test, for example, was planned a week later, because the implementation wasn't ready. Fortunately though, this extension was already considered while making the planning at first, thus no real delay occurred.

Chapter 4
Implementation

This section will briefly describe the way that the system was implemented and the considerations we made during the implementation.

4.1 Focus during the implementation

During the project we had a clear focus on the hardest and possibly the most important bit of the system, namely the connectivity with the two existing databases. This took some time meeting with the ones responsible for the databases and some puzzling with the methods offered by the layers on top of the databases. A commonly accepted way of interaction was through web services, which will be discussed later on. Since the gathering of the data was a big part of the project, we decided that this was the first that needed to be working. Also, since we had several meetings with SKION, we needed to show them that we could retrieve from the data.

Another big part of the application was the layout. Martijn was the one responsible for the layout, since he already did some research on the user interactivity. Also, in order to actually show what we could retrieve from the data, we needed to present it.

4.2 Web services

As said before, we interact with the databases through the use of webservices. This required some technical details about the databases as well as information needed to get authenticated and be able to talk with the web service. The support in obtaining this information was well organized, but the implementation itself took time, since it was not clear how PHP would talk with the webservices and what responses we would get. In figure 4.1 an overview of the SAP and ProMISe webservices can be seen. A user asks for information, the application then decides where this information has to come from and asks the corresponding web service for the information. The web service which is connected with the database gets the data from the database (after some necessary checks) and returns the data to the application, which then shows it on the screen.

We are able to retrieve the data from both ProMISe and SAP. And we tested writing to ProMISe, but our authentication credentials are not valid anymore. So even though we were able to write, it is not present in the proof of concept. However, the rest of the implementation facilitates writing to ProMISe, so it is really just a matter of getting the correct connection info.

4.3 Alpha test

Since the goal was to make a proof of concept, it was necessary to have people interact with the system. This gives a good overview of how and why people would or would not use it and is an important way in finding bugs. Three people at SKION have participated in an alpha test [2], which can be found in the appendix H. These tests have revealed several bugs in the primary use of the program, for example some assertion errors occurred in model classes. We directly fixed these bugs, because they affected the primary functionality of the application. Furthermore some other issues were raised mostly concerning the user interface. We have listed these improvements for future development on the application.

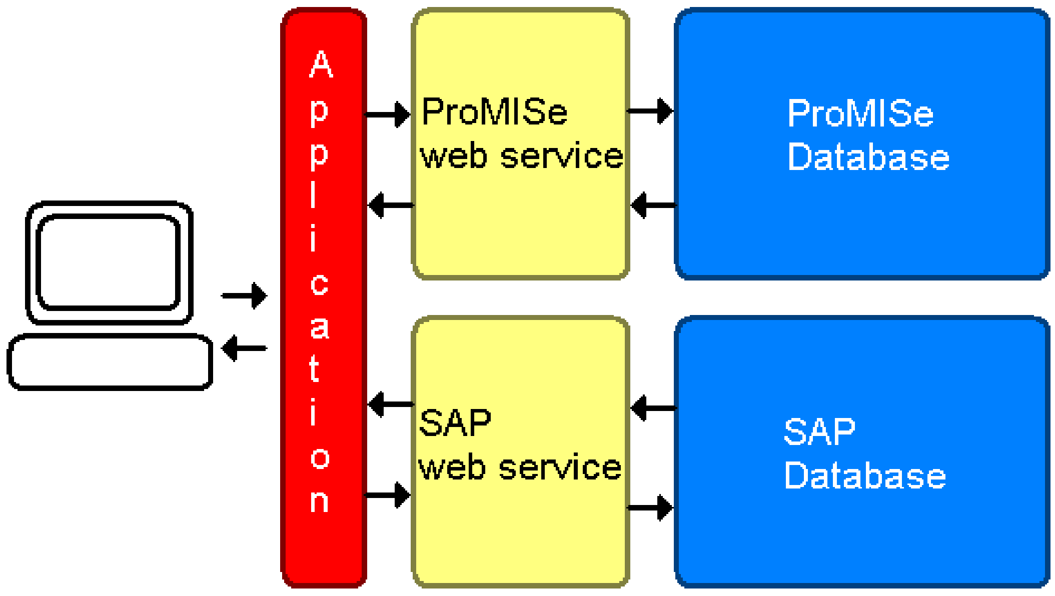


Figure 4.1: An overview of how the web services work.

4.4 Tools

The tools used for the implementation were Eclipse (with several add-ons[1]) and PHPUnit (for unit testing), as well as Apache and PHP5 for testing the implementation. For setting up the SQL database, we used phpMyAdmin, which basically is a PHP script for maintaining a SQL database on the host.

4.5 Authentication

The authentication method of the application is implemented using PHP sessions. Passwords are stored in SHA1 hashed strings in the database, as required by the NEN. This way, should anyone get access to the database, passwords can never be decrypted. If this method is used over an SSL connection, we completely adhere to the NEN for medical applications.

4.6 Email

The application is equipped with email functionality, which can send out emails for password recovery or notifications to actors in the review process. However, SMTP port 25 must be forwarded to the server for this functionality to work. For our proof of concept, with the server at the HaGa Hospital, this was not possible, thus this functionality cannot be shown. However, emails were sent when the server stood at Bobs home, so the functionality is working. This means that all kind of emails can be send, fulfilling the requirement to email.

4.7 Digital signature

Another important feature of the application is the digital signature. When implementing this, we followed the recommendations we proposed for the digital signature in the digital signature

[1]Add-ons used: PHPeclipse, Subclipse, Texlipse, ArgoUML

research document, appendix F. When a new User is added to the system, an RSA key generator generates the public and private key pairs. Signing a Form generates a signature based on the date of signing, the signing user and all the values filled into the form. When viewing a signed form, the application tests for authenticity. If the form is invalid, i.e. something has been changed somehow, a warning message tells that the form is not authentic.

4.8 State machine

When we were presented the flow of events for the BMF review, which can be found in the RAD (appendix A), it looked very similar to a Finite StateMachine we met in some courses. So we decided to model the flow of events with a StateMachine. After implementing an abstract StateMachine model, we made a concrete StateMachine for the BMFReview. This StateMachine is depicted in figure 4.2. We enabled the StateMachine to work with PHP functions. This way, transition conditions can be checked at run-time and upon entering a State an email can be send.

Although our implementation from the BMFReview is not complete, the transitions between states we did implement, have proven to work. For example, when a patient is added to a review and some basic data has been filled in, the StateMachine sends the datamanagers an email.

The StateMachine is a proven way of handling the flow of events in a Review, so for extending the application with more types of review, one can easily model the flow of events of that particular review. It might even be possible to have a graphical editor for the StateMachines, so that the programmer is complete left out the process of adding a new type of review to the application.

4.9 Forms

Continuing on the extendability of the application, we created a dynamic Form class to build forms. For each form can be chosen where the initial data can come from and where to write data entered in the form to. For example, the PreCRF can be loaded from ProMISe, if the user is already added to the system in the current way. But because we can't write to ProMISe now, we write the info entered in the PreCRF to our own database. In the specification of a Form is also set whether or not the form should be signed with a digital signature.

Like the StateMachine, these forms are very generic, and allow for the implementation of a graphical editor. So when a new type of review, users (specifically the review manager) are able to add and edit forms. However, these graphical editors are a feature that is nice to have later on. A requirement of the application is easy extendability. These graphical editors are a part of that, but noting that they are relatively easily created, proves that we fulfill the requirement.

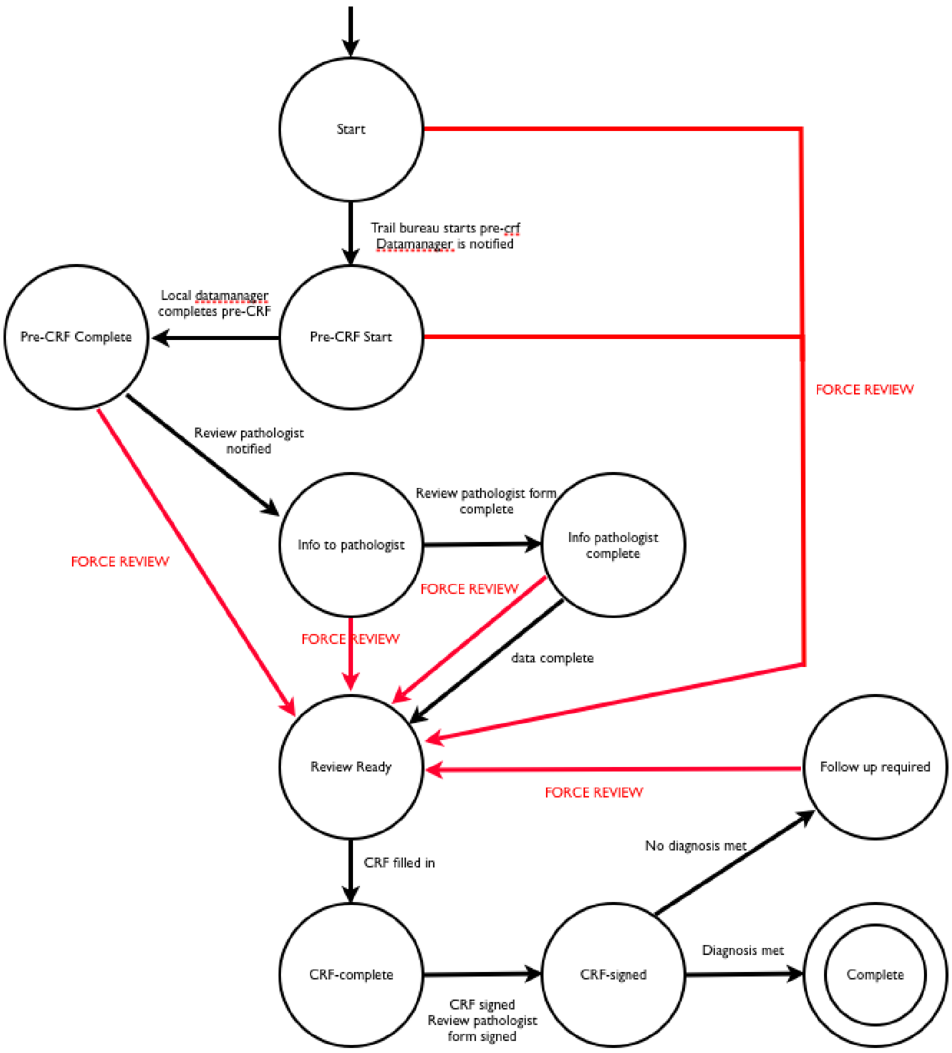


Figure 4.2: Implemented BMFReview StateMachine

Chapter 5
Conclusions

The goal of this project was to create a proof of concept for the digital review application. Looking at the application we have created, we can conclude that this concept is technically possible. We say technically possible, because we programmers can not assure anything on the acceptation of the application by the end users. We can only do our best to create an application as user friendly as possible. However, since we had a lot of technical challenges, the user friendliness was not our main focus. If our program is taken to a higher stage of proof of concept, namely testing with all the people that will actually have to use it, it will be wise to change the View (note that only the View has to be changed) to a more user friendly version.

We have achieved this goal by following the software development process we have learned in the past years. The application had to be build from scratch, so we the most difficult challenge was to make a connection with the different databases to get the information needed. However, we managed to read information out of the databases. We have also build functionality to model the flow of events of a review, by the example of the BMF MDS review.

The user interface is designed by Martijn Tijsma and so a well looking interface was not the focus during the project. However it turned out to be an important feature of the application. By connecting the user interface to our source code, in other words, connecting a new View with the Model and Controller part of the system, revealed some flaws in our implementation of the MVC architecture. Some time was needed for fixing these bugs, but we managed to stay on schedule.

We had several moments of feedback on the application. Besides continuous tips of Martijn, we performed an alpha test. What came forward as lacking or incomplete can be found in appendix I. Furthermore, we held a presentation for the workgroup of the digital review and during this presentation we received feedback from the workgroup. Some important issues are listed here:

- A reminder should be send to the datamanagers if the PreCRF has not been filled in a certain number of days before the review. This can be achieved by modifying the StateMachine describing the BMF MDS review and adding a function to the BMFFunctions class.
- When a signature is generated for the PreCRF, this will disable the editing of the PreCRF. However, the data will be stored in the ProMISe database and can thus be edited in ProMISe, invalidating the signature. This is a pretty important point to think about.
- Due to not being able to write to ProMISe yet, data entered in the proof of concept program, is stored in our own database. This leads to some information from the dynamic forms being incorrect, as they expect the data to be stored in and retreived from ProMISe. We created a workaround for the PreCRF (if a form is filled in, retrieve it from our database). But this does not work correctly for the forms to show patient info in the detailed patient overview screen (i.e. the physical examination tab). This problem will be eliminated when we can write to ProMISe, but the problem might arise again if a form should be retrieved from another location than it should be stored, so we note it here.
- Lastly, when viewing the Logs, the workgroup noticed that the administrator was able to see that somebody edited a form, but that is was not clear whether or not the person also signed it. We agree that this might be valuable information, so a log entry should also be written when the form is signed.

Apart from this all, we ourselves have a list of things that must be done before the system is really operational as described in the design documents:

- Writing forms to ProMISe. This is currently not possible.
- Some improvements to the user friendliness and intuitive operations like the placement of submenu's and adding patients to a review.
- Better integration of the StateMachine and visualisation of the StateMachine in the interface. It is present, but is not used as much as it was planned.
- More thorough testing. Due to time constraints, some classes are not or not properly tested.
- Stricter contracts in functions. We specified some preconditions for functions, but we could also extensively specify and assert postconditions.
- An SSL connection to ensure the safety.
- Storing the StateMachine in the database. It is now stored in a file, unlike the rest of the Model.
- Move some functions to a library class. Some functions that are used a lot in a similar form can be abstracted out of the code and into a library.
- Even more modularity of the code. We have hardcoded several references to the BMF MDS review in the View. Ideally, the entire View would be dynamic, retrieving all reviews from the system.
- Editors to create StateMachines and Forms. A part of the specification says that the application should be easily extended. This has to be done by programmers now, so creating an editor allows normal users to add new types of review.
- Increase the performance of the application by speeding up the retrieval of data through the web services or by using a cache.

Chapter 6
Process

The development of the application process was carefully planned as the reader might have seen above. In this chapter, we will point out some problems we faced during the project of which we think were critical in the continuation of the project.

6.1 ProMISe and SAP database

In the development phase we visited two people which are involved in the development and maintainance of the ProMISe and the SAP databases. They ensured us that we could communicate with the databases by using webservices. This sounded very trivial and at first we thought of it very lightly. At the implementation phase however, it turned out that webservices are not trivial at all and many hours of work were required to make them work. Also a lot of support from the database management instances, at Leids Universitair Medisch Centrum and HaGa hospital, was needed to get any information out of the databases. To get some information out of ProMISe, all was set up relatively easy and worked fine except for writing to ProMISe. To connect with the SAP database, however, was more difficult. It turned out that the SAP database was not accessible from outside the HaGa network. So to get a connection with this database we decided to put our server at SKION. When all connectivity with the databases was set up, another problem occurred, namely that relatively frequent execution time errors occur. These are indicating that one or both of the databases do not respond within thirty seconds. This is very annoying, but we decided to put this on the list to be fixed later, because more important functionality had to be build for the proof of concept.

6.2 Test driven development

Another problem came up, when the model of test driven development was not applied correctly. Sometimes, when we needed a piece of code quickly, this piece of code was not unit tested sufficiently or not even tested at all. If an error was in that piece of code, more time was wasted by finding this error than writing the test initially. Thus the lesson learned here is that carefully unit testing prevents wasting time on hunting bugs.

6.3 Subversion

The comments of the revisions at the Subversion server were not always entered. In this way, we could not always see what other people had contributed to the project and neither what the status of some piece of code was.

6.4 Server complications

When the project was in its last weeks, we thought it would be nice to access the server on which the application was developed from our homes or the Univeristy and also have the application send out emails. It seemed very easy, but it was not because system administrators at the HaGa hospital did not let us do it. They were arguing that someone might break in on the servers of the HaGa hospital. A solution was given by using VPN[1], but this had some consequences for the application. It is not able to send out emails, although the functionality is implemented.

[1]Virtual Private Network

Chapter 7
Evaluation

All involved people at SKION, especially the workgroup involving the digital review project, were very enthusiastic about our work on the project. During the midterm evaluation we presented our findings, models and ideas to the workgroup and they were impressed by the work we had done and complimented our fast understanding of the problem. It was very stimulating for us to work in an environment where people were always available to answer questions and were giving compliments. It really kept us going and it made us enjoy our project.

7.1 Planning

We had a pretty tight and realistic planning, that we only slightly adjusted along the project. We took an extra week to finish our design documents and we moved some milestones around in the implementation phase. The partition of the project in milestones had one flaw; one milestone (the review specific functionality) was large and was planned in a too short time span. This resulted in the Model not being completely ready when some Controller and View features were implemented and this had its influence on the code, for example the StateMachine is not as widely used in the program as originally intended.

7.2 Group

Our team consisted of three persons, all with different competences. This is a good thing, since a project like this calls for different tasks like implementing, testing, contact with external sources. However, sometimes the small size of the group required members to do tasks in which they do not excel. This resulted in some extra work. Considering the size of the project and the fact that it had to be build from scratch, might have called for an extra group member assisting in the programming.

Our mentor, Martijn Tijsma, was not only a mentor but also the fourth man in our project team. He solely designed and implemented the user interface of the application. Thanks to the Model, View and Controller implementation of the application, it was easy to connect his work with ours.

Although we had several informal meetings a week, we never had a structural planned weekly meeting. Maybe next time it is useful to plan a weekly meeting to talk about the status of the project and the things to be done in that week. We feel that the lack of these meetings in this project is not a major problem because of the small size of the group, but when the group size increases, it is certainly recommendable to plan weekly meetings.

7.3 Technical University of Delft

At the Technical University of Delft (TUD), there were no sufficient resources available for us to constructively work on the project. This was mainly concerning the implementation phase. The computers provided did not match the required functionality we needed, so we decided to take our own laptops with us. But then it turned out that we were not permitted to make use of the wired LAN connection of the TUD. Also the climate control of the room we were in was very bad, which resulted in very hot working days. Also, we needed a server to develop our code on. This seemed to be very difficult to arrange at the TUD, so we decided to turn on one of our own personal computers act as a server. As a last point of annotation, the TUD could not provide a subversion server for the development of our application. As a solution to the issues above, we decided to go five days a week to SKION to work on the project. This also resulted in direct contact with the people who could provide answers to our questions. For the SVN server, we signed up for an the SVN server at Origo [1].

In conclusion, we have enjoyed the project, it was a fun and grateful project to work on. If we would have to do it again, we would do the following things differently, to get more done in the reasonably short span of time:

1. Extension of the size of the project team
2. Go to SKION five days a week from the start on
3. Plan weekly meetings to talk about updates etcetera

[1] http://www.origo.ethz.ch/

Chapter 8

Recommendations

In this chapter we will present some recommendations for future development of the digital review application and also some overall recommendations which reflect on the workgroup and other projects.

First we think that the application can be improved on several points namely:

1. The points described in the conclusion, chapter 5.
2. In the detailed patient view screen, tabs are used to display information. In order to get a better performance and usability, it is a strong improvement to implement these tabs using AJAX[1]. Using this technology, instead of calling ProMISe three times in order to get a patients overview, a call will be made whenever the tab is opened, improving the performance.
3. Due to the slowness of ProMISe, a caching mechanism would be in place to drastically increase operation time in some parts of the system. This can be done, because no data will be changed often. This means that the data read from the databases could be temporarily saved so no call have to be made when reaccessing the same data.

Secondly, a server needs to be set up which will run the application in the future. It is important that the server which will run the application is within the HaGa hospital network in order to make a connection with the SAP servers. SKION already has a server in the HaGa serverroom, this one would be suited for the job. Third, we will strongly recommend to add some knowhow of webapplications to the workgroup. This implies that a programmer, or a computer scientist will join the workgroup in order to give advice and possibly to assist/judge the work of future programmers on the project. Furthermore, by the development of this proof of concept, and specifically the connection to both ProMISe as SAP, the so-called W-kaart can be developed. The W-kaart wants to combine the information from SAP and ProMISe and run queries on it. This poses a challenge, but is not undoable. This may even be a bachelor project for Computer Science students for next year.

[1]Asynchronous Javascript And XML

Appendix A
Requirement Analysis Document

Digital Review
Requirements Analysis Document

Analyzing the requirements of the desired system at SKION

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
June 23, 2009

Contents

Chapter 1
Introduction

This document is written for the Bachelor of Science project at the TU Delft in collaboration with SKION(DCOG)[1]. SKION is a foundation that facilitates research of children with cancer.

Several times a year, Dutch oncologists and pathologists from academic hospitals all around the country meet in real life to review patients. As the reader might notice, this is a very inefficient way of working, since all participants have to come to a central location. Therefore Martijn Tijsma has made an outline of possibilities to improve the efficiency of reviewing, introducing the Digital Review.[2] The main idea of this new concept is that oncologists and specialists, such as pathologists, radiologists and cytogentici, can communicate and collaborate without leaving their own hospital. All this should be achieved through video conferencing, where additional streams, such as real-time microscope feeds and administrative documents, are shared. The scope of the project is to realize the automation of the administrative part as mentioned above. To get a complete idea on the Digital Review concept, please consult Tijsma's report. Please note that at this stage, we already have the knowledge about the intensively researched solution to the original problem which SKION faces.

This document describes the requirements that have to apply to the system that has to be built.

[1]Stichting Kinderoncologie Nederland (Dutch Childhood Oncology Group)

[2]Tijsma, 2009, Digital Review possibilities for replacing central review at SKION

Chapter 2
Current Situation

Currently there is only one pathologic Review, the MDS/BMF[1] review, in operation facilitated by SKION. So the best way to describe the current situation is to look at this review. First Bone marrow or blood samples are send, by academic hospitals, to SKION to be researched. The samples are researched the same day as they come in. The patient is registered in the SAP database. The case is given a number (SKION number[2]). The registration information in SAP is exported and later imported into the Promise database. If there is suspicion of BMF or MDS a patient is added to the review and the treating doctor is informed. More information from the hospital is needed therefore a pre-CRF[3] is sent to the data manager in the academic hospital. be researched.

The top-part of the pre-CRF is filled in by the SKION laboratory. Together with the treating doctor the data manager completes the form, the doctor signs it and the form is returned to SKION. The data from the pre-CRF is entered by the SKION trial bureau into the promise database.

Four weeks before the review is going to take place, SKION checks if all the necessary clinical information is complete. If so, SKION sends the review pathologist who is in charge of the Review the following documents:

1. pre-CRF as filled in by SKION and local data managers.
2. A copy of the letter as recieved by SKION containing the diagnosis of the local pathologist.
3. A list of patients who are to be reviewed.
4. Slides of bone marrow or blood and the results of the research as performed by SKION.

Within a maximum of three weeks before the Review, the central pathologist requests bone biopts at the local hospitals. When receiving these, he performs a first review at the samples, writing his findings on the Pathology review & classifying diagnosis form.[4]

Every three months, the actual review is being held in Utrecht. The chairman of the review guides the meeting so that the topics can be visited in a structured way. The central pathologist as mentioned before is presenting his findings to the board of specialists participating in the Review. A discussion, if necessary, is being held and a diagnosis is stated. This is done by filling in the CRF form. The form is signed by the protocol chair and collected by SKION. If no diagnosis could be determined, the patient is scheduled for review again. For the next review, new samples are requested from the local hospital.

[1]MDS = Myelodysplastic syndrome, BMF = Bone Marrow Failure
[2]In Dutch also called 'ziektegevalnummer'
[3]CRF: Case Report Form; see appendix 5
[4]See the appendix

Chapter 3
Desired System

3.1 Overview

Currently, sending the pre-CRF to and getting a reply from the local data manager takes up to a month. This time could easily be reduced by letting the datamanager know the pre-CRF has to be filled in (for example by email) and enabling him to do so on his local computer.

This data should now be easily accessable by the central pathologist for every patient that is now in the review phase, so that he can prepare the Central Review. The central pathologist will use this data, in combination with the slides sent from the academic hospitals, to give an opinion on the patient. This information should be entered in the system aswell.

All this data should be available at the Central Review for the participants of the Review. After the Review a classifying diagnosis has to be filled into the system. This requires a digital signature to verify the data that has been filled in.

Now the patients for whom a diagnosis has been made, might leave the review phase. This should be notified to the system, so that the list of patients that should go into the next review can be made. This list consists of the patients that require another review after the last review, the patients that didn't recieve a review, but are still in the review phase, and the patients who are added to the review program since the last review and have sufficient data to start a review with.

3.2 Functional requirements

In the definition of the system, we distinguish the following users:

- SKION review manager
- SKION Trial Bureau
- Datamanagers
- Review pathologists
- Clinicians
- Chairman
- Minutes secretary

3.2.1 SKION review manager

is the main administrator in the system. This actor will keep track and ensure the consistency of the data in the system.

1. SKION review manager needs to be able to plan a new review (set a date).
2. SKION review manager needs to be able to organize a new (not yet existing) review.
3. SKION review manager needs to see the patients that are eligible for the review.
4. SKION review manager needs to be able to add existing patients to a review.

5. SKION review manager needs to be able to force the review of a patient, even if the information is incomplete.

3.2.2 SKION trial bureau

1. SKION trial bureau has to be able to start a pre-CRF and notify the regarding datamanager. This means the system has to execute the following steps.
   (a) SKION trial bureau has to fill in some of the patients details.
   (b) The system has to save these details.
   (c) The system has to notify the datamanager that the rest of the patient details have to be filled in.

3.2.3 Datamanagers

are the datamanagers that are located at the different academic hospitals around the country.

1. The local datamanagers have to be notified if a patient has been suspected of having cancer.
2. The local datamanagers have to fill in a form with information about the patient and have to save this to the promise database.

3.2.4 Review pathologists

are the specialized academic pathologists who requests the samples of the patients before the reviews and present their findings during the review.

1. The review pathologists need to be notified that information on the patient has been completely filled in.
2. The review pathologists need to be able to get this information from the system at any time.
3. The review pathologists need to be able to save their findings into the system.
4. The review pathologists need to be able to see a list of patients that are up for the review.
5. The review pathologists need to be able to indicate that they received the specimen / slides from the academic hospitals.

3.2.5 Clinicians

are several doctors that are positioned at the different academic hospitals around the country that have to be present during the Review. Each of these clinicians have their own specialization.

1. The clinicians need to be able to see the patients information at all times.
2. The clinicians need to be able to see the list of patients included in the review.
3. The clinicians need to be able to see the results of the Review.
4. The clinicians need to be able to sign the results of the Review with a digital signature.
5. The clinicians need to be able to enter additional patient history and background information into the system.

3.2.6 Chairman

is the one who is responsible for the course of the review.

1. The chairman needs to be able to adjust (or enter) the probable diagnosis visible in the patient list.
2. The chairman needs to see the patients that are enlisted for the current Review.
3. The chairman has to be able to have insight in the patients' information during the Review.

3.2.7 Minutes secretary

is the one who is responsible for writing down what is said during the course of the review and thus the assistant of the chairman.

1. The minutes secretary has to be able to save the findings of the review into the system.

3.3 Nonfunctional requirements

In this section we will look at the other requirements for the system to be built. These requirements will be the requirements of the system that don't have something to do with the functionality of it, but rather the boundary conditions in which the system needs to operate and the requirements of the system other than the system itself.

3.3.1 User interface and human factors

All users have basic knowledge of computing and computers. It is expected that all users have a minimum degree of a University of Professional Education[1]. The user interface should be clear and intuitive to the users and it should be able to be used without consulting an user manual. Since the application is expected to be web based, only a webbrowser is needed. Users are expected to know how that browser works.

3.3.2 Documentation

The software to be build must be documented very well. This is important because the software is expected to be a fundament for future development of this project. No user manual is provided, but a Frequently Asked Questions section would be nice to have.

3.3.3 Logging

It is desirable to log all activities preformed in the webapplication. If implemented, an administrator can check who has performed which action. This can be of use if the need is there to verify if a user has performed some action.

3.3.4 Hardware considerations

The users of the system will all be working at medical institutions. These institutions have reasonable quality computers, a good Internet connection, but strict firewalls. We expect that all computers are able to run a browser.

[1]Dutch: HBO opleiding

3.3.5 Performance characteristics

The system is expected to be consulted mostly during work hours, between 08:00 am and 17:00 pm. From practical experience, existing oncologic meetings are held at the end of the day. So preferably the system has to be online till 20:00 pm. On normal basis, it will be highly exceptional that two or more users are working with the system at the same time. However during a Review, it is possible that ten or more users at the *same time* access the same information. During the review, the information is also editted by the minutes secretary and possibly updated in real time at other clients.

The response time the application has, may be no more than an avarage web page. It has to function in a smooth way, so that the users get the feeling they are working on a local program. In future plans, it will also be possible to have multiple Reviews at the same time so the system will be loaded more heavily. Even then, the system is required to operate in a smooth way.

3.3.6 Reliability

If data is submitted, the user must be ensured the data is saved. If an error occurs, the user must be notified about this error.

3.3.7 Error handling and extreme conditions

Errors on user machines may not influence the performance of the system. If an error occurs during digital validation, the validation is not completed and the document not signed.

3.3.8 System interfacing

The system should have an interface to the ProMISe system, an application containing patient data, driven by a MySQL database. Both read and write operations are to be performed from and to ProMISe. The system must also have a read-only interface to SAP, where Lab results are stored.

3.3.9 Quality issues

If data is saved on the system, any user must be 100% sure the data is really saved.

Official review documents have to be signed by a digital signature. More details on the digital signature can be read in our Digital Signature Research document.

The software must be as safe as possible in order to protect the privacy of patients involved.

3.3.10 System modifications

The system, as to be build at this point in time, serves mostly as proof of concept of Digital Review. At first the system will only contain functionality for the Digital Review for the MDS/BMF review. However, in the design of the program there will be the possibility of extending it easily for different types of new reviews.

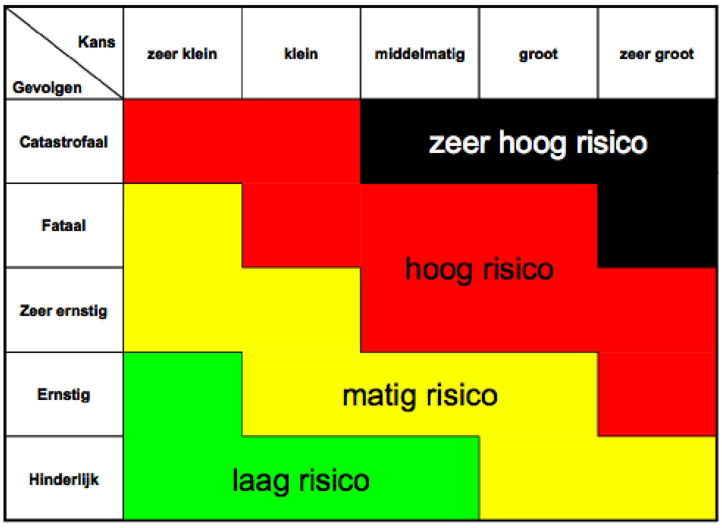
3.3.11 Physical environment

No special requirements.

3.3.12 Security issues

Security is a very important topic in the system. Since we are dealing with sensitive medical information, the system should be secure enough. How secure exactly is determined by the strict NEN[2] standards 7510, 7511 and 7512. We consider these standards bureaucratic, so we will only highlight some of the important topics for our system.

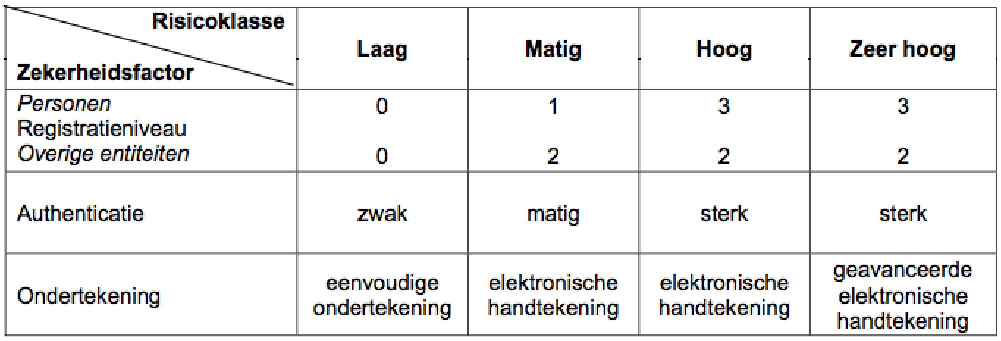
NEN7512 contains a nice indication of techniques to prevent bad things from happening in a medical information system. First, an estimation for the risk factor of our application is needed, using NEN7512 figure 3.1. NEN7512 figure 3.2 describes what kind of security is needed to prevent incidents with the system.



Kans / Gevolgen	zeer klein	klein	middelmatig	groot	zeer groot
Catastrofaal	🟥	🟥	⬛	⬛	⬛
Fataal	🟨	🟥	🟥	🟥	⬛
Zeer ernstig	🟨	🟨	🟥	🟥	🟥
Ernstig	🟩	🟨	🟨	🟨	🟥
Hinderlijk	🟩	🟩	🟩	🟨	🟨

zeer hoog risico
hoog risico
matig risico
laag risico

Figure 3.1: NEN figure 1



Risicoklasse / Zekerheidsfactor	Laag	Matig	Hoog	Zeer hoog
Personen	0	1	3	3
Registratieniveau				
Overige entiteiten	0	2	2	2
Authenticatie	zwak	matig	sterk	sterk
Ondertekening	eenvoudige ondertekening	elektronische handtekening	elektronische handtekening	geavanceerde elektronische handtekening

Figure 3.2: NEN figure 2

We estimate our application has a medium to high risk factor. From figure 3.2 can be deduced that an medium risk factor requires a medium strength of authentication, a check upon registration that the given identity is correct. Medium strength of authentication is secure implemented password authentication, biometric authentication, a physical object proving authenticity and a password protected digital certificate on a easy copyable medium.

A high risk facter requires a strong method of authentication, like biometric authentication in combination with a password, or a physical authentication object with a password. Or the

[2]NEderlandse Normen

use of digital certificates with a smart card. Also, the registration should happen face-to-face, in a document according to article 3 of the WID[3].

With an eye on the amount of time we can spend on this problem, we will provide the security for a medium risk application. Future work might include the use of smart cards with our system, ensuring that if the application is a high risk application, it is secure enough.

Apart from this, different actors are expected to have different read and write rights. Furthermore they also vary in restriction of access to information.

3.3.13 Resources and management issues

The system as designed in this document, has a special role for a minutes secretary. Currently in the Central Review, there is no minutes secretary, so to make use of this system, SKION should arrange someone to fill in the forms in the system, preferably someone with knowledge of medical terminology.

3.4 Constraints

A sufficient Digital Signature system must be used to ensure the safety of signing digital forms. The system will be build a combination of HTML/PHP/CSS/Javascript and backed by a MySQL database.

SKION has a quality manual, containing procedures for all activities of the lab and the trial office. Some paragraphs in the chapter "Commodoties" supply some requirements for information systems. Section 4.1.9 is about the validation of the information system and it describes in a few steps how to validate the application.

1. Create a conceptual design, what should the application do?
2. Create a functional design, how is it going to do that?
3. Write a validation plan, describing how to validate:
   - infrastructure
   - security
   - archiving
   - maintenance
   - description of raw data
   - test specifications
   - the functional requirements
4. Test the system with:
   - functionality tests in the testing environment,
   - functionality tests in the normal environment and
   - user tests
5. Evaluation and acceptance of the results
6. Writing a validation report

Our documents, as we will produce them, are very similar to the ones in this scheme.

[3]Wet Identificatie bij Dienstverlening, the law of identification for care providers

Chapter 4
System Models

4.1 Scenarios

Scenario 1a - Adding an user and assigning rights

Name:	Adding an user and assigning rights
Actors:	Martijn: SKION Review manager

Flow of events:

1. A new actor has to be added to the system. Martijn has all data available.
2. Martijn chooses the function *add new user* and fills in the data of the new user. Variables such as name, title, hospital, email address and phone number.
3. Next, Martijn chooses a type, which will be assigned to the new user.
4. Martijn submits the data.
5. An email with a verification link is send to the new user.

Scenario 1b - Forgot password

Name:	Forgot password
Actors:	Actor x: Any actor

Flow of events:

1. Actor x has forgotten his/her password and clicks on the 'forgot password' link at the inlog screen.
2. Actor x fills in his/her email address with which Actor x is registered in the system.
3. Actor x recieves an email with a verification link in his/her inbox.
4. Actor x clicks the link and a new password is send to him/her.
5. The first time Actor x logs in, he/she is requested to change the password.

Scenario 2a - Submitting a pre-CRF (Filling by datamanager)

Name:	Submitting a pre-CRF	
Actors:	Martijn:	SKION Trial agency
	Dirk:	Datamanager

Flow of events:

1. Martijn wants to send a pre-CRF about a patient they have examined.
2. Martijn starts the program and fills in the patient's ID and presses submit.
3. The request for filling in the pre-CRF has been sent to Dirk, who now has access to a form with patient ID, initials, birth date and sent hospital filled in.
4. Dirk fills out the form and submits the form to the system. The patient's information is now saved into the database.

Scenario 2b - Submitting a pre-CRF (Filling by SKION trial agency)

Name:	Submitting a pre-CRF	
Actors:	Martijn:	SKION trial agency
	Dirk:	Datamanager

Flow of events:

1. Martijn wants to send a pre-CRF about a patient they have examined.
2. Martijn starts the program and looks up the patient's id.
3. Martijn fills in the information on the pre-CRF paper and sends it to Dirk.
4. Dirk fills out the pre-CRF on paper and sends it back to Martijn.
5. Martijn fills in the information in the system and submits. The patient's information is now saved into the database.

Scenario 3 - Sending letter to regarding doctor(s) and pathologist(s)

Name:	Sending letter to regarding doctor(s) and pathologist(s)	
Actors:	Martijn:	SKION Trial agency
	Dirk:	Datamanager
	Lennert:	Review pathologist

Flow of events:

1. Since Martijn and Dirk now filled in the pre-CRF, a review can be arranged.
2. Four week in advance of the reviewdate, Lennert will recieve a copy of the letter that was sent to Martijn by the attending doctor requesting the examination of the patient, as well as the pre-CRF filled in by Martijn and Dirk.
3. The system creates a form for Lennert that can be filled in and submitted into the system.

Scenario 4 - The Review

Name:	The Review	
Actors:	Lennert:	Review pathologist
	Frank:	Chairman
	Debrah:	Minutes secretary
	Jozef:	Clinician

Flow of events:

1. The review starts and Frank starts the program.
2. Lennert and Jozef start the program so that they can see the patients information.
3. Frank retrieves a list of the patients and decides to start with the first patient and selects this one from the list.
4. Lennert and Jozef pay attention to what Frank is saying while they can see information about the patient independent of what kind of information Frank is currently showing with his program.

5. After the patient has been thouroughly discussed, Debrah enters the findings into the system.
6. Frank chooses the next patient and the proces starts over.

Scenario 5 - End of Review

Name:	End of Review	
Actors:	Debrah:	Minutes secretary
	Jozef:	Clinician

Flow of events:

1. For each patient that has been discussed, Debrah saved the findings into the system.
2. When every patient has been discussed, Jozef opens the program and sees that the findings need his signature.
3. For each patient he places his signature and submits the findings to the system.

Scenario 6 - Printing a completed form

Name:	Printing a completed form
Actors:	Actor x: Any actor

Flow of events:

1. Actor x selects a form to get a hardcopy duplicate and prints it.

Scenario 7 - Adding anamnese

Name:	Adding anamnese
Actors:	Jozef: Clinician

Flow of events:

1. Jozef decides he wants to add an anamnese for a patient who is registered for the review phase.
2. Jozef starts the program and opens up the form to add the anamnese.
3. Jozef fills in the fields and presses submit and the anamnese is saved into the system.

Scenario 8 - Planning a new review date

Name:	Planning a new review date
Actors:	Martijn: SKION Review manager

Flow of events:

1. Martijn employee chooses plan new review.
2. Martijn selects the review type and enters the date and time.
3. The participants are notified of the new review.
4. If necessary a follow-up CRF is requested for repeating patients from the academic hospitals.

Scenario 9 - Remove patient from current review list

Name:	Remove patient from current review list
Actors:	Martijn: SKION Review manager

Flow of events:

1. Martijn starts up the program and requests the list of patients up for the next review.
2. Martijn selects a patient that has to be removed for the next review.
3. Martijn deletes the patient, information on the patient is saved but is not in the list for the next review.

Scenario 10a - The follow up (filled in by datamanager)

Name:	The follow up	
Actors:	Martijn:	SKION Trial agency
	Dirk:	Datamanager

Flow of events:

1. Martijn decides it is time to start a follow up.
2. Martijn starts up the program and creates a follow up form.
3. The system sends a notification to Dirk that he has to fill in the from.
4. Dirk fills in the rest in the form and submits the information. The information is stored in the system.

Scenario 10b - Follow-up CRF (filled in by datamanager)

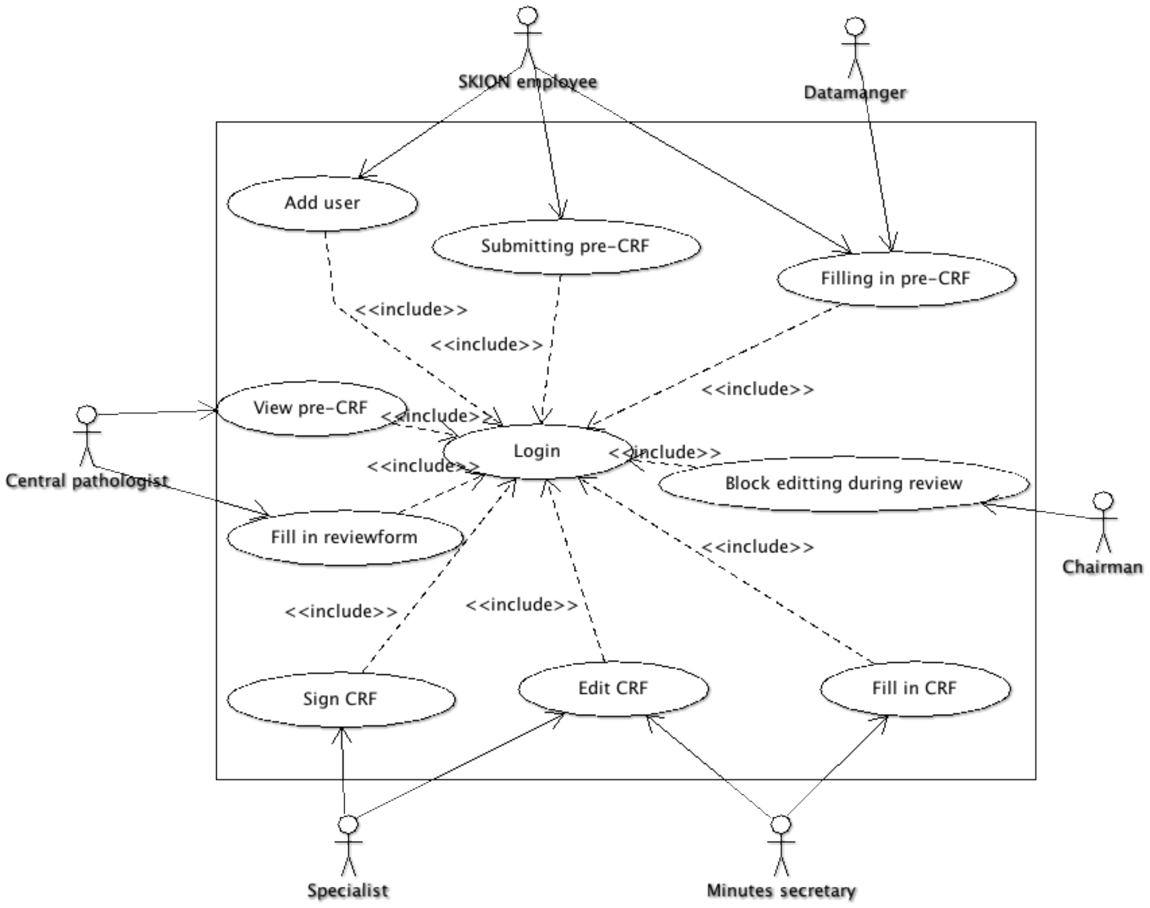
Name:	The follow up	
Actors:	Martijn:	SKION employee
	Dirk:	Datamanager

Flow of events:

1. Martijn decides it is time to start a follow up.
2. Martijn sends a follow up CRF to Dirk.
3. Dirk fills in the information and sends it back.
4. Martijn fills in the information in the system and submits the information. The information is stored in the system.

4.2 Usecases

4.2.1 Usecase model



4.2.2 Usecases

Use case 1 - Logging in

Actor:	SKION Review manager, SKION Trial agency, datamanagers, review pathologist, clinician,
Entry condition:	Program has started

Flow of events:

1. The actor sees two fields, one where the username has to be filled in, one where the password has to be filled in.
2. The actor fills in the username and password correctly and presses the login button.
3. The system verrifies the submitted information and certain rights have been given depending on which actor logged in.
4. The interface for that actor is shown by the system.

Exit condition: The actor has logged in and rights have been given.

Use case 2 - Logging in failure

Actor:	SKION Review manager, SKION Trial agency, datamanagers, review pathologist, clinician,
Entry condition:	Program has started

Flow of events:

1. The actor sees two fields, one where the username has to be filled in, one where the password has to be filled in.
2. The actor fills in the username and password incorrectly and presses the login button.
3. The system checks the information but no such username/password combination is known to the system.
4. The system returns to the login screen and notifies the actor of his/her error.

Exit condition: The user has not been logged in.

Use case 3 - Starting a pre-CRF

Actor:	SKION Trial agency
Entry condition:	actor has logged in

Flow of events:

1. SKION starts the part of the system for creating a pre-CRF.
2. SKION fills in a correct patient's ID.
3. The patient's initials, birth date and hospital of origin are found by the system and filled in on the interface on the screen.
4. SKION verrifies the information and presses the submit button.
5. The system creates a form and sends a notification to the regarding datamanager.

Exit condition: a pre-CRF form has been partly filled in in the system and a notification has been send to the datamanager.

Use case 4 - Starting a pre-CRF with correct patient ID, but a notification can't be sent

Actor:	SKION Trial agency
Entry condition:	actor has logged in

Flow of events:

1. SKION starts the part of the system for creating a pre-CRF.
2. SKION fills in a correct patient ID.
3. The patient's initials, birth date and hospital of origin are found by the system and filled in on the interface on the screen.

4. SKION verrifies the information and presses the submit button.
5. The system couldn't send the message due a network issue.
6. The system saves the filled information and the interface is shown with the error message.
7. The system will retry to send the message within the day.

Exit condition: a pre-CRF has been partly filled in and saved to the system. The system will send a notification when the email can be send.

Use case 5 - Starting a pre-CRF with unknown patient's ID

Actor:	SKION Trial agency
Entry condition:	actor has logged in

Flow of events:

1. SKION starts the part of the system for creating a pre-CRF.
2. SKION fills in a patient ID.
3. The system finds no patient information for the patient ID.
4. The system returns to the interface with a message to notify SKION of the error.

Exit condition: No pre-CRF has been created by the system.

Use case 6 - Filling in the rest of the pre-CRF

Actor:	Datamanager/SKION Trial agency
Entry condition:	The system has sent a notification to the regarding datamanger. Datamanager logged in

Flow of events:

1. The datamanager selects a patient and fills in the information required.
2. The datamanager presses the send button.
3. The information is stored into the system and a notification is send to SKION.
4. The system registers the patient as to be reviewed.

Exit condition: A complete pre-CRF form has been saved to the system and the system registered the patient as to reviewed.

Use case 7 - Creating and filling in a pre-CRF form.

Actor:	SKION Trial agency
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to create a new pre-CRF form.
2. SKION fills in the complete form.
3. The form is saved into the system.

Exit condition: The pre-CRF form has been saved into the system.

Use case 8 - Succesfully planning the review

Actor:	Review pathologists, clinician, chairman, minutes secretary, SKION Review manager
Entry condition:	The system registered the patient as to be reviewed.

Flow of events:

1. The system creates a form for the central pathologists.
2. The central pathologists log into the system and see a list of patients registered for the next review.
3. The central pathologists fill in the form of the patient three weeks in advance of the review and submits the form.
4. The form is saved into the system.
5. The system registers the patient for the next review.
6. The system sends a notification to the central pathologists, specialists, chairman, minutes secretary and SKION that review has been planned.

Exit condition: The system has saved the information filled in by the central pathologists and the patient is registered for the next review.

Use case 9 - Unsuccesfully planning the review

Actor:	Review pathologists
Entry condition:	The system registered the patient as to be reviewed.

Flow of events:

1. The system creates a form for the central pathologists.
2. The central pathologists log into the system and see a list of patients registered for the next review.
3. The central pathologists doesn't fill in the form three weeks in advance of the review.
4. The information in the form is saved into the system.

Exit condition: The filled form is saved into the system and the patient(s) is still registered as to be reviewed.

Use case 10 - Consulting the patient's data

Actor:	Review pathologists, clinician, chairman, SKION Trial agency
Entry condition:	The actor is logged in.

Flow of events:

1. The actor sees a list of the patients up for the next review.
2. The actor selects one of the patients and the details interface of the patients is shown.
3. The actor selects another patient.
4. The interface is refreshed with information on the newly selected patient.

Exit condition: The actor is logged in.

Use case 11 - Logging out

Actor:	Revies pathologists, clinician, chairman, datamanager, minutes secretary, SKION Review m
Entry condition:	The actor is logged in.

Flow of events:

1. The actor selects logout.
2. The system notifies the actor that he/she is about to log out and requires a confirmation.
3. The actor selects that he indeed wants to log out.
4. The system removes the actor's rights and shows the login screen.

Exit condition: The actor is logged out and thus has no rights.

Use case 12 - Cancelling logging out

Actor:	Review pathologists, clinician, chairman, datamanager, minutes secretary, SKION Review m
Entry condition:	The actor is logged in.

Flow of events:

1. The actor selects logout.
2. The system notifies the actor that he/she is about to log out and requires a confirmation.
3. The actor selects that he doesnt want to log out.

Exit condition: The actor is logged in.

Use case 13 - Adding a user to the system

Actor:	SKION Review manager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to add a user.
2. The system shows a form for the user's details.
3. SKION fills in the form and submits.
4. The new user is added to the system with the details filled in by SKION.

Exit condition: The actor is logged in.

Use case 14 - Deleting a user from the system

Actor:	SKION Review manager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to show the list of current users.
2. SKION selects a user and selects delete.
3. A confirmation pops up.
4. SKION confirms and the user is deleted.

Exit condition: The actor is logged in.

Use case 15 - Cancelling deleting a user from the system

Actor:	SKION Review manager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to show the list of current users.
2. SKION selects a user and selects delete.
3. A confirmation pops up.
4. SKION cancels.

Exit condition: The actor is logged in.

Use case 16 - Creating a follow-up form.

Actor:	SKION Trial agency, datamanager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to create a follow-up form.
2. The system draws the follow-up form on the screen.
3. SKION fills in the patient's id and fetches the rest of the necessary information.
4. SKION submits the form and a notification is send to the regarding datamanager.

Exit condition: The follow-up form has been saved into the system. The datamanager is notified.

Use case 17 - Filling in rest of a follow-up form.

Actor:	SKION Trial agency, datamanager
Entry condition:	The actor is logged in.

Flow of events:

1. The datamanager opens up the program and sees the list of patients that require the follow-up form to be filled.
2. The datamanager fills in the form and submits the information.
3. The information is saved into the system and SKION is notified.

Exit condition: The follow-up form has been saved into the system. SKION is notified.

Use case 18 - Creating and filling in a follow-up form.

Actor:	SKION Trial agency
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects to create a new follow-up form.
2. SKION fills in the complete form.
3. The form is saved into the system.

Exit condition: The follow-up form has been saved into the system.

Use case 19 - Removing a patient from the to be reviewed list.

Actor:	SKION Review manager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects a patient from the list of patients to be reviewed on the screen.
2. SKION selects to delete the patient from the patient list.
3. The information is saved into the system.

Exit condition: The patient has been removed for the next review.

Use case 20 - Add a patient to the to be reviewed list.

Actor:	SKION Review manager
Entry condition:	The actor is logged in.

Flow of events:

1. SKION selects a patient from the list of patients available to be reviewed on the screen.
2. SKION selects to add the patient for the next review.
3. The information is saved into the system.

Exit condition: The patient has been added for the next review.

Use case 21 - Adding or editing Anamnese.

Actor:	Clinician
Entry condition:	The actor is logged in.

Flow of events:

1. The clinician selects to add or edit anamnese to the patient's information.
2. The clinician fills in or edits the anamnese form and submits the information.
3. The information is saved into the system.

Exit condition: The patient has an anamnese added or editted to it's information.

4.3 Object model

4.3.1 Data dictionary

Review pathologist
A review pathologist is a person who prepares a Review by reviewing samples in advance.

Chairman
A chairman in in charge of the Digital Review.

Datamanager
A datamanager works in a hospital and provides SKION with the necessary information.

Digital signature
A signature to sign forms in the system digitally.

Minutes secretary
A minutes secretary types the diagnosis stated during a Review in the system.

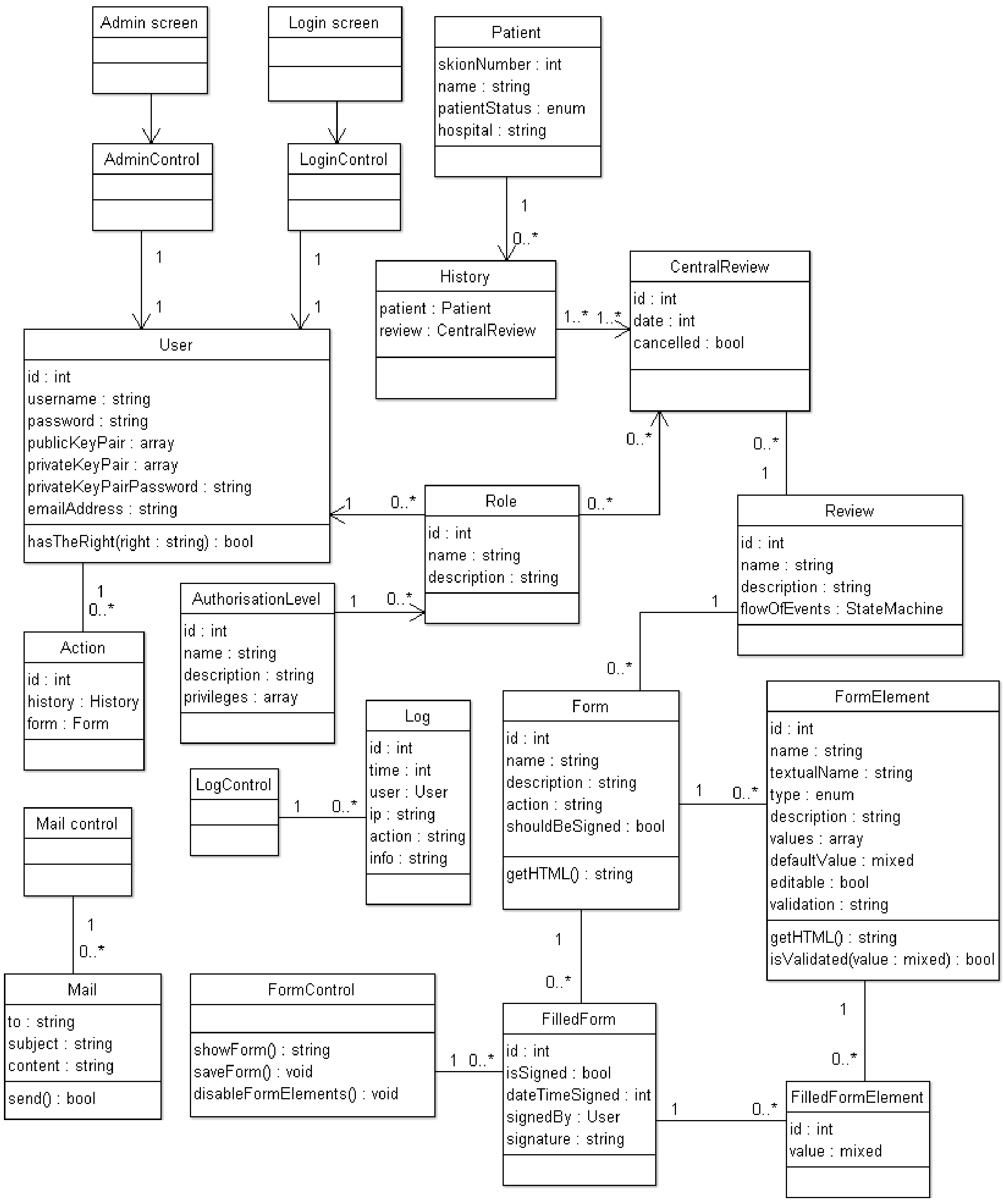
SKION Review manager
A person who manages all the data involved in the system.

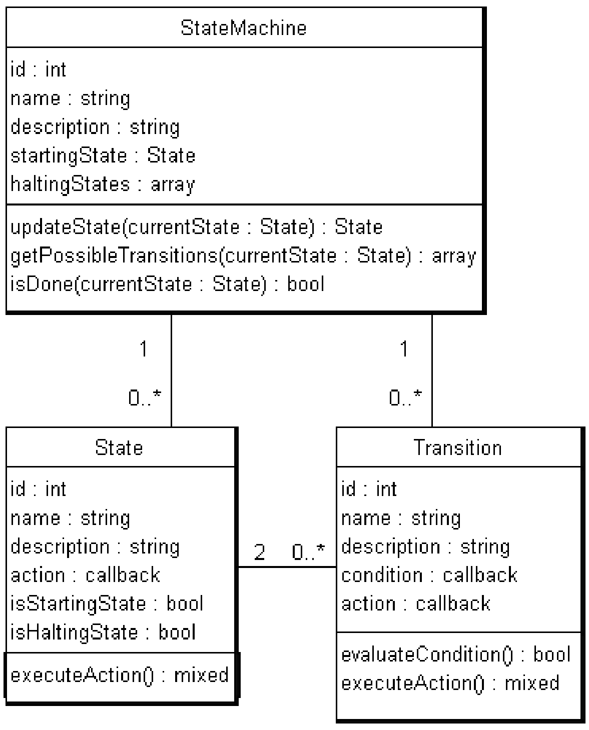
SKION Trial agency
A person of the Trial agency starts pre-CRF forms and follow-up CRF forms.

Clinicians
Clinicians that are positioned at the different hospitals around the country that have to be present during the Review.

System
The program that is being build.

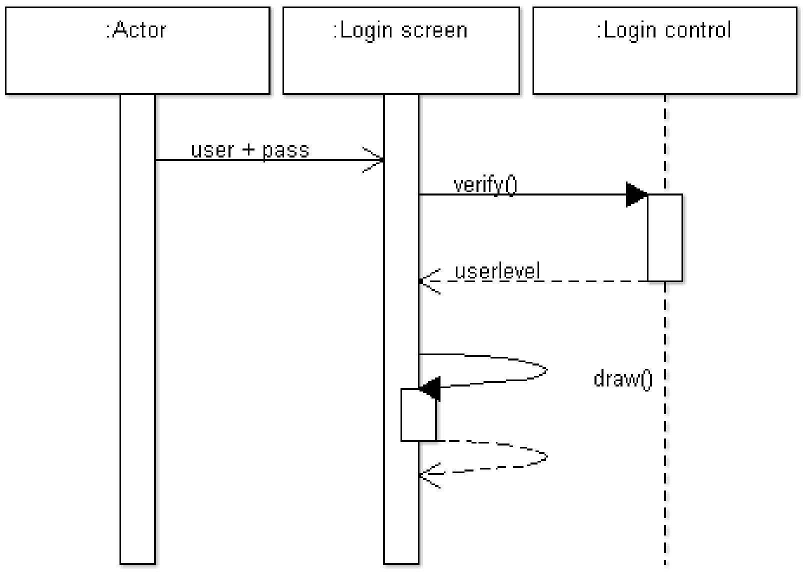
4.3.2 Class diagrams



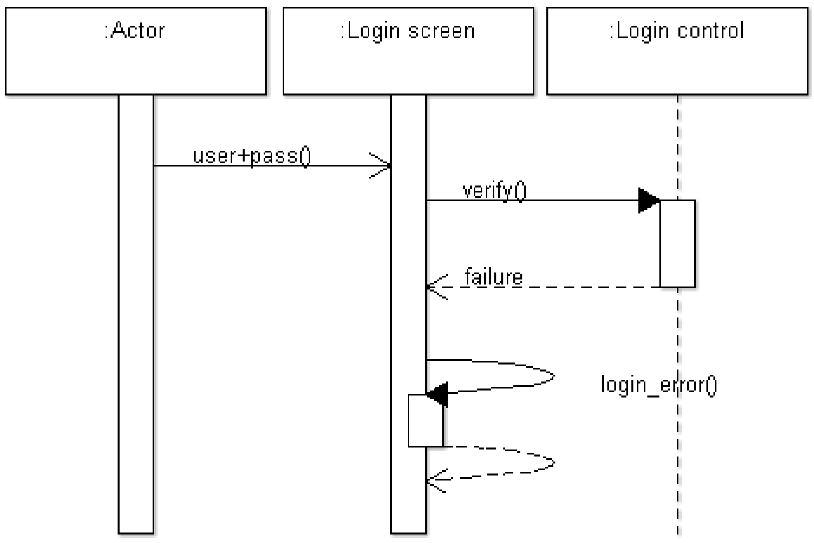

StateMachine
id : int
name : string
description : string
startingState : State
haltingStates : array
updateState(currentState : State) : State
getPossibleTransitions(currentState : State) : array
isDone(currentState : State) : bool
1
0..*
1
0..*
State
id : int
name : string
description : string
action : callback
isStartingState : bool
isHaltingState : bool
executeAction() : mixed
2
0..*
Transition
id : int
name : string
description : string
condition : callback
action : callback
evaluateCondition() : bool
executeAction() : mixed

4.3.3 Dynamic model

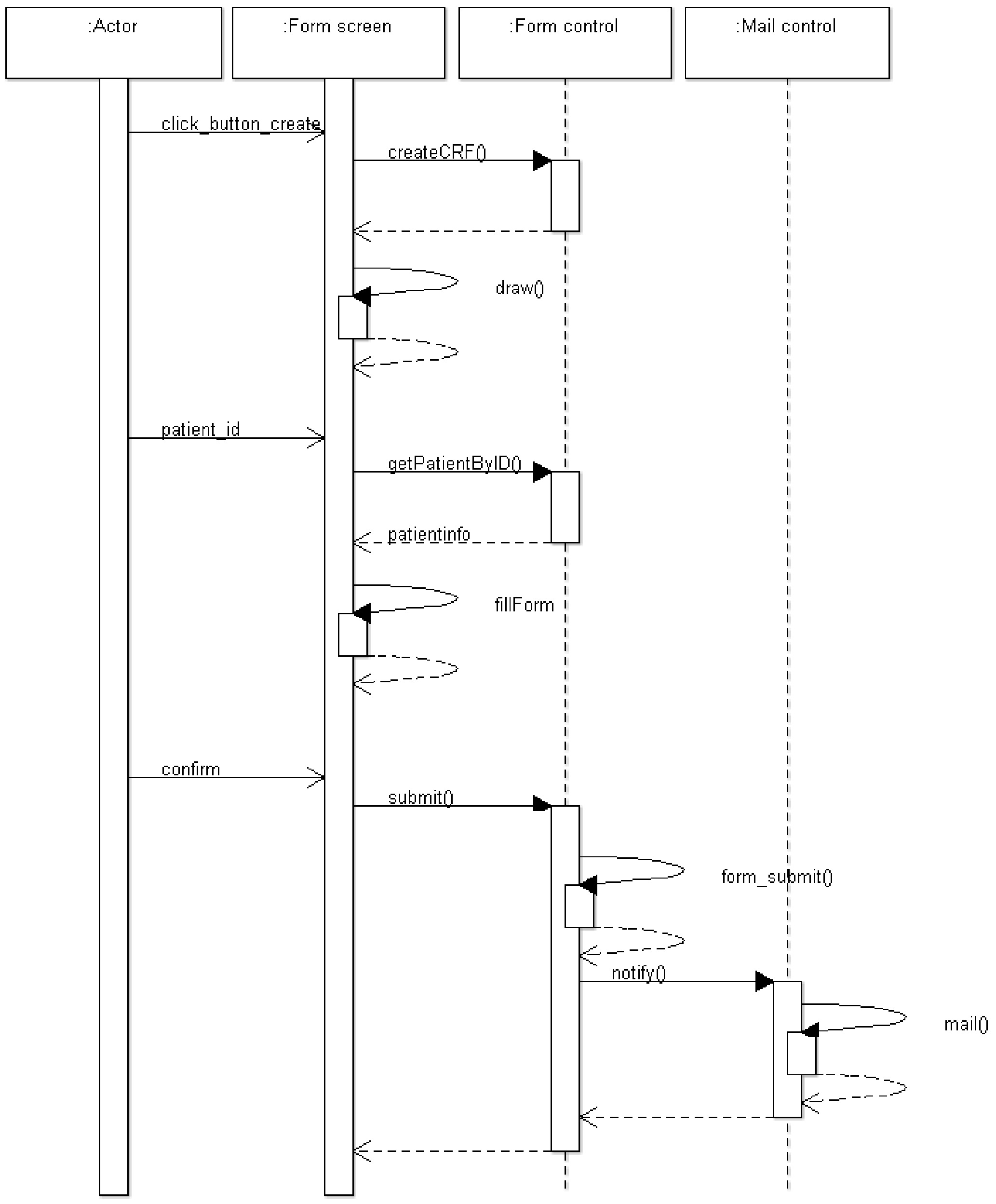
Logging in



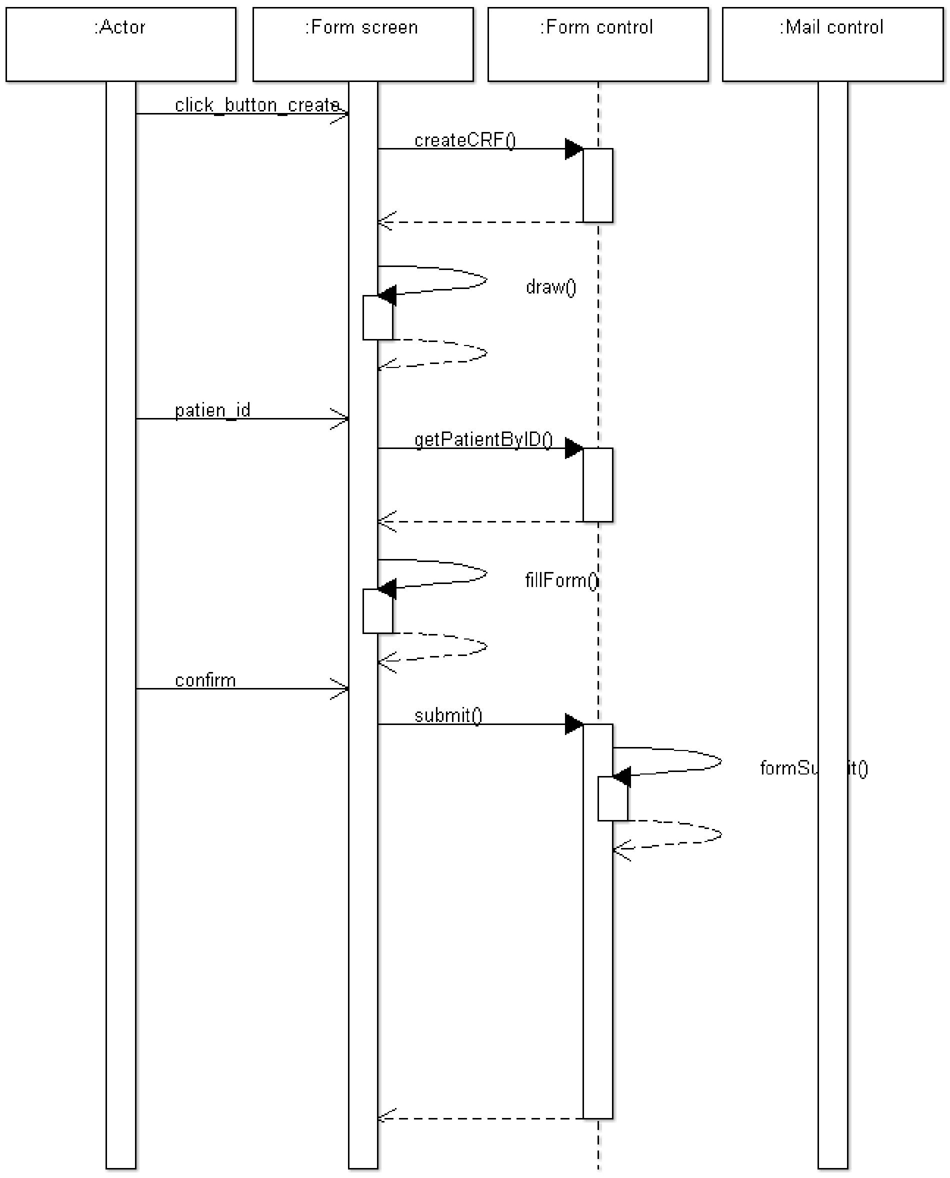
Logging in failure



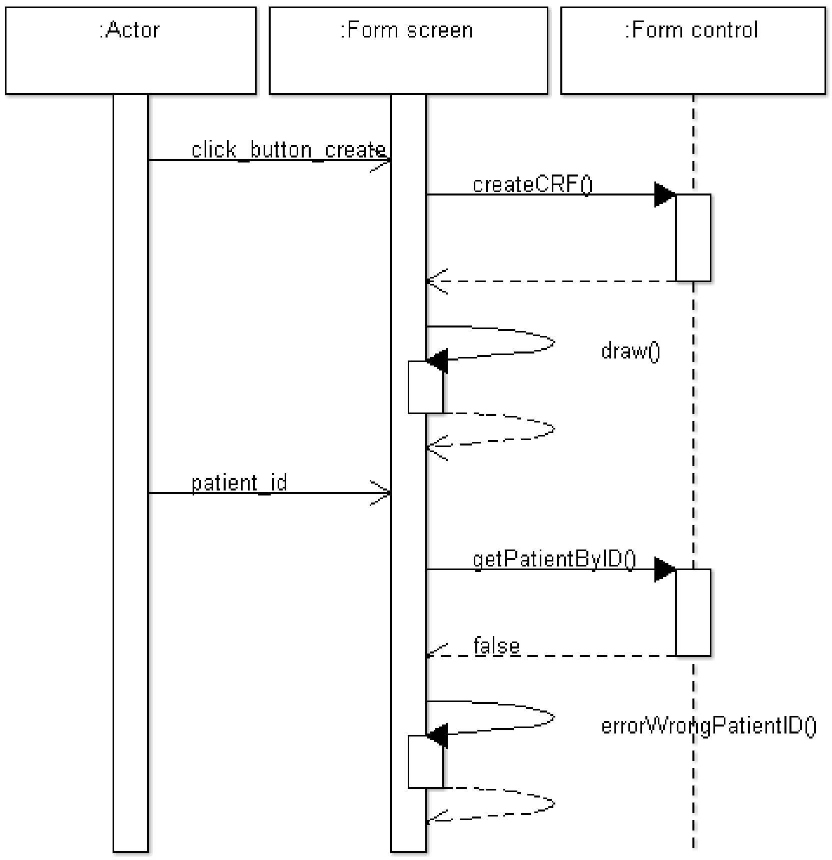
Starting a pre-CRF



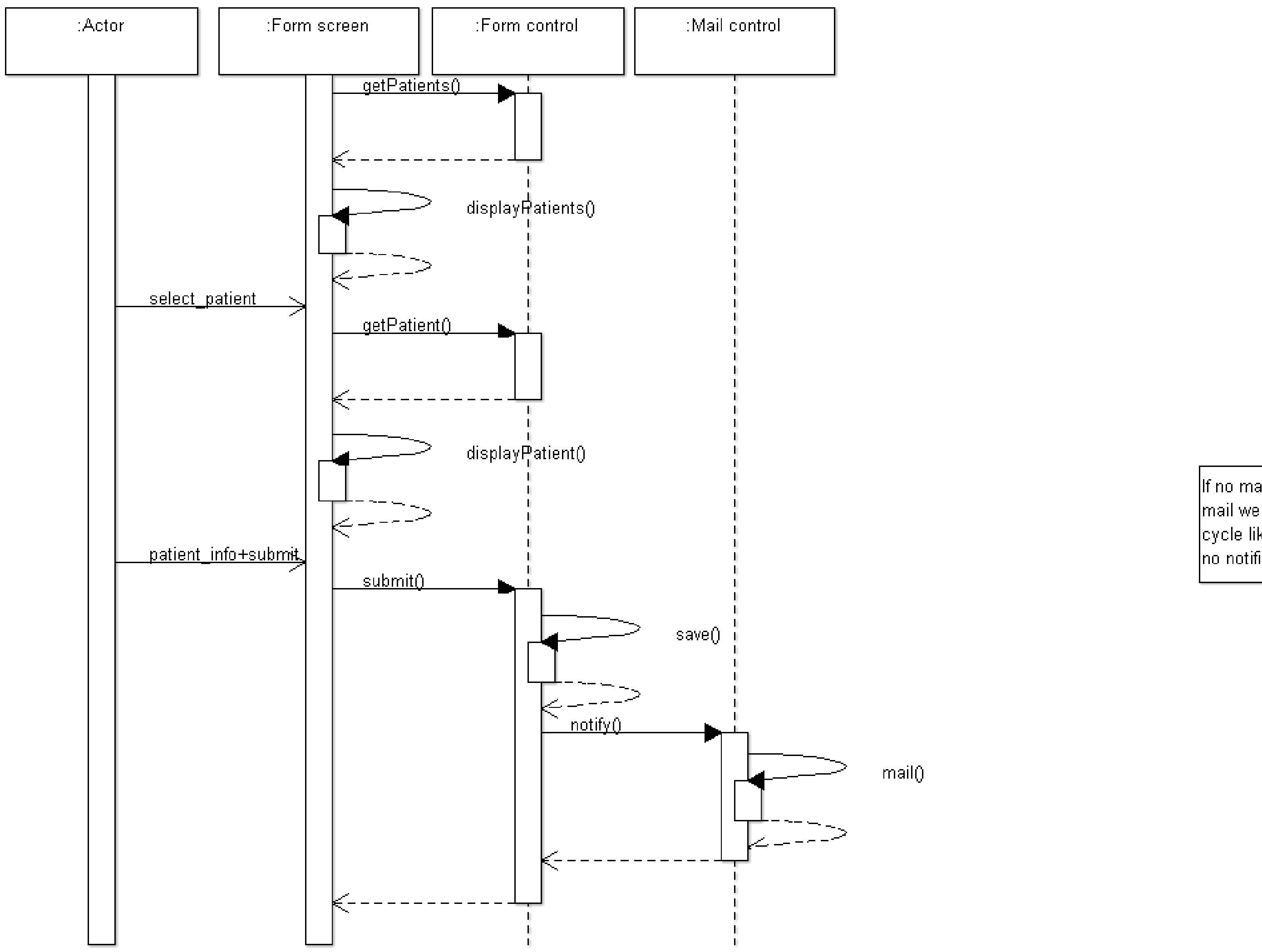
Starting a pre-CRF with correct patient ID, but a notifcation can't be sent



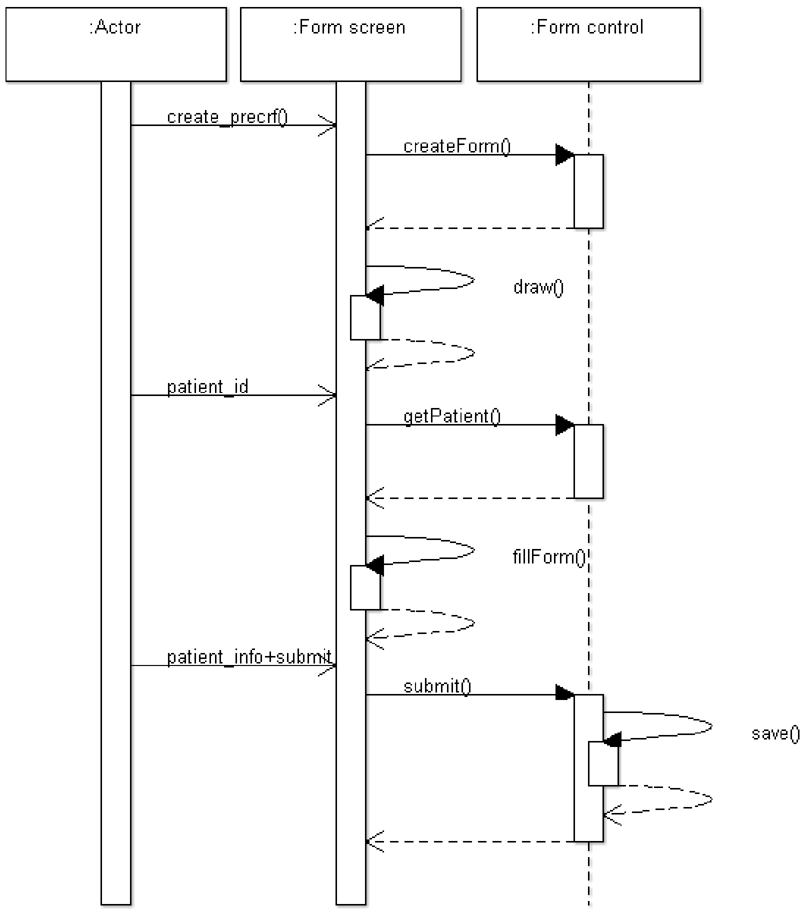
Starting a pre-CRF with unknown patient's ID



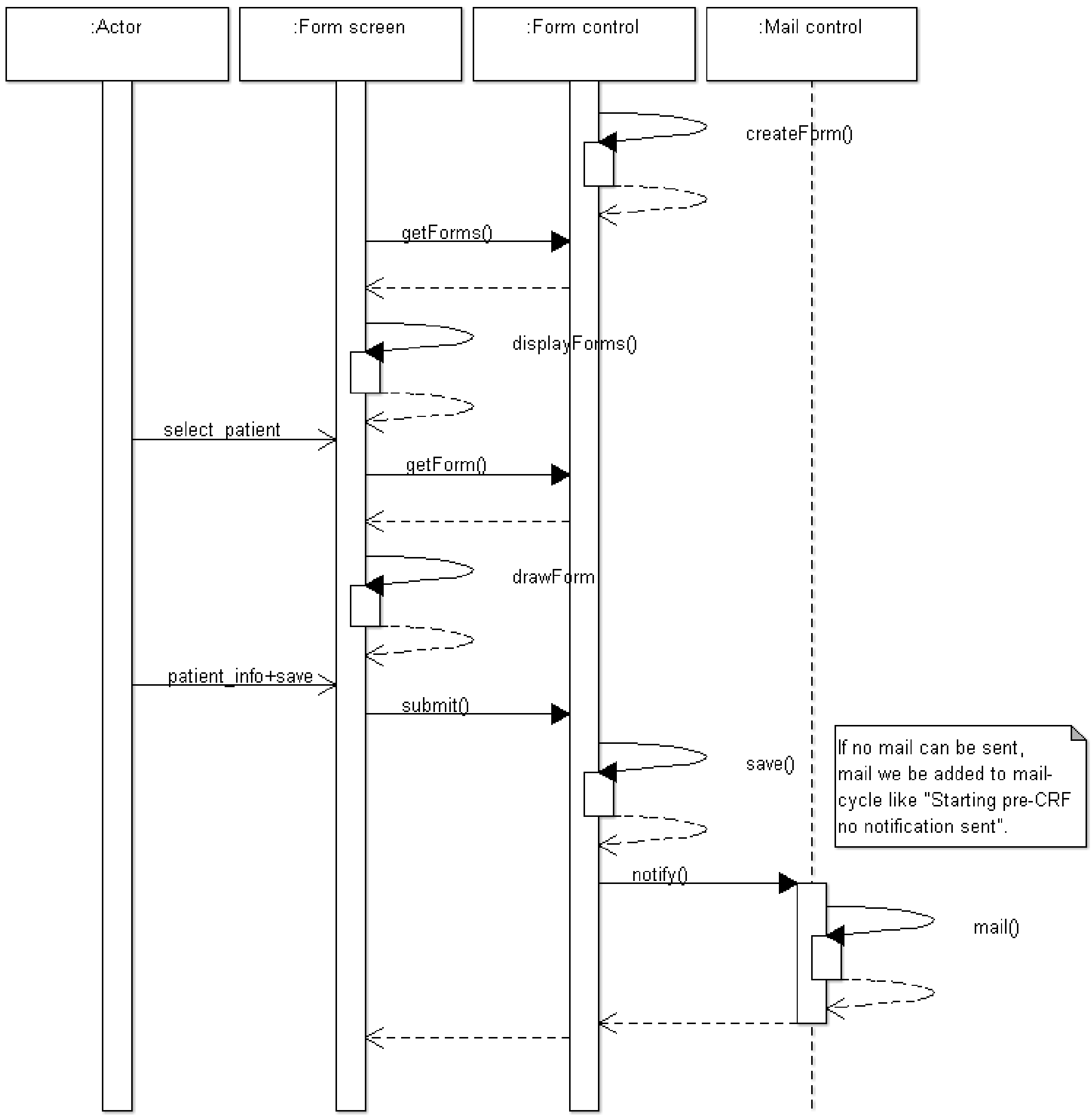
Filling in the rest of the pre-CRF



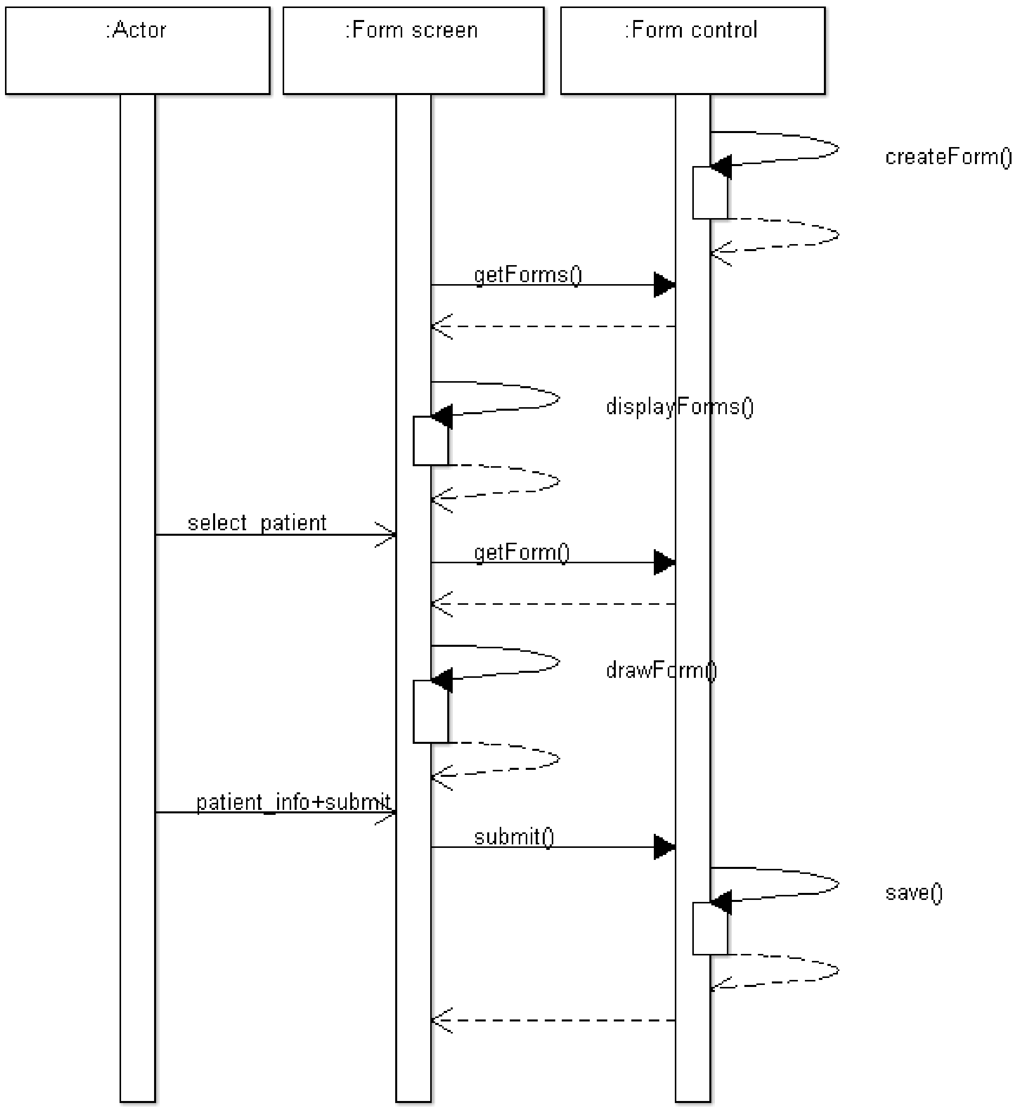
Creating and filling in a pre-CRF form



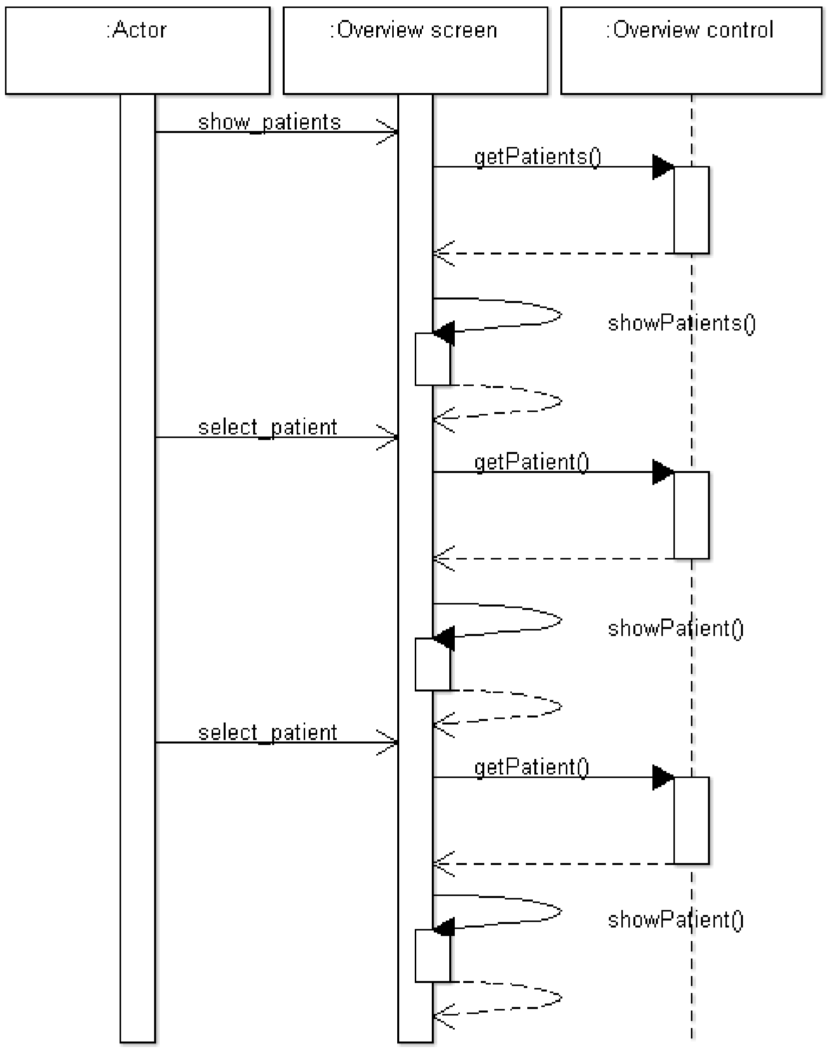
Succesfully planning the review



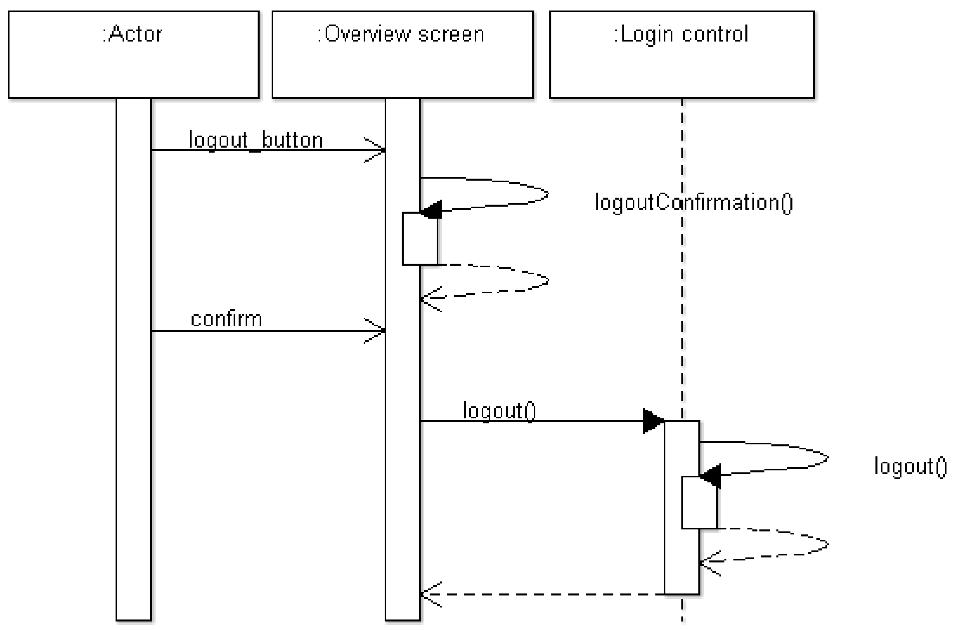
Unsuccesfully planning the review



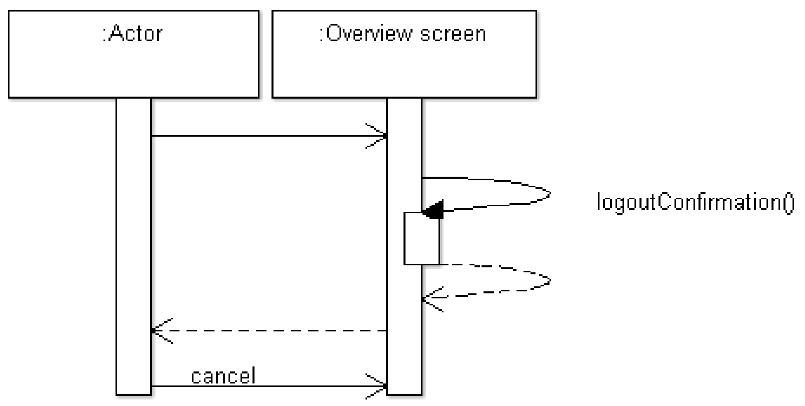
Looking in the patient's data



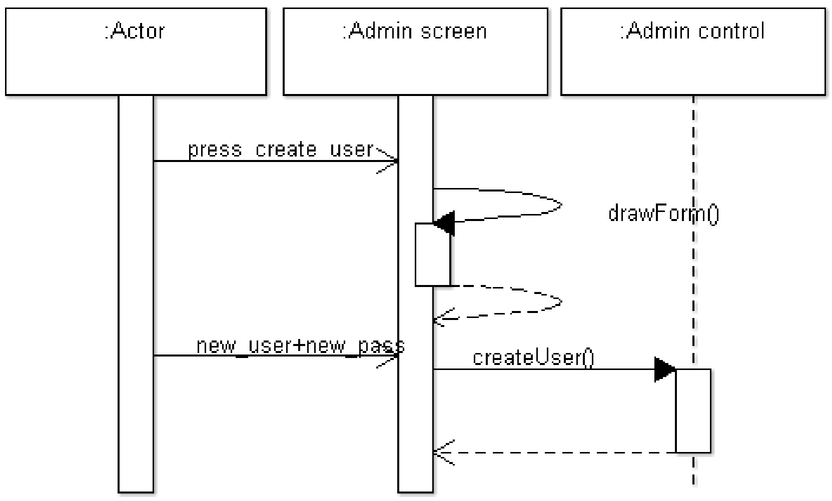
Logging out



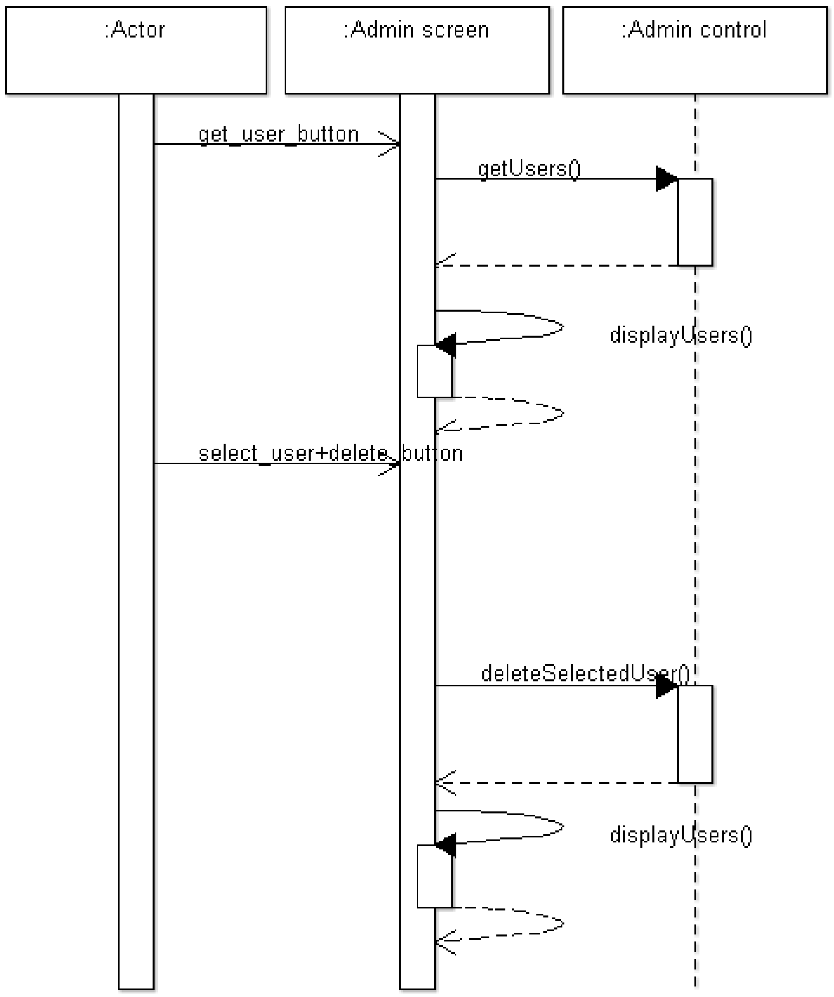
Cancelling logging out



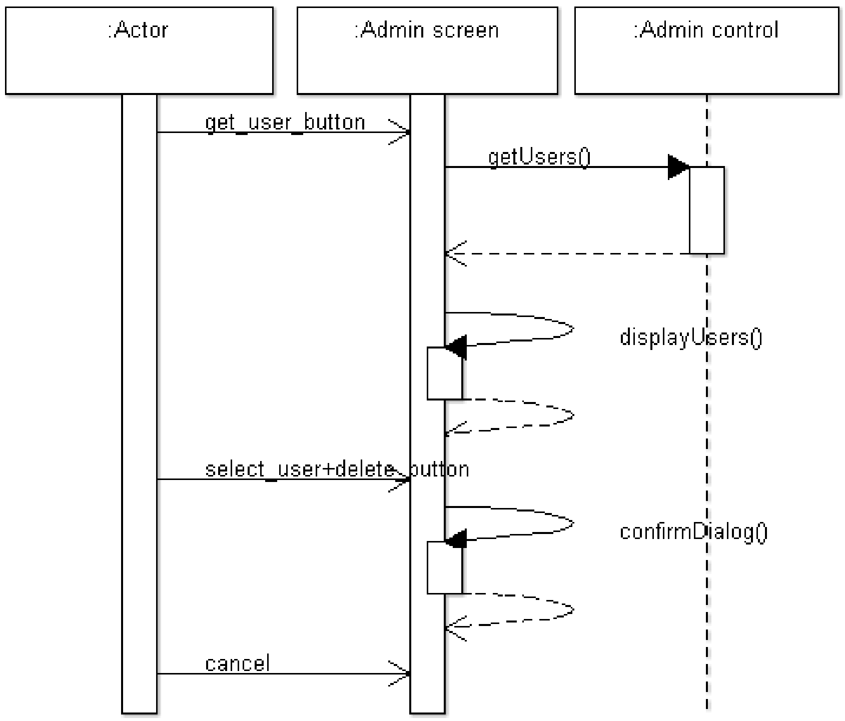
Adding a user to the system



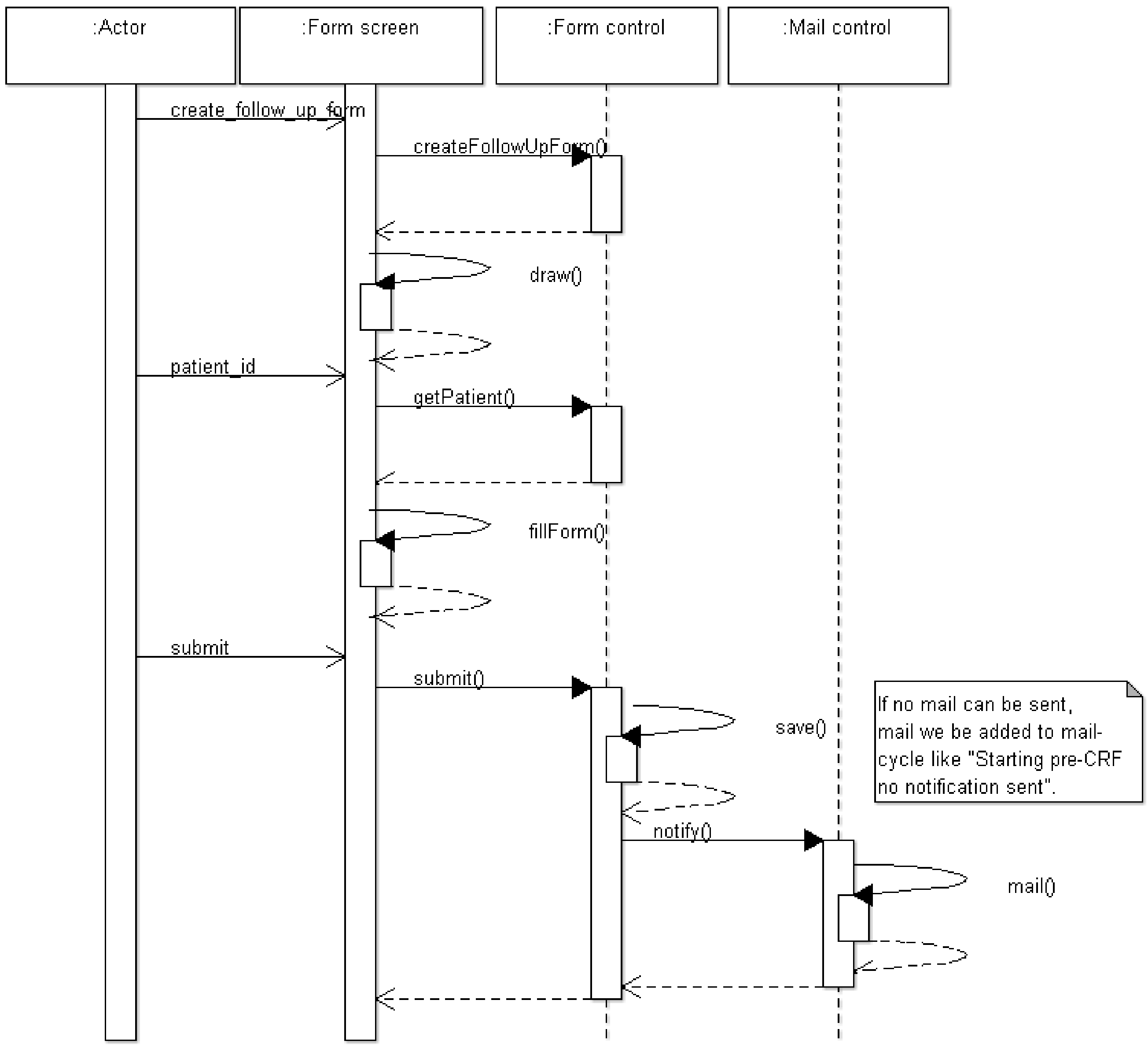
Deleting a user from the system



Cancelling deleting a user from the system

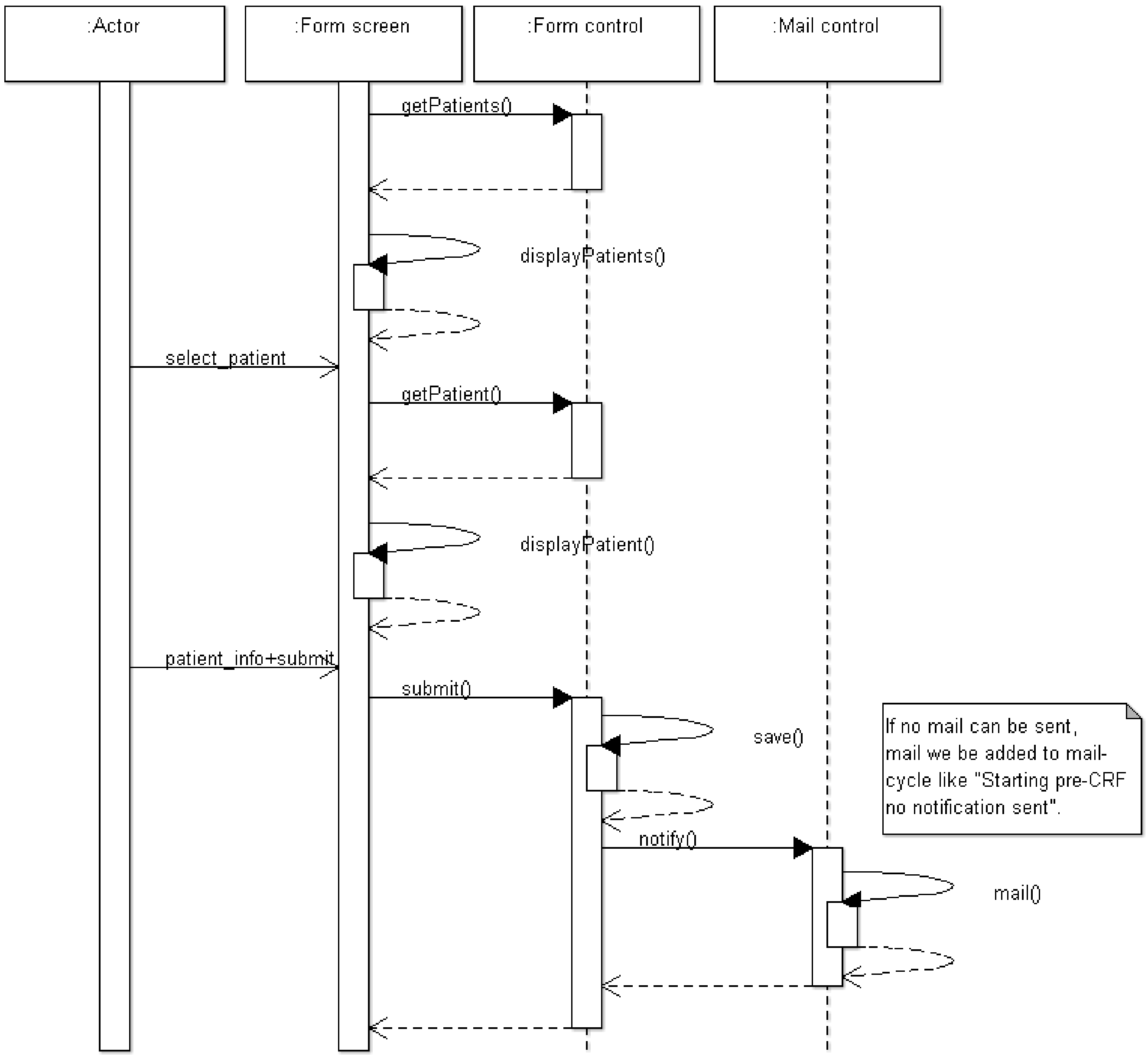


Creating a follow up form

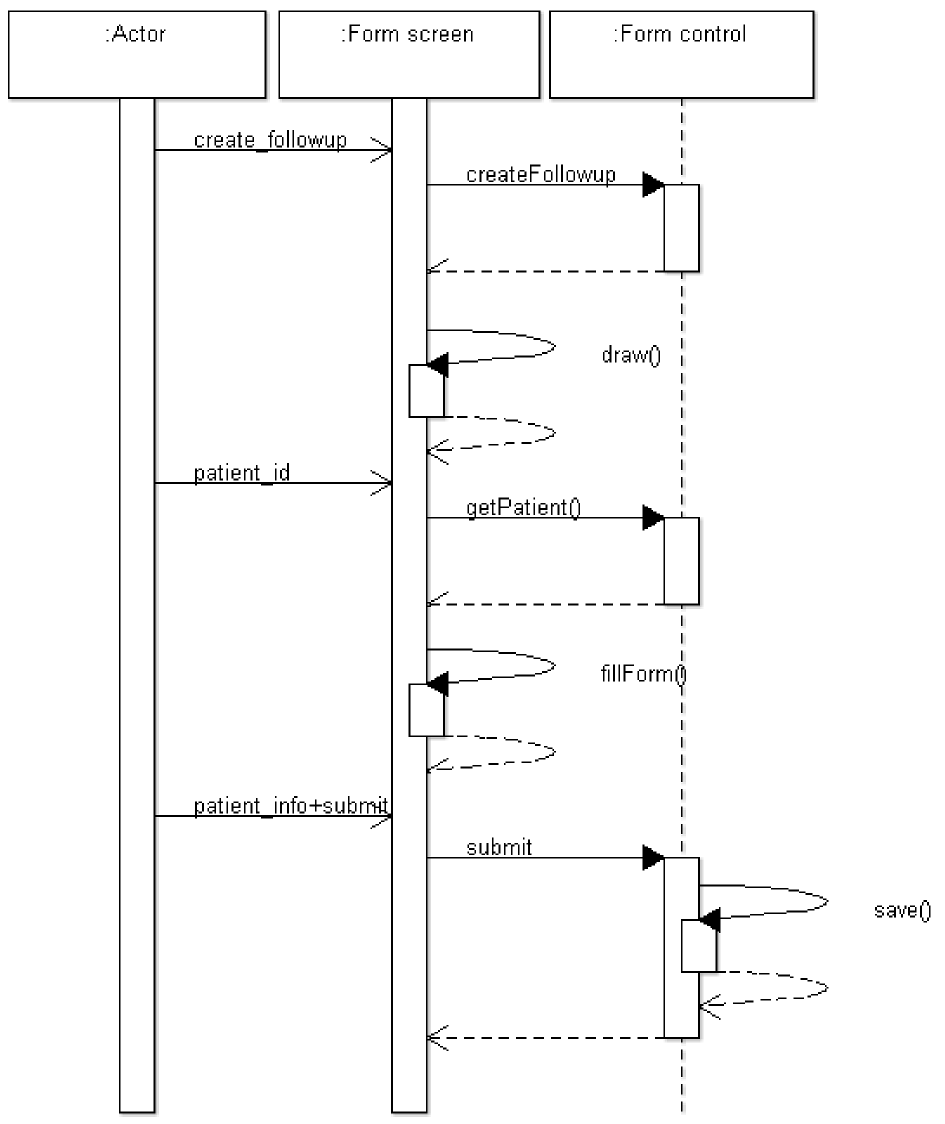


If no mail can be sent, mail we be added to mail-cycle like "Starting pre-CRF no notification sent".

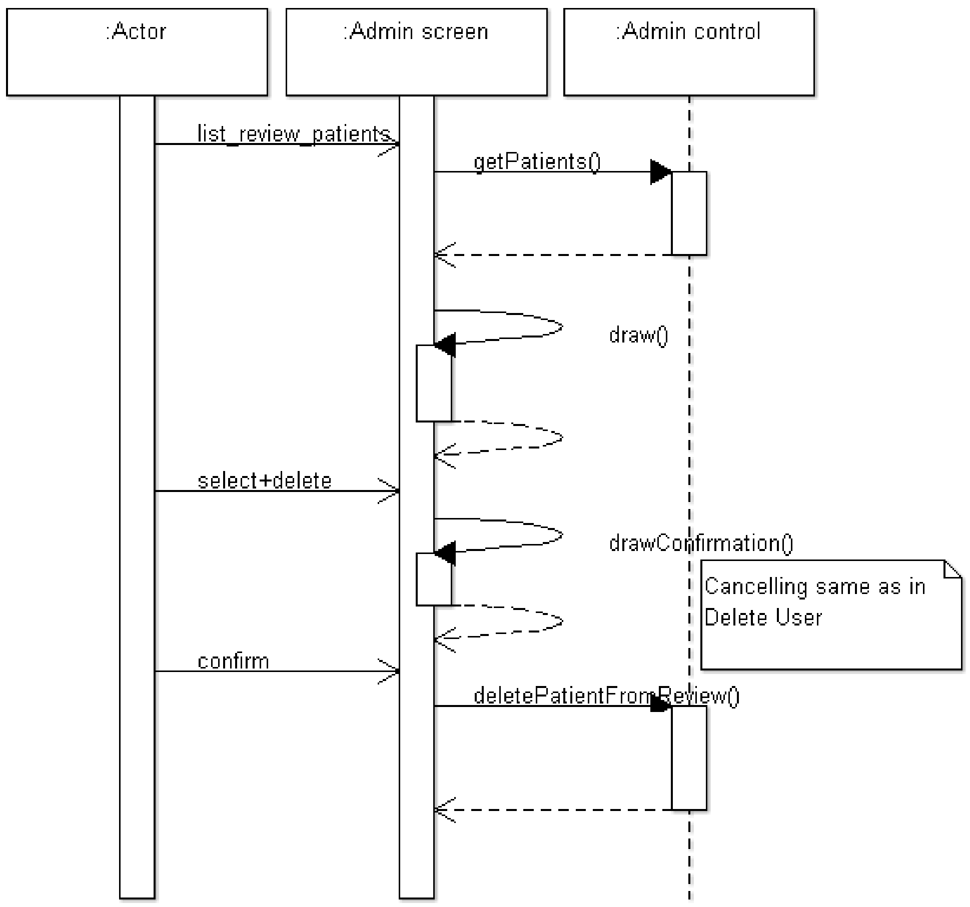
Filling in the rest of a follow up form



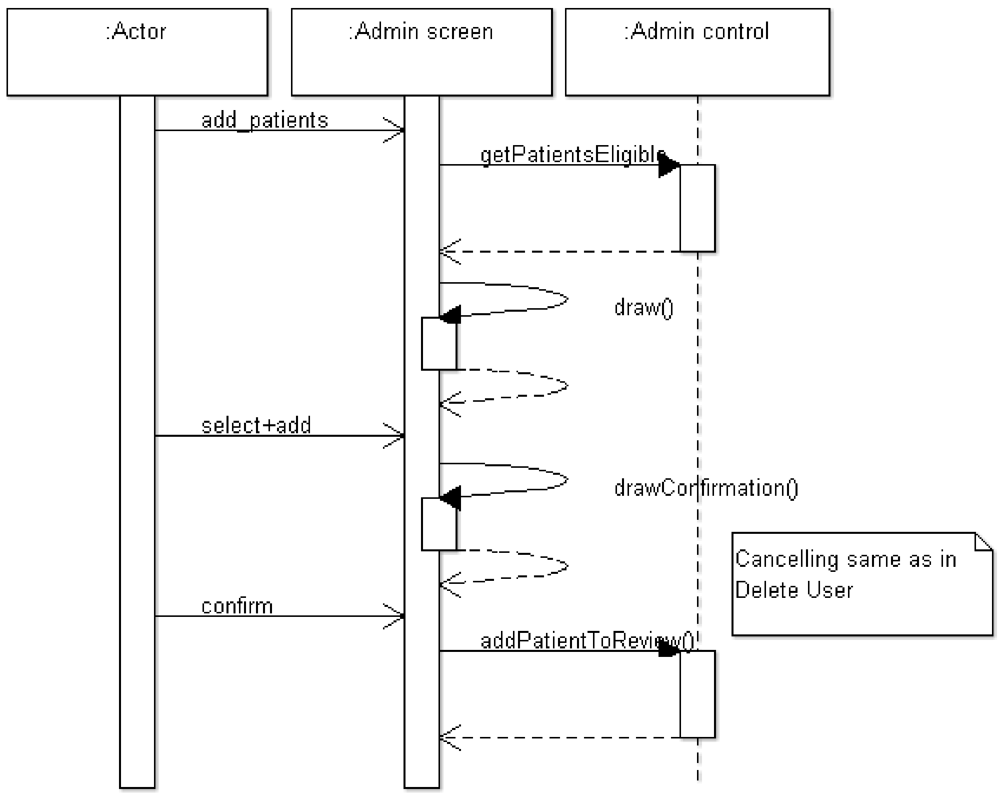
Creating and filling in a follow up form



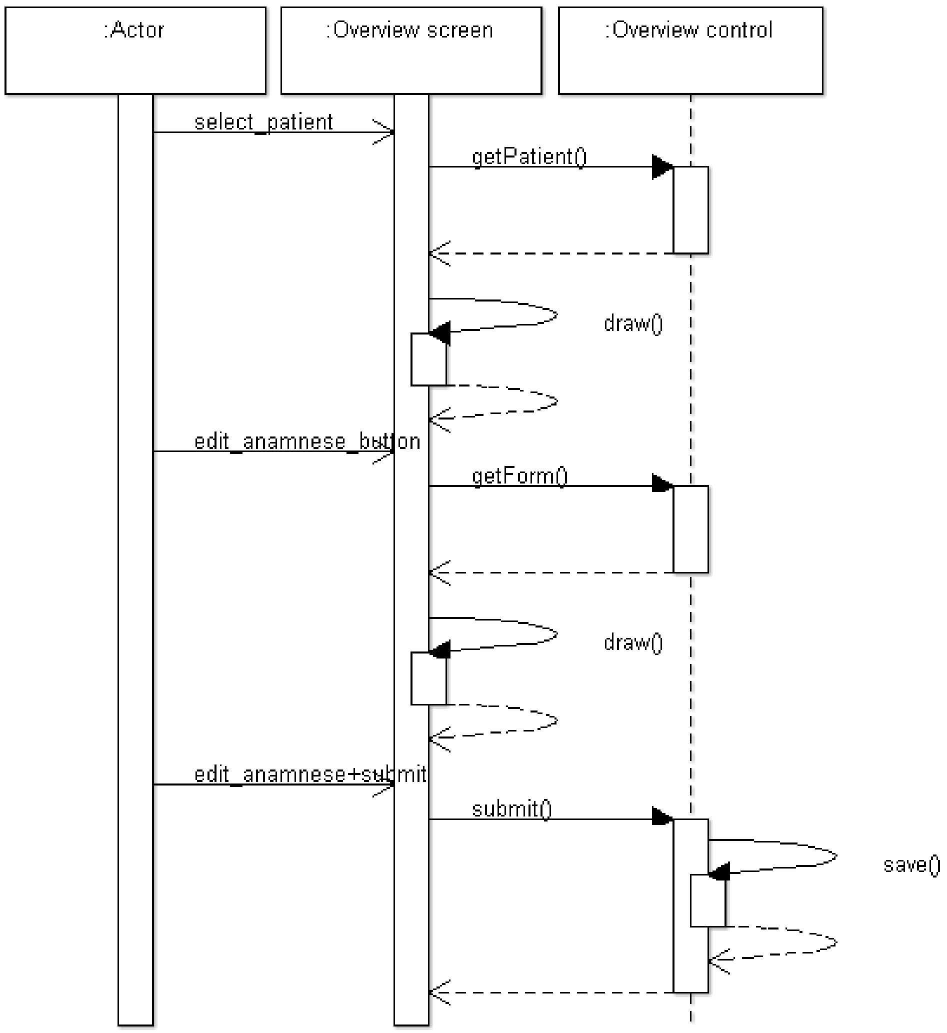
Removing a patient from the to be reviewed list



Add a patient to the to be reviewed list



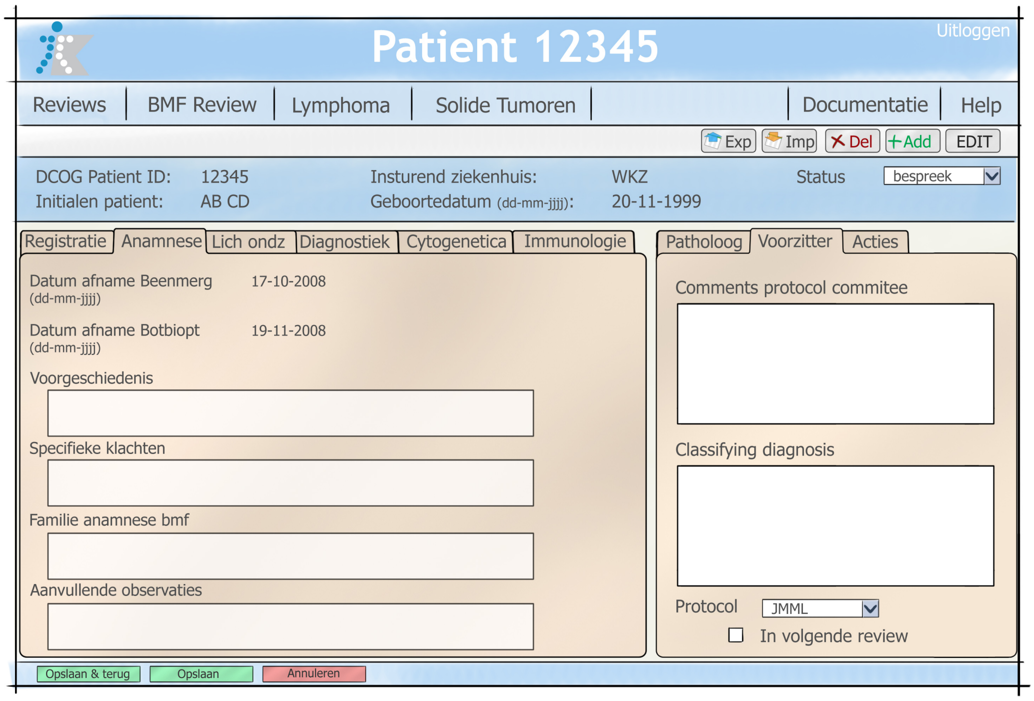
Adding or editing Anamnese



4.3.4 User interface

These images are taken from Tijsma's report to give an impression of how the application could look like.

Patient overview

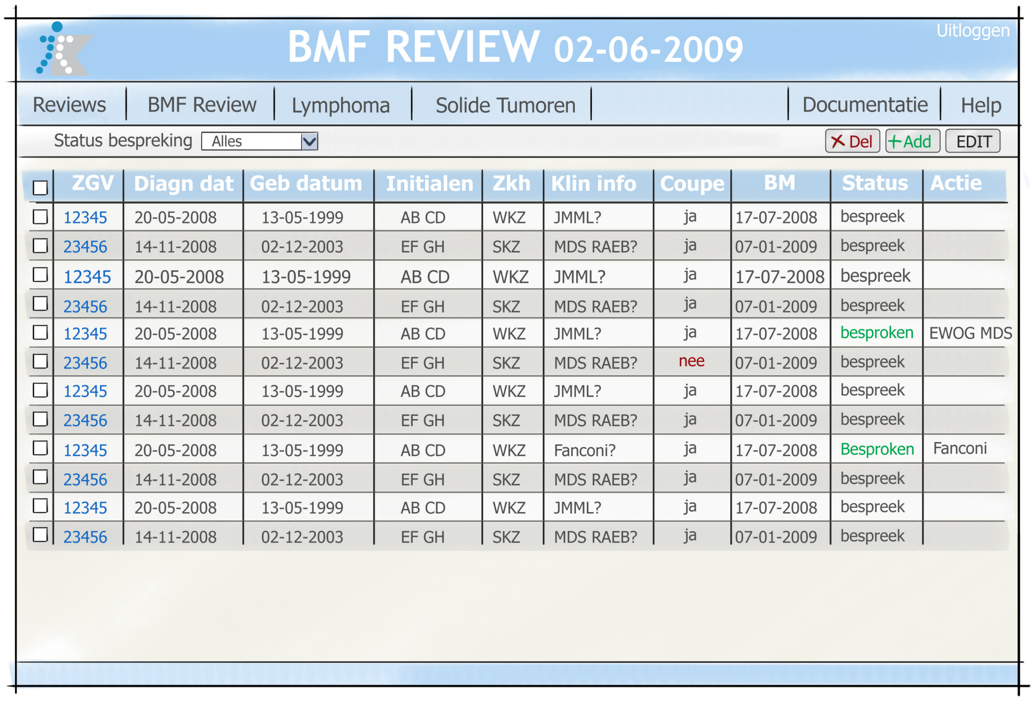


DCOG Patient ID: 12345
Initialen patient: AB CD
Instured ziekenhuis: WKZ
Geboortedatum (dd-mm-jjjj): 20-11-1999
Status: bespreek

Datum afname Beenmerg (dd-mm-jjjj): 17-10-2008
Datum afname Botbiopt (dd-mm-jjjj): 19-11-2008
Voorgeschiedenis
Specifieke klachten
Familie anamnese bmf
Aanvullende observaties

Comments protocol commitee
Classifying diagnosis
Protocol: JMML
☐ In volgende review

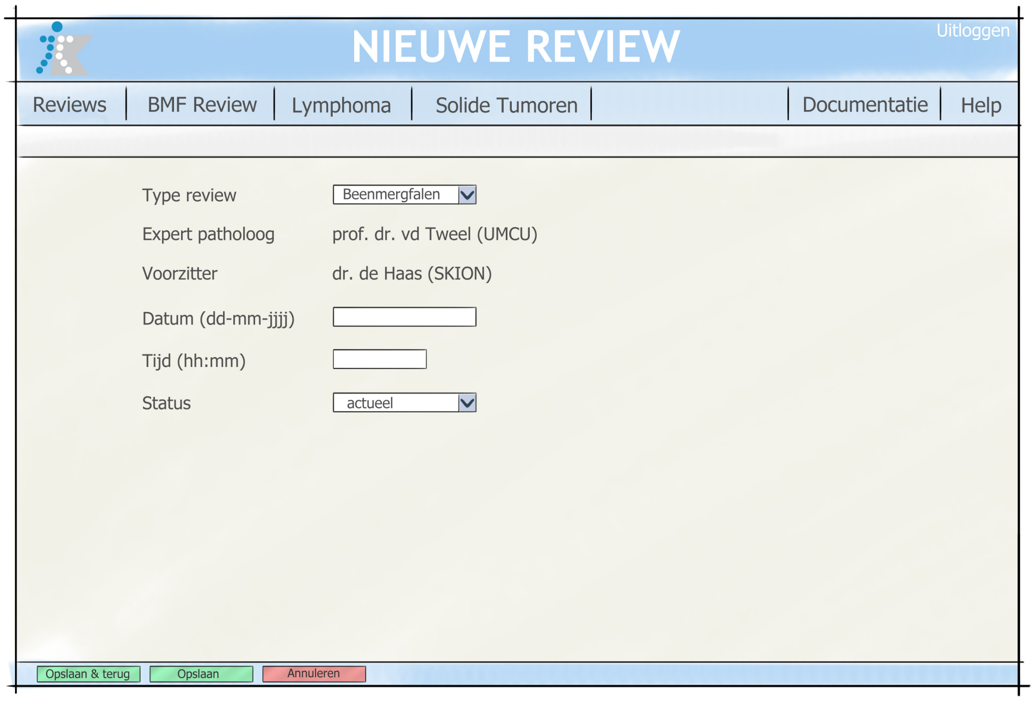
Patients overview



Status bespreking: Alles

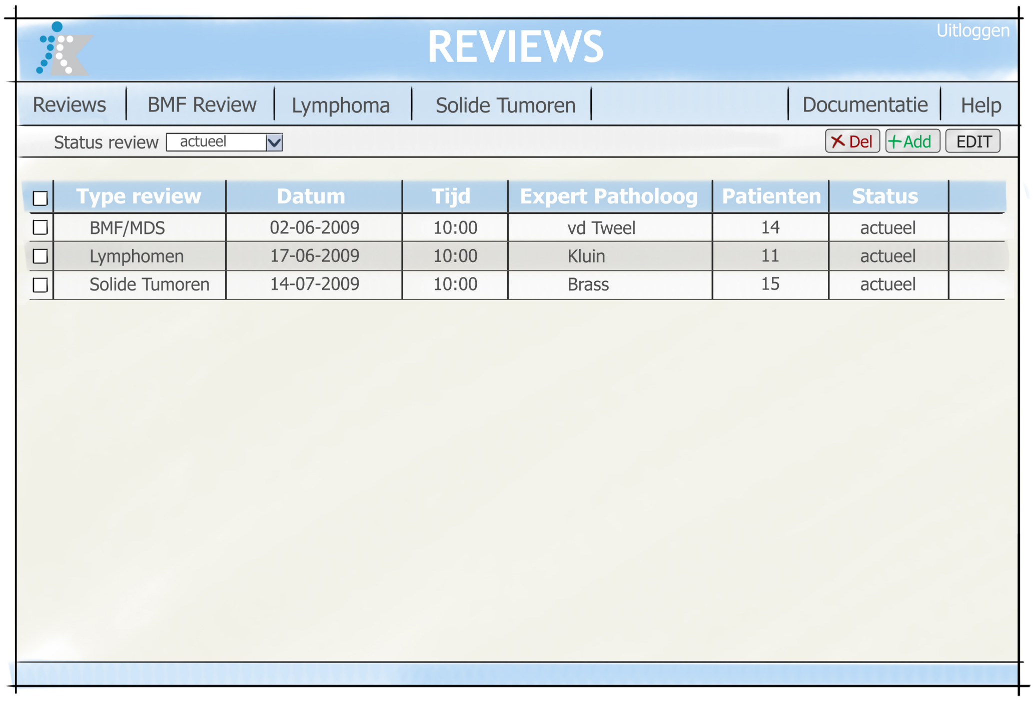
☐	ZGV	Diagn dat	Geb datum	Initialen	Zkh	Klin info	Coupe	BM	Status	Actie
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	JMML?	ja	17-07-2008	bespreek	
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	ja	07-01-2009	bespreek	
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	JMML?	ja	17-07-2008	bespreek	
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	ja	07-01-2009	bespreek	
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	JMML?	ja	17-07-2008	besproken	EWOG MDS
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	nee	07-01-2009	bespreek	
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	JMML?	ja	17-07-2008	bespreek	
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	ja	07-01-2009	bespreek	
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	Fanconi?	ja	17-07-2008	Besproken	Fanconi
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	ja	07-01-2009	bespreek	
☐	12345	20-05-2008	13-05-1999	AB CD	WKZ	JMML?	ja	17-07-2008	bespreek	
☐	23456	14-11-2008	02-12-2003	EF GH	SKZ	MDS RAEB?	ja	07-01-2009	bespreek	

Start pre-CRF



Type review	Beenmergfalen
Expert patholoog	prof. dr. vd Tweel (UMCU)
Voorzitter	dr. de Haas (SKION)
Datum (dd-mm-jjjj)	
Tijd (hh:mm)	
Status	actueel

Reviews overview

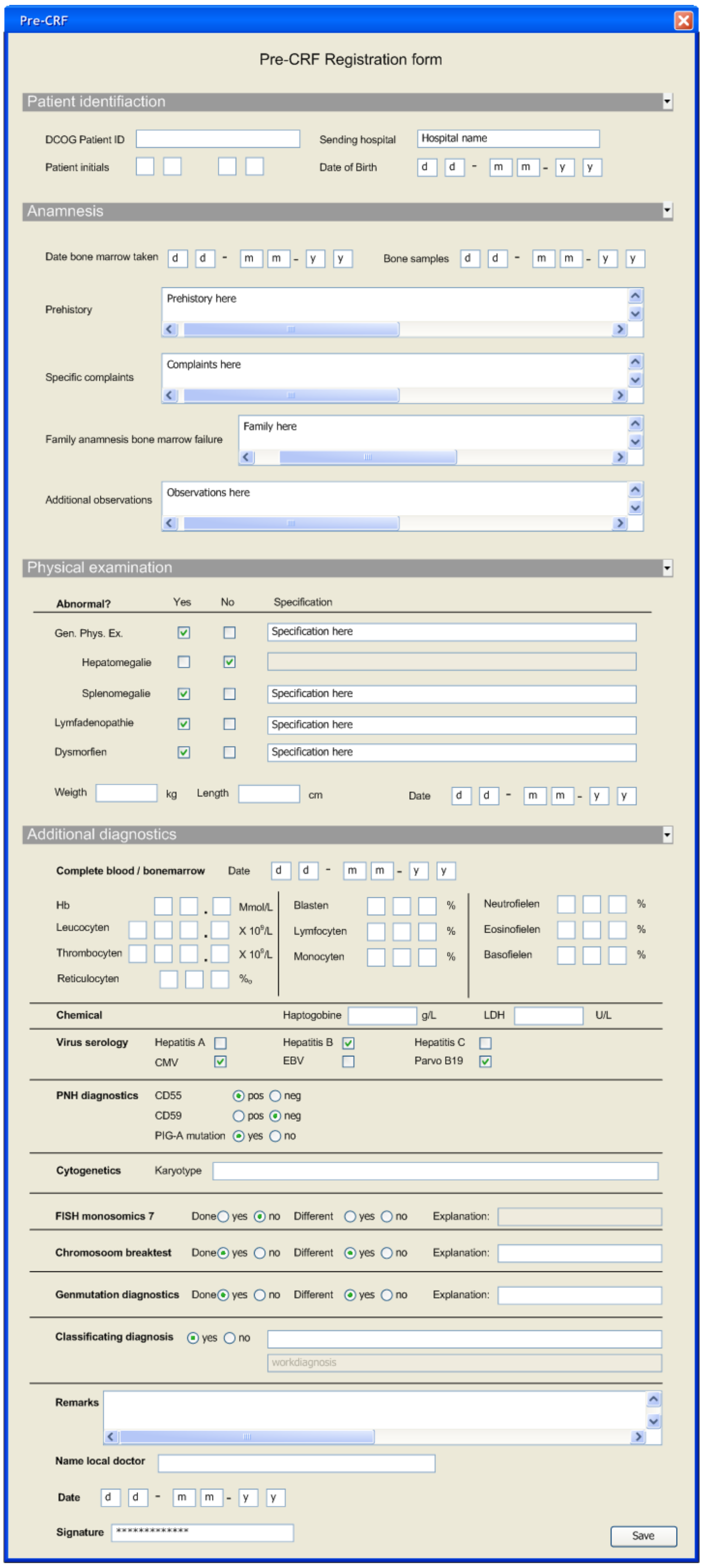


Status review actueel

☐	Type review	Datum	Tijd	Expert Patholoog	Patienten	Status
☐	BMF/MDS	02-06-2009	10:00	vd Tweel	14	actueel
☐	Lymphomen	17-06-2009	10:00	Kluin	11	actueel
☐	Solide Tumoren	14-07-2009	10:00	Brass	15	actueel

We have created the following images to give an impression of how the digital forms will look like in the application:

Pre-CRF



Pre-CRF

Pre-CRF Registration form

Patient identifiaction

DCOG Patient ID | Sending hospital: Hospital name
Patient initials | Date of Birth: d d - m m - y y

Anamnesis

Date bone marrow taken: d d - m m - y y | Bone samples: d d - m m - y y

Prehistory: Prehistory here

Specific complaints: Complaints here

Family anamnesis bone marrow failure: Family here

Additional observations: Observations here

Physical examination

Abnormal?	Yes	No	Specification
Gen. Phys. Ex.	✓		Specification here
Hepatomegalie		✓	
Splenomegalie	✓		Specification here
Lymfadenopathie	✓		Specification here
Dysmorfien	✓		Specification here

Weigth kg | Length cm | Date d d - m m - y y

Additional diagnostics

Complete blood / bonemarrow Date d d - m m - y y

Hb	Mmol/L	Blasten	%	Neutrofielen	%
Leucocyten	X 10^9/L	Lymfocyten	%	Eosinofielen	%
Thrombocyten	X 10^9/L	Monocyten	%	Basofielen	%
Reticulocyten	‰				

Chemical Haptogobine g/L | LDH U/L

Virus serology Hepatitis A | [✓ Hepatitis B] | Hepatitis C | [✓ CMV] | EBV | [✓ Parvo B19]

PNH diagnostics
CD55 [✓ pos] neg
CD59 pos [✓ neg]
PIG-A mutation [✓ yes] no

Cytogenetics Karyotype

FISH monosomics 7 Done yes [✓ no] Different yes no Explanation:

Chromosoom breaktest Done [✓ yes] no Different [✓ yes] no Explanation:

Genmutation diagnostics Done [✓ yes] no Different [✓ yes] no Explanation:

Classificating diagnosis [✓ yes] no
workdiagnosis

Remarks

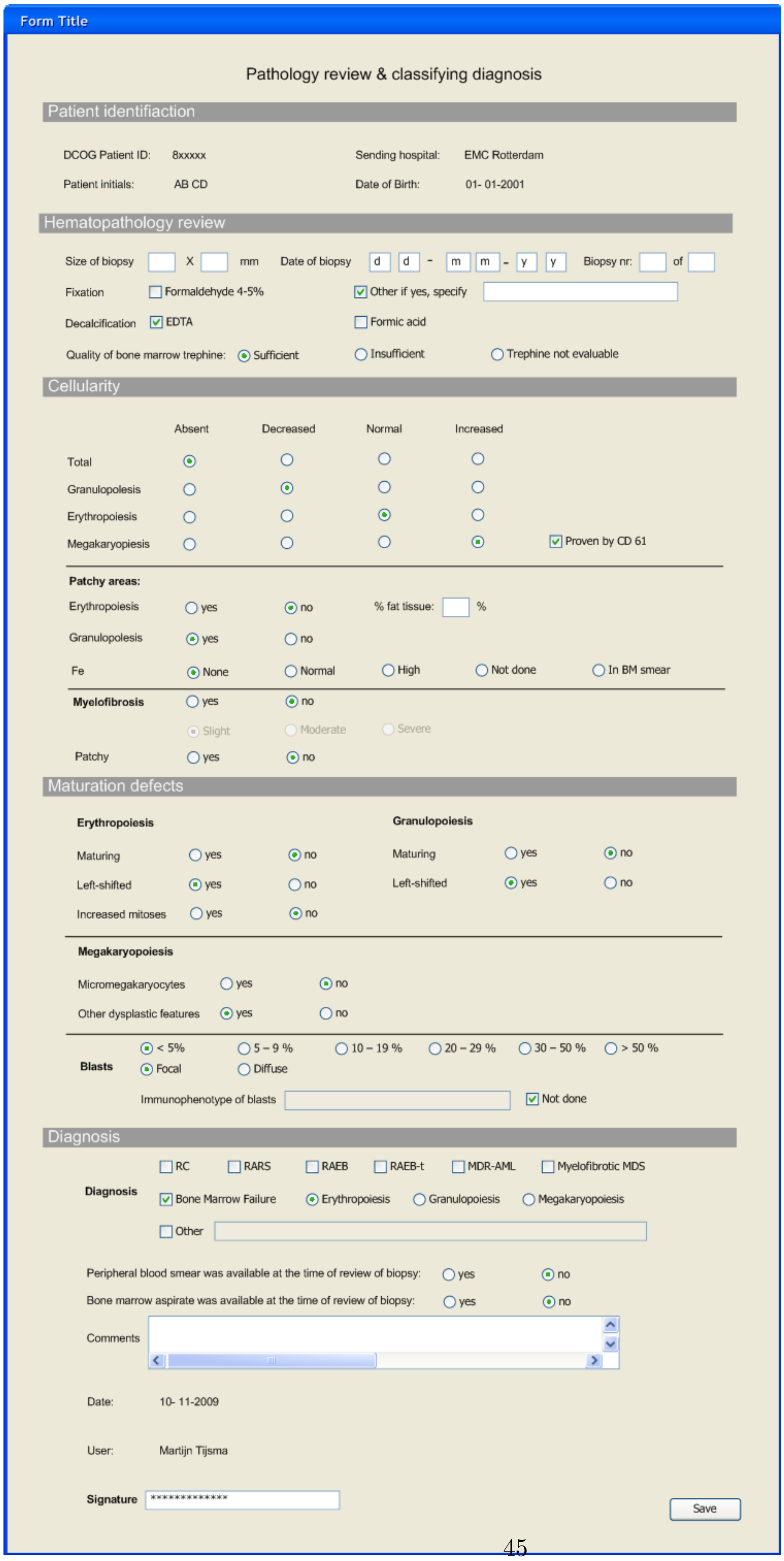
Name local doctor

Date d d - m m - y y

Signature *************

Save

Form for the pathologist



Form Title

Pathology review & classifying diagnosis

Patient identifiaction

DCOG Patient ID:	8xxxxx	Sending hospital:	EMC Rotterdam
Patient initials:	AB CD	Date of Birth:	01- 01-2001

Hematopathology review

Size of biopsy [ ] X [ ] mm   Date of biopsy d d - m m - y y   Biopsy nr: [ ] of [ ]

Fixation   Formaldehyde 4-5%   [✓ Other if yes, specify] [ ]

Decalcification   [✓ EDTA]   Formic acid

Quality of bone marrow trephine: [✓ Sufficient]   Insufficient   Trephine not evaluable

Cellularity

	Absent	Decreased	Normal	Increased	
Total	●	○	○	○	
Granulopoiesis	○	●	○	○	
Erythropoiesis	○	○	●	○	
Megakaryopiesis	○	○	○	●	[✓ Proven by CD 61]

Patchy areas:

Erythropoiesis   yes   [✓ no]   % fat tissue: [ ] %

Granulopoiesis   [✓ yes]   no

Fe   [✓ None]   Normal   High   Not done   In BM smear

Myelofibrosis   yes   [✓ no]

[✓ Slight]   Moderate   Severe

Patchy   yes   [✓ no]

Maturation defects

Erythropoiesis

Maturing   yes   [✓ no]

Left-shifted   [✓ yes]   no

Increased mitoses   yes   [✓ no]

Granulopoiesis

Maturing   yes   [✓ no]

Left-shifted   [✓ yes]   no

Megakaryopoiesis

Micromegakaryocytes   yes   [✓ no]

Other dysplastic features   [✓ yes]   no

Blasts   [✓ < 5%]   5 – 9 %   10 – 19 %   20 – 29 %   30 – 50 %   > 50 %

[✓ Focal]   Diffuse

Immunophenotype of blasts [ ]   [✓ Not done]

Diagnosis

Diagnosis   RC   RARS   RAEB   RAEB-t   MDR-AML   Myelofibrotic MDS

[✓ Bone Marrow Failure]   [✓ Erythropoiesis]   Granulopoiesis   Megakaryopoiesis

Other [ ]

Peripheral blood smear was available at the time of review of biopsy: yes   [✓ no]

Bone marrow aspirate was available at the time of review of biopsy: yes   [✓ no]

Comments [ ]

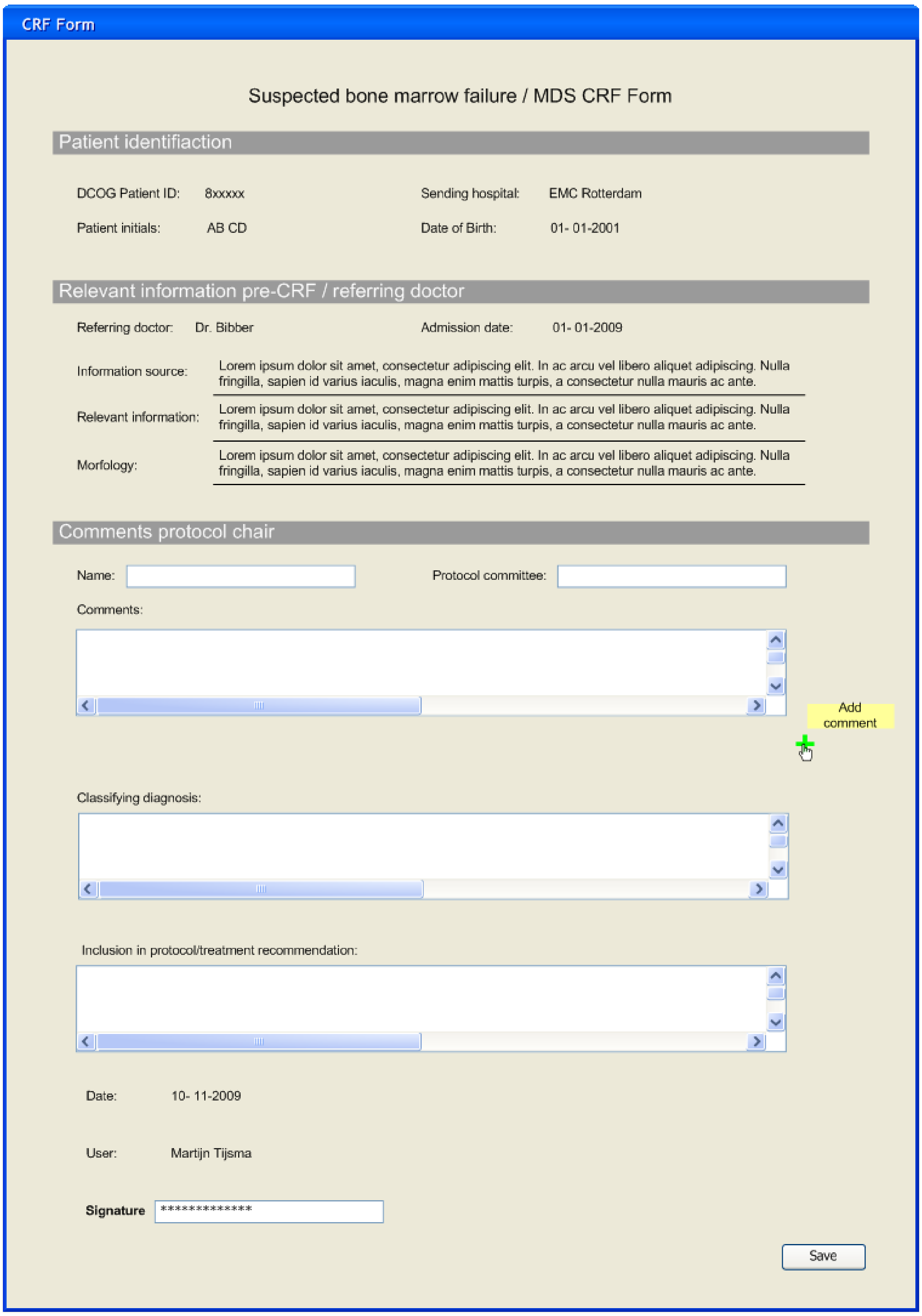
Date: 10- 11-2009

User: Martijn Tijsma

Signature **************

Save

Central Review Form


CRF Form

Suspected bone marrow failure / MDS CRF Form

Patient identifiaction

DCOG Patient ID:	8xxxxx	Sending hospital:	EMC Rotterdam
Patient initials:	AB CD	Date of Birth:	01- 01-2001

Relevant information pre-CRF / referring doctor

Referring doctor:	Dr. Bibber	Admission date:	01- 01-2009

Information source:	Lorem ipsum dolor sit amet, consectetur adipiscing elit. In ac arcu vel libero aliquet adipiscing. Nulla fringilla, sapien id varius iaculis, magna enim mattis turpis, a consectetur nulla mauris ac ante.
Relevant information:	Lorem ipsum dolor sit amet, consectetur adipiscing elit. In ac arcu vel libero aliquet adipiscing. Nulla fringilla, sapien id varius iaculis, magna enim mattis turpis, a consectetur nulla mauris ac ante.
Morfology:	Lorem ipsum dolor sit amet, consectetur adipiscing elit. In ac arcu vel libero aliquet adipiscing. Nulla fringilla, sapien id varius iaculis, magna enim mattis turpis, a consectetur nulla mauris ac ante.

Comments protocol chair

Name: Protocol committee:

Comments:

Classifying diagnosis:

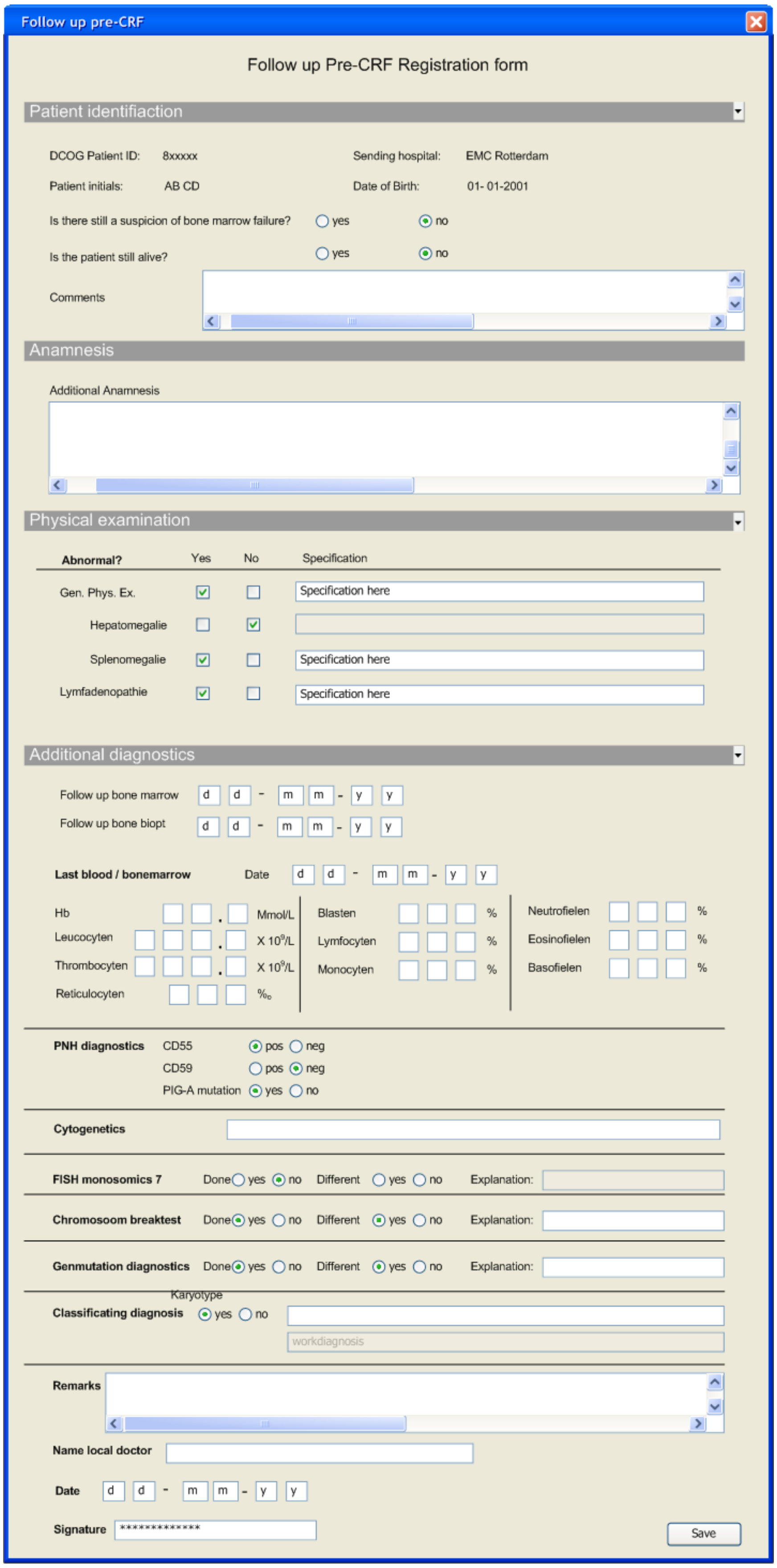
Inclusion in protocol/treatment recommendation:

Date: 10- 11-2009

User: Martijn Tijsma

Signature *************

Follow up to the pre-crf



Follow up Pre-CRF Registration form

Patient identifiaction

DCOG Patient ID:	8xxxxx	Sending hospital:	EMC Rotterdam
Patient initials:	AB CD	Date of Birth:	01- 01-2001

Is there still a suspicion of bone marrow failure? yes [✓ no]

Is the patient still alive? yes [✓ no]

Comments

Anamnesis

Additional Anamnesis

Physical examination

Abnormal?	Yes	No	Specification
Gen. Phys. Ex.	✓		Specification here
Hepatomegalie		✓	
Splenomegalie	✓		Specification here
Lymfadenopathie	✓		Specification here

Additional diagnostics

Follow up bone marrow d d - m m - y y

Follow up bone biopt d d - m m - y y

Last blood / bonemarrow Date d d - m m - y y

Hb	Mmol/L	Blasten	%	Neutrofielen	%
Leucocyten	X 10^9/L	Lymfocyten	%	Eosinofielen	%
Thrombocyten	X 10^9/L	Monocyten	%	Basofielen	%
Reticulocyten	‰				

PNH diagnostics

CD55	[✓ pos] neg
CD59	pos [✓ neg]
PIG-A mutation	[✓ yes] no

Cytogenetics

	Done	Different	Explanation:
FISH monosomics 7	yes [✓ no]	yes no	
Chromosoom breaktest	[✓ yes] no	[✓ yes] no	
Genmutation diagnostics	[✓ yes] no	[✓ yes] no	

Karyotype

Classificating diagnosis [✓ yes] no

workdiagnosis

Remarks

Name local doctor

Date d d - m m - y y

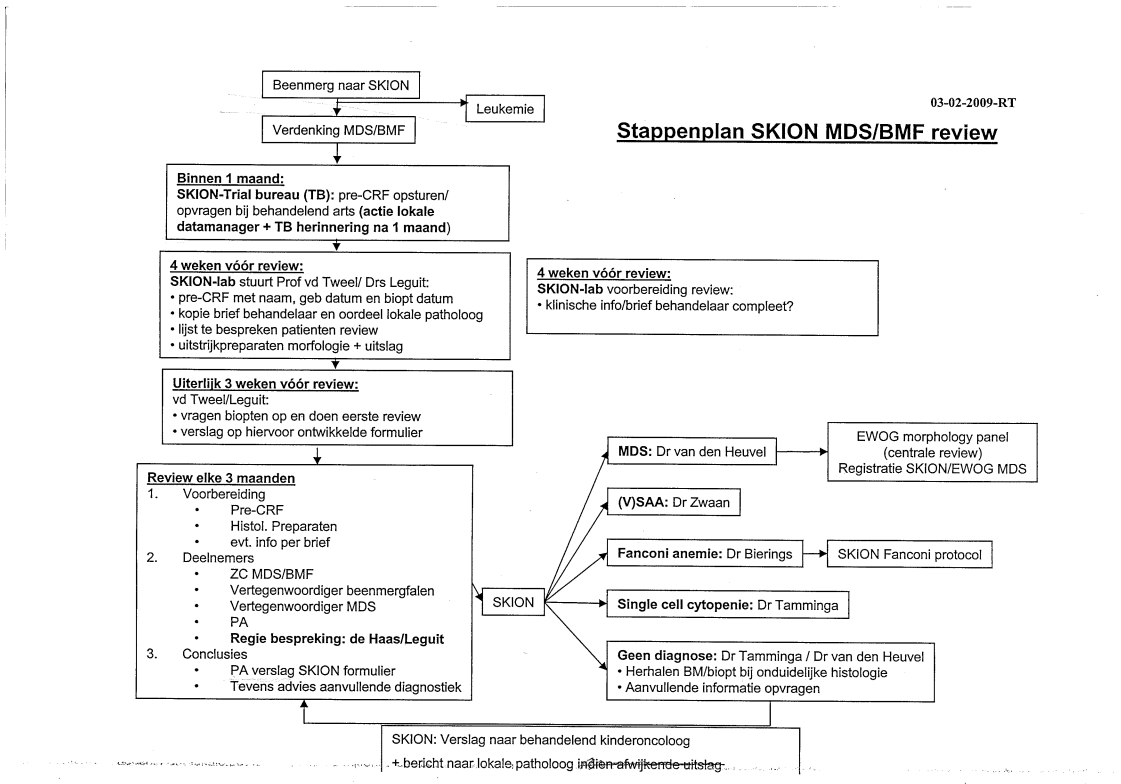
Signature *************

Save

Chapter 5

Appendix

The next pages contain the different forms as they are in use now at the Central Review at SKION. The first scan is the flow of events in the current MDS/BMF review.


03-02-2009-RT
Stappenplan SKION MDS/BMF review
Beenmerg naar SKION
Leukemie
Verdenking MDS/BMF
Binnen 1 maand:
SKION-Trial bureau (TB): pre-CRF opsturen/ opvragen bij behandelend arts (actie lokale datamanager + TB herinnering na 1 maand)
4 weken vóór review:
SKION-lab stuurt Prof vd Tweel/ Drs Leguit:
• pre-CRF met naam, geb datum en biopt datum
• kopie brief behandelaar en oordeel lokale patholoog
• lijst te bespreken patienten review
• uitstrijkpreparaten morfologie + uitslag
4 weken vóór review:
SKION-lab voorbereiding review:
• klinische info/brief behandelaar compleet?
Uiterlijk 3 weken vóór review:
vd Tweel/Leguit:
• vragen biopten op en doen eerste review
• verslag op hiervoor ontwikkelde formulier
Review elke 3 maanden
1. Voorbereiding
• Pre-CRF
• Histol. Preparaten
• evt. info per brief
2. Deelnemers
• ZC MDS/BMF
• Vertegenwoordiger beenmergfalen
• Vertegenwoordiger MDS
• PA
• Regie bespreking: de Haas/Leguit
3. Conclusies
• PA verslag SKION formulier
• Tevens advies aanvullende diagnostiek
SKION
MDS: Dr van den Heuvel
EWOG morphology panel (centrale review) Registratie SKION/EWOG MDS
(V)SAA: Dr Zwaan
Fanconi anemie: Dr Bierings
SKION Fanconi protocol
Single cell cytopenie: Dr Tamminga
Geen diagnose: Dr Tamminga / Dr van den Heuvel
• Herhalen BM/biopt bij onduidelijke histologie
• Aanvullende informatie opvragen
SKION: Verslag naar behandelend kinderoncoloog
+ bericht naar lokale patholoog ~~indien afwijkende uitslag~~

Verdenking beenmergfalen/MDS

Formulier 1, pre-registratie

Stichting Kinderoncologie Nederland
SKION
Dutch Childhood Oncology Group
DCOG

Patiënt identificatie

DCOG patiënt ID ☐☐☐☐☐ Insturend ziekenhuis: ....

Initialen patiënt ☐☐ ☐☐ Geboortedatum (dd.mm.jjjj): _ _ . _ _ . _ _ _ _

Anamnese

Datum afname (dd.mm.jjjj) beenmerg: _ _ . _ _ . _ _ _ _ Botbiopt: _ _ . _ _ . _ _ _ _

Voorgeschiedenis ....

Specifieke klachten ....

Familie anamnese beenmergfalen ....

Aanvullende observaties ....

Lichamelijk onderzoek

Abnormaal J/N ?	Ja	Nee	Zo ja, specificeer:
Alg. Lich. Onderzoek	☐	☐	....
Hepatomegalie	☐	☐	....
Splenomegalie	☐	☐	....
Lymfadenopathie	☐	☐	....
Dysmorfiën	☐	☐	....

Gewicht : ☐☐☐ kg Lengte: ☐☐☐ cm Datum: _ _ . _ _ . _ _ _ _

Aanvullende diagnostiek

Volledig bloedbeeld/beenmerg: Datum: (dd.mm.jjjj): _ _ . _ _ . _ _ _ _

Hb	☐☐.☐ mmol/L	Blasten	☐☐☐ %	Neutrofielen	☐☐☐ %
Leucocyten	☐☐☐.☐ x 10^9/L	Lymfocyten	☐☐☐ %	Eosinofielen	☐☐☐ %
Thrombocyten	☐☐☐.☐ x 10^9/L	Monocyten	☐☐☐ %	Basofielen	☐☐☐ %
Reticulocyten	☐☐☐ ‰				

Chemie: Haptoglobine: g/L LDH: U/L

Virus serologie:	Hepatitis A:	Hepatitis B:	Hepatitis C:
(IgG/IgM: pos/neg)	CMV:	EBV:	Parvo B19:

PNH diagnostiek: CD55 ☐ pos ☐ neg CD59 ☐ pos ☐ neg PIG-A mutatie: ☐ ja ☐ nee

Cytogenetica: Karyotype: ....

FISH monosomie 7	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....
Chromosoom breuktest	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....
Genmutatie diagnostiek	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....

Classificerende diagnose: ☐ ja ....

☐ nee: **werkdiagnose:** ....

Opmerkingen : ....

Naam insturende arts ....

Datum (dd.mm.jjjj) _ _ . _ _ . _ _ _ _ Handtekening..

Stuur origineel naar SKION

Suspected bone marrow failure/MDS

Pathology review & classifying diagnosis

Stichting Kinderoncologie Nederland SKION Dutch Childhood Oncology Group DCOG

Patient identification

DCOG patient ID	☐☐☐☐☐	Referring center:	
Initials patient (first name/surname)	☐☐ ☐☐	Date of birth (dd.mm.yyyy):	_ _ . _ _ . _ _ _ _

Hematopathology review

Size of biopsy	_ _ x _ _ mm	**Date of biopsy (dd.mm.yyyy)**	_ _ . _ _ . _ _ _ _
Fixation	☐ Formaldehyde 4-5%	☐ other if yes, specify:	**Biopsy nr:** __ **of** ____
Decalcification	☐ EDTA	☐ Formic acid	
Quality of bone marrow trephine	☐ sufficient	☐ insufficient	☐ trephine not evaluable

Cellularity

	absent	decreased	normal	increased	
Total	☐	☐	☐	☐	
Granulopoiesis	☐	☐	☐	☐	
Erythropoiesis	☐	☐	☐	☐	
Megakaryopiesis	☐	☐	☐	☐	☐ proven by CD 61

Patchy areas:	Erythropoiesis: ☐ no	☐ yes	Granulopoiesis: ☐ no	☐ yes	% fat tissue: ______
Fe	☐ None	☐ Normal	☐ High	☐ not done	☐ in BM smear
Myelofibrosis	☐ no ☐ yes:	☐ slight	☐ moderate	☐ severe	Patchy: ☐ no ☐ yes

Maturation defects

Erythropoiesis		**Granulopoiesis**		**Megakaryopoiesis**	
Maturing	☐ no ☐ yes	Maturing	☐ no ☐ yes	micromegakaryocytes	☐ no ☐ yes
Left-shifted	☐ no ☐ yes	Left-shifted	☐ no ☐ yes	Other dysplastic features	☐ no ☐ yes
Increased mitoses	☐ no ☐ yes				

Blasts	☐ < 5%	☐ 5 – 9 %	☐ 10 – 19%	☐ 20 29 %	☐ 30 – 50 %	☐ > 50 %
	☐ focal	☐ diffuse	Immunophenotype of blasts: ______________________			☐ not done

Diagnosis

Diagnosis	☐ RC	☐ RARS	☐ RAEB	☐ RAEB-t	☐ MDR-AML	☐ Myelofibrotic MDS
	☐ Bone Marrow Failure:		☐ Erythropoiesis	☐ Granulopoiesis	☐ Megakaryopoiesis	
	☐ other, please specify: ______________________________					

Peripheral blood smear was available at the time of review of biopsy: ☐ no ☐ yes

Bone marrow aspirate was available at the time of review of biopsy: ☐ no ☐ yes

Comments:

Date: _ _ - _ _ - _ _ _ _

Name: ______________________________ Signature: ______________________________

Stichting Kinderoncologie Nederland
SKION
Dutch Childhood Oncology Group
DCOG

Suspected bone marrow failure / MDS

Classifying diagnosis

Patient identification

DCOG patient ID :	Referring center :
Initials patient :	Date of birth : dd/mm/yyyy

Relevant information pre-CRF / referring dr

Referring dr :	Admission date : dd/mm/yyyy
Information source:	Relevant information:
Morfology :	

Hematopathology review

Pathologist :	Date review : dd/mm/yyyy
Report:	
Conclusion:	

Comments protocol chair

Name :	Protocol committee :
Classifying diagnosis:	
Inclusion in protocol / treatment recommentdation:	

Date : dd/mm/yyyy

Name : Signature :

Verdenking beenmergfalen/MDS

Formulier 2, follow up pre-registratie (na 6 maanden)
of jaar of 1½ jaar

Stichting Kinderoncologie Nederland
SKION
Dutch Childhood Oncology Group
DCOG

Patiënt identificatie

DCOG patiënt ID	☐☐☐☐☐	Insturend ziekenhuis:	....
Initialen patiënt (voornaam/achternaam)	☐☐ ☐☐	Geboortedatum (dd.mm.jjjj):	_ _ . _ _ . _ _ _ _

Bestaat er nog verdenking op beenmergfalen? Ja ☐ Nee ☐ | **Is de patiënt nog in leven?** Ja ☐ Nee ☐

Toelichting:..

Anamnese

Evt. aanvullende anamnese ....
....

Lichamelijk onderzoek (alleen veranderingen tov 1e CRF doorgeven)

Abnormaal J/N ?	**Ja**	**Nee**	**Zo ja, specificeer:**
Alg. Lich. Onderzoek	☐	☐	....
Hepatomegalie	☐	☐	....
Splenomegalie	☐	☐	....
Lymfadenopathie	☐	☐	....

Aanvullende diagnostiek

Vervolg beenmerg (dd.mm.jjjj)	_ _ . _ _ . _ _ _ _	_ _ . _ _ . _ _ _ _	_ _ . _ _ . _ _ _ _
Vervolg botbiopt (dd.mm.jjjj)	_ _ . _ _ . _ _ _ _	_ _ . _ _ . _ _ _ _	_ _ . _ _ . _ _ _ _

Laatste bloedbeeld (dd.mm.jjjj) _ _ . _ _ . _ _ _ _

Hb	☐☐.☐ mmol/L	Blasten	☐☐☐ %	Neutrofielen	☐☐☐ %
Leucocyten	☐☐☐.☐ x 10^9/L	Lymfocyten	☐☐☐ %	Eosinofielen	☐☐☐ %
Thrombocyten	☐☐☐.☐ x 10^9/L	Monocyten	☐☐☐ %	Basofielen	☐☐☐ %
Reticulocyten	☐☐☐ ‰	MCV	☐☐☐ (fL)		

PNH diagnostiek: CD55 ☐ pos ☐ neg CD59 ☐ pos ☐ neg PIG-A mutatie: ☐ ja ☐ nee

Cytogenetica: Karyotype: ....

FISH monosomie 7	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....
Chromosoom breuktest	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....
Genmutatie diagnostiek	Verricht: ☐ ja ☐ nee	Afwijkend: ☐ ja ☐ nee	Toelichting: ....

Classificerende diagnose: ☐ ja ....
☐ nee: **werkdiagnose:** ....

Opmerkingen : ....

Naam insturende arts ....

Datum (dd.mm.jjjj) _ _ . _ _ . _ _ _ _ Handtekening..

Stuur origineel naar SKION

Chapter 6
Appendix NEN norms

This appendix will cover the NEN[1] norms which the application is required to fulfill. Many of the requirements stated below are already stated in this Requirements Analysis Document; those who are not are expected to be taken care of during implementation.

- Registration of (authorized) users accessing the application, including time spend, ip address and action specification. This will be ensured by keeping a log.
- Create regular backups stored on a separate system. This is a task for the server providers / administrators.
- Saving data as long as the Dutch law requires. For the application, this concerns only the logs.
- Secured connection for exchanging data from and to the application. This will be ensured by using SSL[2]
- Ensure the confidentiality of data during transportation. This is a feature of SSL.
- Ensure the delivery of messages. This is a feature of the transport protocol.
- Ensure the integrity of messages and data. This is also a feature of the transport protocol.
- Ensure that a sent message is unrefutable coming from a specific individual.
- Access to the application must be managed through user registration. The SKION review manager will create the users.
- Non-registered users may only access public data and facilities. The application has no public data or facilities, so this is not applicable.
- Registered users must have a unique user identification for personal use.
- User credentials must be chosen such that no authorisation level can be determined from them. This means that user names should be chosen without mentioning i.e. pathologist, manager or administrator. This is a requirement for the SKION review manager, who will create the new users.
- A password authentication must be used, at least, to verify the identity of a user. For the approach used in this application, see section 3.3.12.
- Automatic connections with computersystems must be authenticated.
- Data which is entered into the system must be validated on correctness, completeness and date. This will be enabled by being able to review the data entered into the system.
- The application must minimize the risk of data corruption by running checks on the data mutations. All data that is entered to the system, will be checked for some basic conditions.

[1]Nederlandse Normen
[2]Secured Sockets Layer, an encryption method for the traffic between browser and server

- Data used for testing purposes, must be secured and managed.
- The usage of original data is allowed, however the data must be made anonymous.

Appendix B
Architectural Design Document

Digital Review
Architectural Design Document

Architectural decisions made for the webapplication for SKION

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
May 8, 2009

Contents

Chapter 1
Introduction

This document describes, in detail, the software system to be build for SKION. Furthermore it gives a functional decomposition of the system components. Aside from that, this document lists and explain any technologies used with the goal of implementing the features listed in the functional requirements document. We expect the reader of this document to have some understanding of the system, by having read the Requirements Analysis Document.

1.1 Overall system architecture and design goals

The overall architecture is designed to be highly expandable in an easy way.

1.2 Overview of the document

In chapter 2 we give a general overview of the system. It should give the reader a general idea of the system's structure and purpose, before a more technical and detailed design is presented in the following chapters. The design considerations and decisions, influencing the rest of the design document, are presented in chapter 2 as well.

Chapter 3 discusses the relation between the various software components of the system and the hardware they run on. It describes in detail the specifications of the hardware used in the system, and why certain parts of the software need to run on a specific piece of hardware.

In chapter 4 we give a thorough description of the way data is handled in the system. We consider several solutions to data design issues, so that we can create a concrete design. Important considerations in this chapter concern data storage, data management, data integrity and data performance. Basically, this chapter shows how data is stored, presented and used in the system.

Chapter 5 provides an overview of any problems that may arise from the concurrent character of the system. This means that we discuss concurrency in any form and on any level, and give solutions to the emerging problems. On a higher level, we examine the concurrent character of the way subsystems communicate, but we inspect local concurrent tasks as well. This chapter also contains information on software control, i.e. the way control and messages flow during execution.

Chapter 6 describes how the system deals with security issues and what access policies it enforces. Security on both the physical and the software level is discussed.

Perhaps the most important chapter, is chapter 7. It describes in detail all components of the system and the way they communicate. All technical details of the subsystems can be found in the diagrams and descriptions throughout this chapter. However, it should be noted that we give no implementation details. We will present those in the technical design document.

The last chapter, chapter 8, deals with the boundary conditions of the system. Roughly, it describes how the system should act in the situation of system initialization, termination and failure. By investigating these special cases beforehand, most threats to a correct functioning of the system can be eliminated.

1.3 Design priorities

1.3.1 Priorities

During the design process, we kept the following priorities in mind.

Ease of use The users of the system don't have a lot of time to learn how to use a new application, so the application should be very easy to use.

Ease of implementation We have only limited time to implement the proof of concept. Also, the system must be extensible, to be easily extended in the future. The design must stay simple so that the implementation can be done quickly.

Security The system is a medical application, with sensitive patient information, so it should be very secure. Also, the digital signature is an important aspect of the system.

Robustness The system should be robust, so no weird things can happen during operation. And if something goes wrong, it should be handled nicely and correctly.

1.3.2 Tradeoffs

Many of the listed priorities are in conflict with each other. Therefore, some trade-offs had to made.

Ease of use and security

Security usually makes a program more complex to use. Since the users are generally not very experienced with computers, the security must not make the program so complex that it is annoying to use. Security is absolutely necessary. Ease of use is more important though, since the application will be used more and more frequently. We'll have to make sure that all security requires as little as possible from anyone that uses the system to make sure that the system is easy to use.

Ease of implementation and robustness

The system must be operational at all times, even if network lines are down. However, when the system gets more robust, it also gets more complex. Ease of implementation is slightly more important for us, since we have a deadline preventing us from using really sophisticated technologies to facilitate robustness. All robustness that we can implement in this time must be implemented though.

Ease of implementation and security

Security makes programming an application more complex. Since we have a deadline, we can't implement the best security possible, since this would take a considerable amount of time. Therefore, ease of implementation has a higher priority.

Chapter 2
System overview

2.1 Model, View and Controller (MVC)

Building the system, the design pattern of Model, View and Controller will be applied. This pattern is first chosen because of the different actors which will be active in the system and secondly the different sources of data which have to be controlled apart from each other. The different roles of the pattern can be outlined as follows:

- Model
  The Model package will contain all the data. Objects in this package will give the raw data meaning, by representing the raw data as objects. The methods in this package will only recieve function calls and does not perform any function calls to the View or Controller.

- View
  The View package will represent the User Interface for the different users. The View will not make any call to the Model functions, but the controller will notify the View if updates are available. And the View will notify the controller if information needs to be passed through to the Model.

- Controller
  The Controller is the glue between the Model and the View. It facilitates the communication between the View and the Model in such a way that View and Model do not nessecary need to know the existance of eachother.

A separation like this will ensure that, should the need arise, a totally different User Interface can be "swapped" in. For example, if a new User Interface is eligible, all that has to be done is write the appropriate View classes. In the same manner a different Model can be used without requiring the View to change at all, while the Controller will only require minor to change. One might decide one day that instead of a Database, a XML file-based approach is more desired. By creating the Model classes that do such, and updating the Controller in minor places this can be achieved in a relatively short time period. Should a different Controller be required (e.g., when different sets of behavior is desired), one only needs to implement the Controller classes. Neither the View or the Model would be affected by such a change.

2.2 Design decisions

Because there are several versions of the Model, View and Controller pattern, a choice had to be made which version we would implement. The following possibilitites have been considered:

1. Implementation of the MVC pattern as a Passive Model. In this implementation, objects in the model are completely unaware of being used in the MVC functionality. This implementation is extremely useful in webapplications which implement MVC, especially with PHP objects, because no state is retained between requests on those objects.

2. Implementation of the Command pattern within the MVC pattern. The Command pattern provides an application with unlimited undo/redo functionality.

After some discussion, the first implementation has been chosen. The arguments for this choice are quite obvious; the first implementation is very suitable for working with PHP and the application does not require unlimited undo/redo operations as one of the top priorities. Also, it may not always possible to undo an operation. For example when a digital signature has closed and secured data from a form, it may not be possible to undo this operation.

Chapter 3
Hardware and Software mapping

This section takes a brief look at the hardware that makes up the system and how the software runs on that hardware. This section is kept fairly short since there are very little requirements that have to be fulfilled for the actual hardware that the system is to be run on.

3.1 Overview

The deployment situation of a webapplication is a client-server architecture. For the server, it is desirable that there is full control over both the hardware and the installed software. This already fails for the proof of concept, which will be hosted on a server at the TU Delft, over which we have little to say. The clients will use there web browsers, so they are likely to be already present, resulting in limited control over the installed software and none at all over the hardware.

3.1.1 Server

The server that is used for hosting the web application will be equipped as a suitable server. For this reason a single machine to run the software, handle the connections and host the database is used. Especially since the total amount of users connecting at any given time is expected to be quite low this should give a more then sufficient performance. When deployed, this server will also have a firewall setup to increase security against rogue connections and hacking attempts.

3.1.2 Client

As mentioned, it is not exactly known what hardware and software the clients run, as all the client hardware will be provided by SKION and the hospitals themselves. It is known that hardware used at SKION and in hospitals is not brand new. As a minimum, Internet Explorer 6 or higher, Firefox 2 or higher or Safari 2 or higher, is required. There will be no mobile version of the application.

3.1.3 Connections

To allow multiple clients to connect to the server at any one time, the server itself is hooked up directly to the review database, though seperate connections have to be made to allow data transfer between the server and the SAP and ProMISe database. Connections with these databases will only be made when data is actually needed by the client. Clients can connect to the server using any connection method that is available to them, these will include both dial-up and broadband connections.

Chapter 4
Data management

This chapter will briefly go over the considerations that have to be made about the data that is used in the system. This includes handeling and storage. Subjects that will be discussed are the data storage, structure, integrity, load and performance, persistance, backup possibilities and extensibility.

4.1 Physical data storage

Since most of the data used in the system is already stored in seperate databases, there is no need to save a lot of data directly on the server. The server however, needs to consider the possibility that data can't be currently sent to the other databases, due to a connection problem for example. When this kind of failure occurs, the data that couldn't be sent has to be saved on the server aswell. But since so little information will actually be sent over a larger period of time, this will not change the fact that still the same hardware and software can be used on the server. This results in the usage of a MySQL database on the server, which will be filled with information needed for the review and data that couldn't be sent by the server at that moment.

4.2 Data structure

The choice for MySQL automatically means that the database will use a relational scheme to store all the data. In this scheme, all the models, as described in the system decomposition, will be mapped to their own table. Every object class will get its own identifier field to make it unique and allow for easy retrieval of specific object instances. For the relations between objects one can generally distinguish the following situations:
One-To-Many
Object A belongs to object B, Object B has a list of objects of the same type as A. This type of relation can be modelled using a identifier field in object A pointing to the id of object B.
One-To-One
Basically a limited One-To-Many relation. Object A belongs to object B and vice versa. This can be handled in the same way as a One-To-Many relation, but there should also be a check to make sure that no more then one object can be related to any other object. In practice this kind of relation doesn't happen very often. When it does, it is often a sign that the database design could be improved upon.
Many-To-Many
Object A can be associated to any objects of the type of B and object B can be associated with to any objects of the type of A. When implementing this relation in a relational database it is customary to use a third link table which will hold a reference to the id of both A and B. Advanced relations can be build by adding additional data to this table.

4.3 Referential integrity

The most important part of any database system is to keep the data consistent and uncorrupted. One part of this is the question of referential integrity. When an object (or row actually) in a table is linked to an object in another table by using a foreign key, the first object can be potentially orphaned when the second object is deleted without updating any present links. To make sure this doesn't happen one can use the extended relationship capabilities that were introduced in MySQL 5 to automatically cascade any required changes across the tables.

4.4 Estimated load and performance

Since only few data is transferred to and from the server when preparing a review, this will probably not take a lot of load, compared to the amount of data transfered during an actual review. During this review however, data from the MySQL, SAP and ProMISe database will be read, but not editted. Therefore the estimated load will be not that high for the server that will be used. The performance however could be an issue, but as long as the size and amount of calls to the SAP and ProMISe database are small, the client's system should be sufficient to keep the performance high. For improvement of performance, some of the data that is searched by the client could be temporarily saved up till a certain amount of space. When, during a Review, a call is done to same data; instead of a call to the server, a call to the local data will be made and this will save a lot of time for the client, but also improves the response time for the other clients trying to reach the database.

4.5 Backup

The possibility of making backups of the system is a very important feature. To implement this feature an integrated database 'dumper' can be chosen, which will selectively export all or parts of the database. To do this, the system has to lock down the rest of the database so that no changes can be made until the data is dumped. As an alternative, the databases' native dumping utilities can be used if available. Using this solution however, could make a switch in database implementations harder, since certain databases ask for different dump commands.

4.6 Extensibility

The data in the database should be extensible with regards to the data that is to be modelled. It is not unlikely to expect fields to be added later or even new object classes. This should present no direct problems other than those which are inherent to making any changes to a system that has been running in production for any amount of time. The most important aspect of this is to make sure that data does not get corrupted or lose integrity. In this respect, any updates should be executed with care and the system should be taken off-line before any updates are made.

Chapter 5
Concurrency

Several users can use the system at the same time and in some cases these users are accessing the same data. To make sure no conflicts with harmful effects, such as data corruption and data inconsistency, arise in this process, we need to consider concurrency in our system design. This section provides an overview of concurrency problems.

5.1 Client / Server communication

Whenever clients need to communicate with the server, they do this via HTTP[1]. This method does not cause any concurrency problems by itself, but the tasks a requested method performs may inherently be troublesome in a system with multiple concurrent users (and thus requests). Every HTTP request spawns a new thread on the web server, so HTTP is concurrent by default.

An important fact to note is that all communication is initiated from the client to the server. A client does a HTTP request and the server responds. However, due to the nature of HTTP, multiple users can send concurrent HTTP requests accessing the same data. This enables the problem that two users might edit the same data at the same time. Which might result in incorrect data. Mark that at this point in time it is very unlikely that this occurs. But in the future, when more Reviews are added, it might be possible. However, it will not be corrupt, so there is no real danger of data corruption. All other problems are more of a problem to the user than to the system state. The 'solution' to all of these problems often consists of a warning being sent to a user, notifying them of some possible unexpected result. This way of handling concurrency is called *optimistic locking.* It generally means no actions are prohibited by the system in advance, so no objects are really being locked out of any action. Locking, here, merely means that no request will be accepted without checking if the state of the client that submitted the request matches the system state. If a conflict appears to exist, the user that has placed the request, will be notified.

5.2 DBMS/Hardware communication

The DBMS[2] is responsible for all physical data access. It provides an interface for our application to read and write to the database containing all data used in the system. As such, all direct disk I/O is abstracted. In effect, all concurrency problems concerned with disk I/O are taken care of by the DBMS. This does, however, mean that we need to configure the DBMS in the most optimal way for our system. While configuring the DBMS, we mainly optimize for performance, and let the DBMS take care of the concurrency issues concerning disk I/O. This means that we will choose the right table types and modes, but let the DBMS decide on threading its operations. This is, after all, one of the reasons to use an existing DBMS and not develop our own.

[1]HyperText Transfer Protocol, the protocol used by the Internet to display websites.
[2]DataBase Management System, an application that stores and acceses the data in a database.

5.3 Message flow

The server will never try to initiate contact with the client. Whenever the client needs anything from the server, be it (login) data or sending mutations, it is always the client that contacts the server. Using 'one-shot' transactions, inherent to HTTP, the client sends requests to the server which the server replies to with one 'answer'.

5.4 Call structure

The program is event-based. The client sits passively, waiting for the user to do something that it has to react to. Whenever the client has completed some requested action, it goes back to waiting. Periodically it might perform certain tasks (such as pulling mutations from the server), but apart from this, it will not do anything on its own initiative.

Chapter 6
Access control and security

6.1 Global resource handling

A local (on the server) MySQL database must be secured and managed locally. The ProMISe and SAP database, both accessable through webservices, are administered by third party actors. The application will not be responsible for any security errors at these resources.

6.2 Physical access to the hardware

Only the responsible administrator, assigned by SKION, will have access to the hardware on which the application is running. Other users will have access to their own hardware, whether or not with local administrator rights.

6.3 Software access to resources

Only the server on which the application runs has access to the local database, ProMISe database and SAP database. The latter two through webservices. The webservices need to be configured in ProMISe and SAP, by creating executable queries beforehand. These queries can be called by the webservices.

6.4 Authentication and security

Access control is handeled server side. This is done to prevent abuse of the system which will contain patient information. Because of this, it is relatively easy for the user to gain access to the application. There will be no need to install additional security software, next to a standard firewall and virusscanner. Most internet browsers, according to the minimum browser requirements for this application, are equipped with antiphising filters. If a user is trying to perform a request to the server for which it is not priviliged, the server should deal with this by sending propriate error messages.

Chapter 7
System architecture

7.1 Client / Server

The system will be build up using two distinct layers comprising a client / server architecture. SKION will run a web server which will contain the web application. The maintainance at the server will be performed by a classified administrator, which is capable of running additional server applications. The description of these applications is outside of the boundaries of this project. All users will communicate with the application being a client. Using their internet browsers, users can access the information for which they are granted.

7.2 Subsystem decomposition

As can be seen in chapter 2, the system uses the Model, View and Controller Design Pattern. A more detailed decomposition can be found in figure 7.1.

7.3 Interface design

In support of the development of the application Martijn Tijsma has developed a scheme which outlines how the interface should work according to the user of the application. This scheme is shown in figure 7.2.

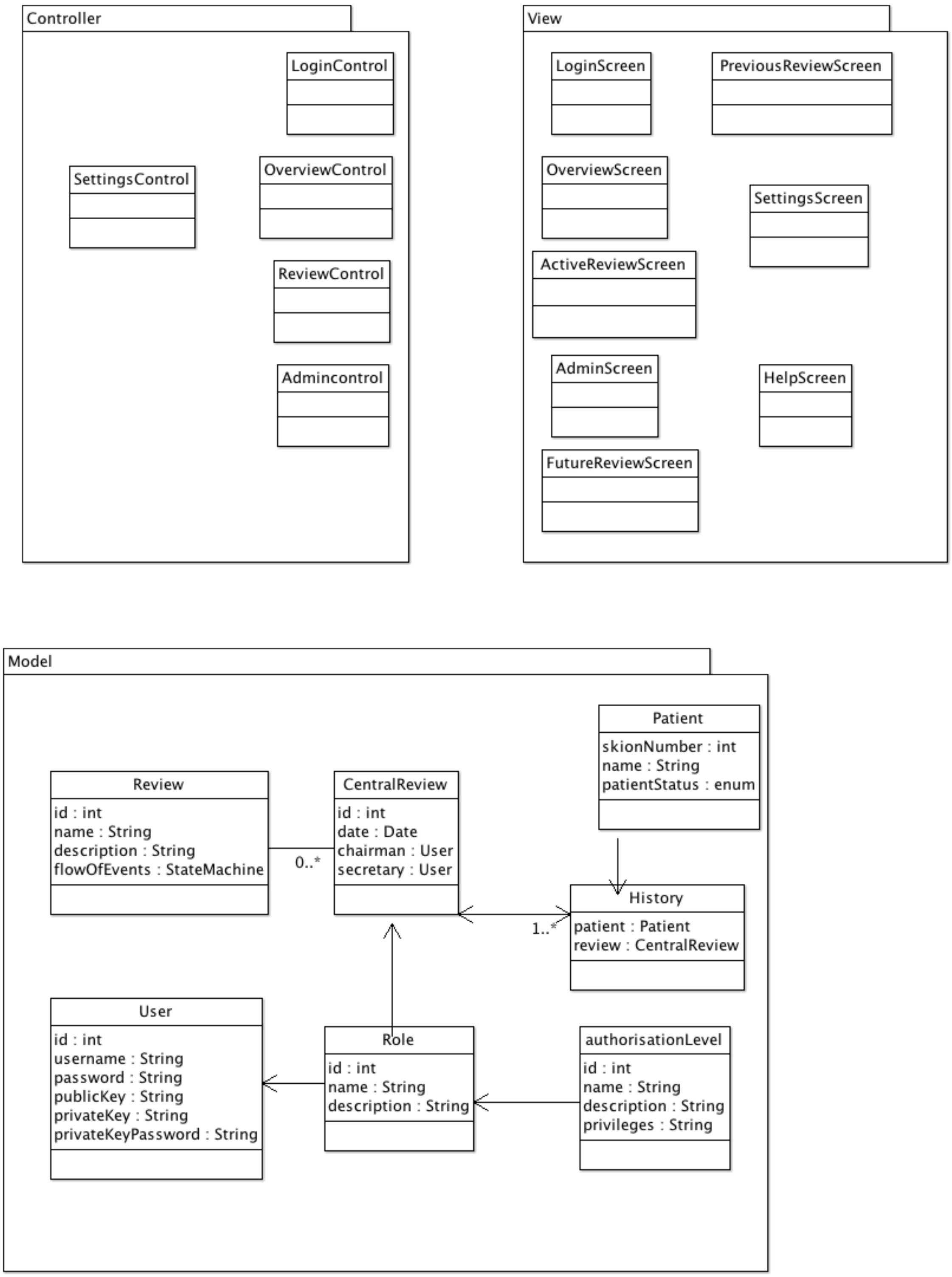


Figure 7.1: Subsystem decomposition in packages

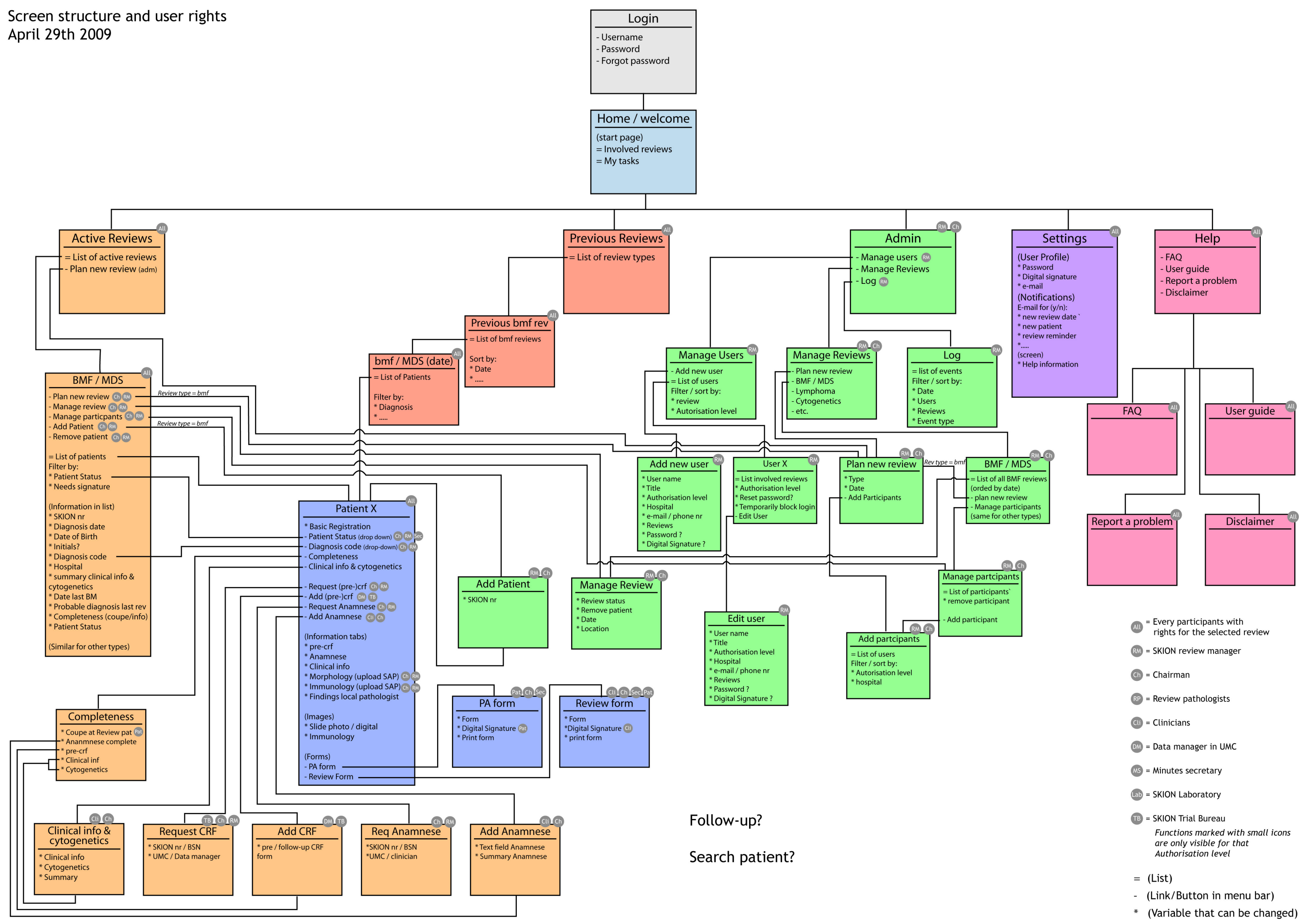


Figure 7.2: Scheme describing the interface design

Chapter 8
Boundary conditions

In this chapter, different situations in the system, that require some special care, will be described. These situations are divided in three catagories: the initialisation, the termination and failure of the system.

8.1 Initialisation

When the system is being initialised at the server, it should startup all of its subsystems and be able to read from the different databases and send mails. If no database can be connected to, it should retry at intervals of several minutes. The same goes for when not being able to send mails. In this case however, the system needs to retry to send the mail at intervals. It should also find out if some data is stored that should be stored to the database. Furthermore it needs to check whether there are still parties that should be notified for certain changes in the database. For these latter 2 problems it needs to start a cycle until all the changes made are executed and all notifications are sent. The system should still record all new changes made to the database and the new mails that are to be sent while performing this cycle.

8.2 Termination

For the client it doesn't matter when he or she decides to close the application. However since precious data might be accessed when the client didn't log out properly, the system needs to make sure that the client is logged out when closing the application. Of course this might become a nuisance when the client accidentaly closes down the application and has to log in again in order to complete his or her task. For this problem, we need to make sure that when data is filled in and the client closes the application, a call is made to the server to save the data currently filled in, so that when the client logs back in, the client's data is saved and ready for use. The server however is not supposed to ever be terminated.

8.3 System failure

For the client a system failure is always annoying, thus it makes sence to at least inform the client properly when such a system failure occures. However we need to make it as simple as possible for the client to handle the failure. Thus the system itself should handle failures when they occure. When a failure occurs, the system needs to logout the client and save the data into the database. If this can't be done because the server could not make a connection with the database, the server need to locally store this data. It's not possible for the server to lose connection with the local database, unless it is shut down. If for some reason the internet connection of the client may fail, the responsibility lies at the client. If data is filled in, it will be lost.

Appendix C
Implementation plan

Digital Review
Implementation plan

How and when to implement the Digital Review application

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS

May 8, 2009

Chapter 1
Implementation plan

This implementation plan will outline a plan on how and when to implement the designed application. Therefore a detailed planning is set up, and included in the planning excel file, and several milestones are defined.

1.1 Milestones

1.1.1 Database connectivity

The application is required to facilitate connection with a SAP database from the HaGa hospital and a MySQL database, ProMISe, from the LUMC[1] hospital. If the database connectivity is implemented, the application is able to read and write data from and to the databases. Also, connectivity with a local database is set up.

The relevant classes are: some DatabaseConnector class.

1.1.2 Authentication

The application is required to provide authentication of users. All users will have distinct usernames and passwords. If the authentication functionality is implemented, users can login into the system and change their passwords.

The relevant classes are: SettingsControl, User, Role and AuthorisationLevel.

1.1.3 Patient Overview

The application is required to present an overview of all the patient currently in the Review, patients scheduled for the next review and patient scheduled for previous and future reviews. If the patient overview is implemented, lists of patients can be seen in the User Interface.

The relevant classes are: Patient, History, Central Review, OverviewControl.

1.1.4 Detailed Patient View

The application is required to provide a authorized user with detailed patient information if requested. First is determined which information is needed in the detailed patient view. If the detailed patient view is implemented, a user is able to consult detailed patient information.

The relevant classes are: Patient, the View classes that display the Patient.

1.1.5 Review specific Functionality

The application is required to facilitate specific functionality for different types of reviews. If this functionality is implemented, all the functionality which is Review dependent is working.

Form handler

These are the handlers to create and process the various forms needed by the system.

The relevant classes are: FormScreen, FormElement and FormControl.

[1]Leids Universitair Medisch Centrum

StateMachine

These are the classes to model the flow of events of the Review as a statemachine. Also the flow of events of the BMF/MDS review is modelled.

The relevant classes are: Review, StateMachine, State, Transition.

BMF/MDS Forms

These are the forms used by the BMF/MDS review, they will be modelled in such a way that the Form handler can process them.

The relevant classes are: Review.

1.1.6 Digital Signature

The application is required to facilitate digital signature functionality, which will be used for signing offical diagnosis documents. If the digital signature functionality is implemented, users can sign and lock filled in forms.

The relevant classes are: User.

1.1.7 User interface

The application is required to have an user interface with which users use the system. If the user interface is implemented, the view part of the application is implemented.

The relevant classes are: all classes from the View.

1.1.8 Email

The application is required to send out emails for authentication, and notification of users. If the email functionality is implemented, users will recieve emails send by the application.

The relevant classes are: MailControl.

1.1.9 Logging

The application is required to facilitate sufficient logging functionality, which will be used by the administrator to trace back application access. If the logging functionality is implemented, all possible actions users can do are logged and saved into the database.

The relevant classes are: all Model classes.

1.1.10 Documentation

The project requires to deliver sufficient documentation about the application. This way, future development can be done more easily. If the documentation is complete, the Requirements Analysis Document, Architectural Design Document, Technical Design Document and Test Plan are updated so that the documents represent the current situation of the system.

Appendix D
Test plan

Digital Review

Test plan

How to test the Digital Review application

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
June 24, 2009

Chapter 1
Test plan

This plan will describe how the application and the different subsystems of the application will be tested. The following sections will describe the usage of Test Driven Design and different levels of testing.

Test Driven Design

For the implementation of the application, Test Driven Design (TDD) will be used. In TDD, one first designs a test case for the desired functionality. Clearly this test will fail, since there is no implementation yet. These tests will be designed such that an automated testing framework can run them all automatically. Now the programmer implements the functionality. The entire test suite will be run again by the testing framework and will either pass or fail. If it passes, the functionality is implemented correctly and the development can go on. If it fails, the implementation of the functionality is not correct and has to be reimplemented. These little tests for functionality are called *unit tests*. The proces of TDD is depicted in figure 1.

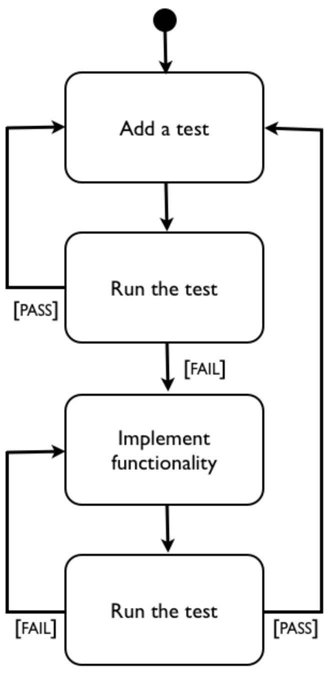


Figure 1.1: Flow of events in Test Driven Design

Databases

The application needs to communicate with three different databases (SAP, ProMISe and the local database containing for exampl the users). For correctness, the following functionality has to be thoroughly tested for each of the three different databases:

- connectivity to the database

- writing new content to the database
- reading from the database
- updating content already in the database
- removing content from the database.

Integration testing

Since unit tests alone are not a sufficient way of testing, we will also need to conduct integration tests. These integration tests check that all the different unit-tested components are properly integrated into one application. We can write some automatic test cases for this, but most will probably has to be tested manually.

Alpha testing

If the application is implemented and tested sufficiently, it will undergo an alpha test. An alpha test is a test by future users of the system in a by the programmers controlled environment. The alpha test is planned on the 8th of june. During the alpha test, employees of SKION will click through the application in order to find bugs and to give their opinions on the application as it is now. The bugs have to be reported to the project team, so that they can resolve them and improve the quality of the application. The opinions will be noted and included in the review of the application.

Appendix E
MoSCoW document

Digital Review
MoSCoW document

Must, Should, Could and Want to haves

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
May 8, 2009

Must haves

This section describes the key factors for the program to be implemented. The points listed below are expected to be the minimal factors with which the application will succeed. In other words; if the following points are not implemented, the application will not meet the minimum requirements.

- Participants need to be able to log in.
- Participants need to be able to retrieve their password if lost.
- Different authorisation levels are available for different users for different reviews.
- An overview screen for every user with personal information, actions and notifications.
- Review can be added, scheduled and rescheduled.
- An overview of patients scheduled for a review including completeness of information.
- Extra information can be added to the patients scheduled for a review.
- Filling in forms such as; pre-CRF, pathologist form, CRF and follow-up-CRF, including primary checks on data-entry.
- Storing forms into the ProMISe database.
- Validate forms using a digital signature.
- Reschedule a patient if no classifying diagnosis is met.
- Sending email notifications if required.
- Display patient information out of the SAP and ProMISe database.
- Providing sufficient logging according to NEN norms.
- Complete User Interface.
- Up-to-date system documentation.
- Adhere to the NEderlandse Normen for information security in the healthcare sector (NEN7510, NEN7511-2, NEN7512).
- Implementation of database.

Should haves

This section describes factors which will make the application more fancier. The implementation of these points will only be achieved if all of the must have factors are correctly implemented en tested sufficiently. The points listed below can be seen as features which will extend a working application.

- Printing the overview of patients (patient list) in the review.
- Functionality for adding coupe images or a link to the patient data.
- Retrieving location of coupe.
- Export information, results and other data to pdf/excel.
- Force the review of a patient, even if the information is incomplete.
- Printing forms.

Could haves

This section describes factors which could be implemented, but are not of essential need to the primary task of the program. The points listed below merely describe tasks which support the primary functionality of the program.

- On-screen support functionality.
- FAQ and user manual
- Substitute roles within a Review.
- Create a letter for doctors, containing classifying diagnosis.

Want to haves

This section describes functionality which will be nice to add to the application in the future. It is not expected to be implemented during this project.

- Full support for opening digital coupes from the interface.
- User profiles on which users can change their personal information and preferences.

Appendix F
Digital Signature research

Digital Review
Digital Signature Research

Suggestions for signing the Digital Review forms

Bachelorproject IN3405

Joël van den Berg
1322982

Joep Weijers
1308432

Bob Westerveld
1308459

Delft University of Technology
Faculty EEMCS
April 24, 2009

Chapter 1
Introduction and motive

This document will present the problem and some possible solutions for the digital signature of forms in the web-application of Digital Review. We will give some arguments, comparing the different solutions and we will then conclude which one will be the most suitable for the web-application.

An important feature of the web-application, is to be able to fill out a form preceding or during the review. Nowadays this is done by hand, and after the complete form is filled in by the chairman or minutes secretary, the specialist signs the form. It is relatively easy to create a digital form, but safely adding a signature from the responsible person is not trivial.

A signature (both digital as in the real-world) is used to ensure authenticity, integrity and non-repudiation. The first one ensures that the real responsible person signs it, not someone else. The second means that nothing is changed after signing. And the last one means that the person signing it, can not deny that he signed it. These three reasons for using a signature are equivalent for both the analog as the virtual world, and in the Netherlands since 8 may 2003, the digital signature has the same legal validity as a signature on paper[1].

However, there are some requirements to make a digital signature legally valid:

- The signature is uniquely bound to the signer.
- The signature makes it possible to identify the signer.
- The signature was made by means solely under control of the signer: It should be done with a computer.
- The signature must be in such a way attached to the document that every edit afterwards can be detected.

Summarising, we want a signature where different users in the web-application can sign a form adhering to the legal requirements. We will first look into the topic of encryption before presenting the solutions. Note however, that a *digital signature* does not necessarily mean a scanned image of a signature. It is something abstract and the concrete version can be an image, lines of text, et cetera.

[1]See Artikel 3:15a Burgerlijk Wetboek (Nederland)

Chapter 2
Encryption

This chapter will present some terminology and ideas about encryption to provide a better understanding of the problem and its solutions.

2.1 The environment of the signing

To start off, we need a clear idea of how the digital signature will be used in the application. A form will be signed when it is completed or at the end of a review by the responsible specialist. There are different specialists responsible for the different possible results from the Review. The responsible specialist will be logged in on a secured line and will be in a screen showing the form as it has been filled in by the one responsible for the forms (for example the chairman or a minutes secretary). Now the responsible specialist needs to perform an action to place his signature under it. The solutions will describe which action exactly the specialist needs to perform.

2.2 Hashing

One important cryptographic technique which will be used is hashing. A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message", and the hash value is sometimes called the message digest or simply digest.[1]

The ideal cryptographic hash function has four main properties:

- It is easy to compute the hash value for any given message.
- It is infeasible to find a message that has a given hash.
- It is infeasible to modify a message without changing its hash.
- It is infeasible to find two different messages with the same hash.

Thus a hash function takes an input string and turns it into a bit string called hashcode. It is very hard (as in: "almost certainly beyond the reach of any adversary who must be prevented from breaking the system") to compute the original input string, when given the hashcode.

Hashing is used for example to store passwords. A system should not store a password in plaintext, but as a hashcode. When the system should check if the password entered matches the stored password, all it has to do is take the hash value of the entered password and compare it to the stored hash.

There are several kinds of hashing functions, the most widely known are MD5 and SHA-1. Unfortunatly, both of these hash functions are not completely secure, they have their weaknesses. MD5 has completely been broken by people from the TU/e[2] and SHA-1 has been proven to

[1]Taken from: http://en.wikipedia.org/wiki/Cryptographic_hash_function

[2]Sotirov et al., MD5 considered harmful today: Creating a rogue CA certificate, http://www.win.tue.nl/hashclash/rogue-ca/ accessed March 29, 2009

have theoretical weaknesses[3]. This means, that with more research and faster computers SHA-1 can be broken in the near future. Therefore, the SHA-2 family of hash functions has been developed, to provide more security. Even though the SHA-2 family is withstanding attacks by cryptographers, it's only a matter of time before a flaw is detected and the computers are much more powerful. Therefore, an even better SHA-3 hash function is expected to be presented in 2011.

It is important to realise that to really decrypt hashes, thorough knowledge of cryptography and supercomputers are required. The cost of decrypting a hashcode is high with either hash function. But the possible damage done in the Digital Review system can be very high; if a malicious person breaks a hash and alters a filled in and signed review form, changing the diagnosis and resigns it, the results may be catastrophic.

Therefore we suggest to use SHA-2. It will take little more work to implement than MD5 or SHA-1, but the cost to break it is much and much higher.

2.3 Digital signature algorithms

There are different algorithms to set a digital signature. All of them are based on the same principle. A digital signature scheme typically consists of three algorithms:

- A key generation algorithm that selects a private key uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding public key.
- A signing algorithm which, given a message and a private key, produces a signature.
- A signature verifying algorithm which given a message, public key and a signature, either accepts or rejects.

Two main properties are required. First, a signature generated from a fixed message and fixed private key should verify on that message and the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party who does not possess the private key.[4]

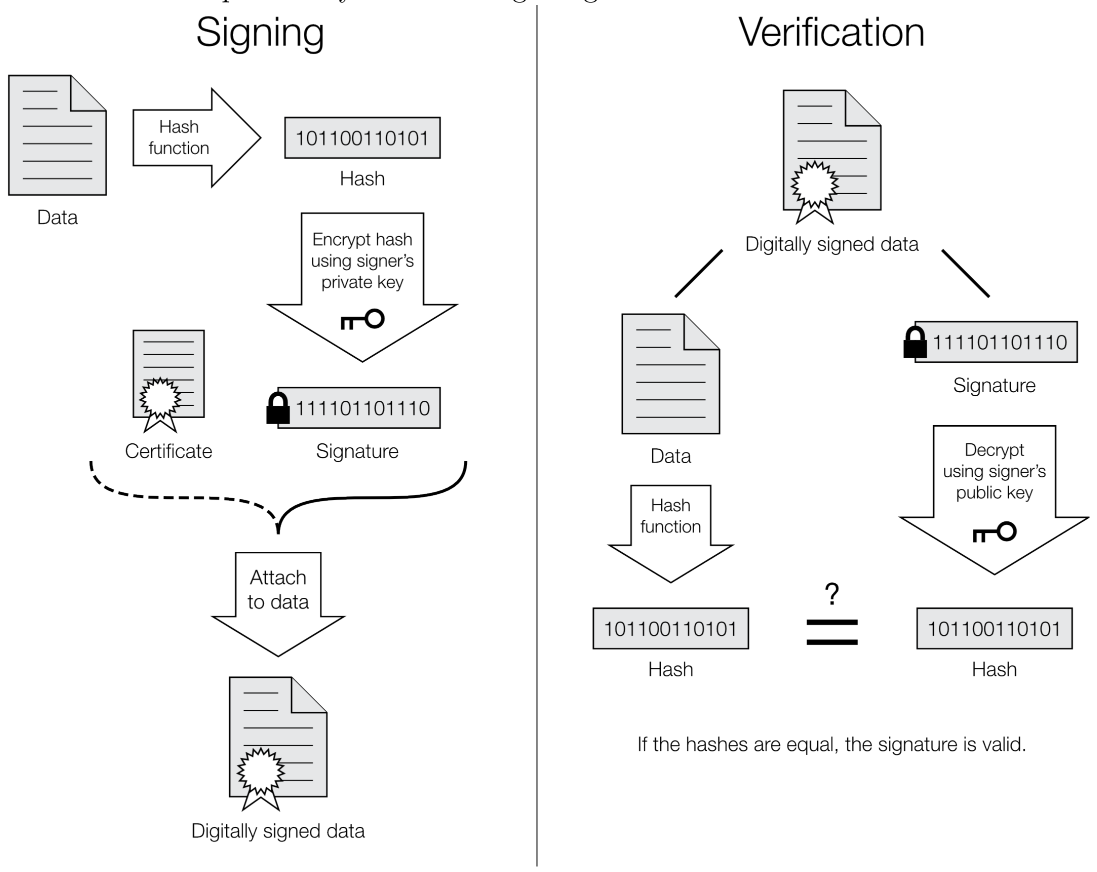
The most used, and best documented algorithm is the RSA algorithm, named after its inventors Rivest, Shamir, Adleman. RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key.

RSA can be used to sign a message in the following way: Suppose Alice wishes to send a signed message to Bob. She can use her own private key to do so. She produces a hash value of the message, encrypts it with her private key, and attaches it as a signature to the message. When Bob receives the signed message, he uses the same hash algorithm in conjunction with Alice's public key. He decrypts the signature with Alice's public key, and compares the resulting hash value with the message's actual hash value. If the two agree, he knows that the author of the message was in possession of Alice's private key, and that the message has not been tampered with since.

[3]Wang et al., Finding Collisions in the Full SHA-1, http://people.csail.mit.edu/yiqun/SHA1AttackProceedingVersion.pdf

[4]Taken from: http://en.wikipedia.org/wiki/Digital_signature

It's use is best explained by the following image:



An important aspect of this algorithm is the certificate. Certificates form the link between the public key and the signer. Real Security Certificates are issued by trusted organisations and are expensive ($399 per certificate per year, and each signing user needs a certificate)[5]. Since we will only use a "certificate" inside our program, the program itself can form the link between a public key and a person, eliminating the need for real security certificates.
We will now move on to the solutions, which may or may not use Hashing and/or RSA.

[5]Trusted Site packages at Verisign, http://www.verisign.com

Chapter 3
Overview of solutions

We will start off with the question if we should use an analog, real-world signature or a digital signature. The analog signature means that a filled-in form must be printed, signed by the responsible doctor and then photocopied for distribution. The advantage for us is that we don't need to implement any digital signature, the system should just leave some space on the printout to sign. The downside is that the system has no idea whether the form is signed or not.

We can pull the digital signature to multiple directions. One is to use a drawing tablet to draw a signature in a form field. The advantage is that the system now knows whether the form is signed or not. However it brings a lot of downsides. This method requires a drawing tablet (Costing around €75,- each)[1] and an interface to be able to actually draw the signature. But the major disadvantage is that this signature does not guarantee that information in the form hasn't been edited since signing.

Signing a form with RSA ensures integrity of the form since the moment of signing, because every change to the form data will result in a different hashcode. The question with RSA is how to store the public and private keys. The public key can be stored in the database containing all the users, since they are only used inside our program.

The private key consists of two large numbers and is very hard to remember, therefore we need some way to store it and make it accessable only to the person it belongs to. Industry standard systems make use of a Smartcard, a small card containing Security Certificates with the private key. Inserting the smart card in a reader enables decryption with a private key. The dutch medical industry is using Smartcards, called UZI-cards[2]. This will result in all dutch healthcare providers having a Smartcard. The organisation UZI-register[3] also provides the cardreaders and signs security certificates.

Alternatively, we could store the private key in the system and allowing access only to the user that is associated with that private key. This is possible in various ways:

- No additional security, the user is logged into the system, so we know who it is. A downside is that if the responsible specialist logs into the system and walks away from his computer, then someone else can sign the form.
- Retyping the password, prevents the downside of the previous item, but if someone got hold of the account details of the responsible specialist, he knows the password and can still maliciously sign forms.
- A second password or a PINcode, used only for the signature. This password will not be used a lot, so it will be harder to get hold of. However, it will be harder to remember by the responsible specialist.
- A scheme using SMS text messages, like the TANcodes used by ING Bank for online banking. This requires an SMS server.
- A fingerprint reader. This solution also needs an interface to the application, and a fingerprint reader, costing around €35,-.

[1]Tweakers pricewatch, http://www.tweakers.net/pricewatch consulted 16-04-2009
[2]Unieke Zorgverlener Identificatie-pas, Unique Healthcare Provider Identification card
[3]Unique Healthcare Provider Identification Register, http://www.UZIregister.nl

Chapter 4
Conclusion

In the previous chapter we presented different solutions to the problem of signing a form. Our recommendation is to store the private key in our system and make it accessible by a second password or PINcode. This will provide a sufficient level of security, is one of the easiest ways to implement and is one of the cheapest solutions.

However we do like to mention to keep the UZI-cards in mind. This will be harder to implement, and requires a lot of extra work and arrangements, but will improve the ease of use of logging into the system. Unfortunatly, due to our tight timeconstraints, we probably will not have time to implement it, so we will stick with the additional password of PINcode.

Appendix G
Project Planning

Datum	Activiteit	Tijd	Locatie
Week 1 6 april - 10 april			
Ma	SKION STARTUP TODO lijst SVN Eclipse		SKION
Di	Webserver regelen Hokje regelen Zaal presentatie regelen		Delft
Wo	RAD ADD TDD voorbereiden PvA schrijven		Delft
Do	RAD ADD TDD voorbereiden PvA schrijven		Delft
Vr	Requirements verzamelen PvA af RAD: Current system & desired system		SKION
Week 2 13 april - 18 april			
Ma			SKION
Di	Requirements verwerken RAD		Delft
Wo	Digitale handtekening uitzoeken RAD		Delft
Do	Digitale handtekening uitzoeken RAD		Delft
Vr	Afspraak SAP database + demonstratie Promise database		SKION
Week 3 20 april - 24 april			
Ma	Afspraak LUMC	14:30	SKION
Di			Delft
Wo			Delft
Do	Afspraak Emile	13:30	Delft HB11.030
Vr	RAD ADD TDD MoSCoW IP TP		SKION
	Afspraak SAP man HAGA		SKION
Week 4 27 april - 1 mei			
Ma			SKION
Di			Delft
Wo			Delft
Do			Delft
Vr			SKION
Week 5 4 mei - 8 mei			
Ma	Voordracht systeem zoals ontworpen		SKION
Di			Delft
Wo			Delft
Do			Delft
Vr	ADD TDD MoSCoW IP TP		SKION
Week 6 11 mei - 15 mei			
Ma	Begin implementatie M1, M2, M5		SKION
Di	M1, M2, M5		Delft
Wo	M1, M2, M5 Afspraak Emile Hendriks	15.00	Delft
Do	M1, M2, M5		SKION
Vr	M1, M2, M5		SKION
Week 7 18 mei - 22 mei			
Ma	Presentatie 'mid term evaluation' M3, M4, M1	12:00	Delft IO
Di	M1, M3, M4		Delft
Wo	M1, M3, M4		Delft
Do			Delft
Vr			SKION
Week 8 25 mei - 29 mei			
Ma	M7 + uitloop M1 t/m M5		SKION
Di	M7 + uitloop M1 t/m M5		Delft
Wo	M7 + uitloop M1 t/m M5		Delft
Do	M7		Delft
Vr	M7		SKION
Week 9 1 juni - 5 juni			
Ma			SKION
Di	M6		Delft
Wo	M8, M9, M6, M7		Delft
Do	M8, M9, M7		Delft
Vr	M8, M9, M7		SKION
Week 10 8 juni - 12 juni			
Ma	~~Alpha test~~		SKION
Di	M10, bug fixing		Delft
Wo	M10, bug fixing		Delft
Do	M10, bug fixing Afspraak Emile	 SKION 16:00	SKION
Vr	Implementatie af M10		SKION
Week 11 15 juni - 19 juni			
Ma	Bob toelating		SKION
Di			Delft
Wo	Ethiek		Delft
Do	Alpha test		Delft
Vr	Software Testing		SKION
Week 12 22 juni - 26 juni			
Ma	Complexiteits		SKION
Di			Delft
Wo	Tentamen Joep Principe afspraak Emile	 14.00	Delft
Do	Presentatie SKION	13.30 TU	Delft
Vr	Verslag af		SKION
Week 13 29 juni - 3 juli			
Ma	Verslag sturen naar leden commissie		SKION
Di			Delft
Wo			Delft
Do	Presentatie 10.00 Shannon zaal Evaluatie bij en met SKION		Delft
Vr			SKION

Legenda
🟥 Geen werkdag
🟩 Afspraak
🟪 Milestone
🟦 Datum

Appendix H
Alphatest

Alpha test Digital Review Application

Deze user test is bedoeld om fouten te ontdekken in de functionaliteit van de digital review applicatie.
De test mag enkel uitgevoerd worden in een gecontroleerde omgeving en alleen met test of acceptatie data van patiënten.

Gebruiker:	……………………..
Datum:	……………………..

1. Ga naar 192.168.120.33/View
   Log in
   a. Gebruik de volgende gegevens
      Gebruikersnaam: ***tudelft***
      Wachtwoord: ***kamerplant***

2. Voeg een patiënt toe aan het systeem
   a. Gebruik dit SKION nummer ……..….
   b. Noteer hieronder de initialen van de toegevoegde patiënt.

      Initialen: …………..

3. Voeg de bij 2 gecreëerde patiënt toe aan de Beenmergfalen review van 20 juni 2009.

4. Verander de instellingen van de BMF MDS review van 2 oktober 2010 naar:
   a. 3 oktober 2010 om 12.00 uur

   b. Chairman = …………………..

   c. Secretary = …………………..

5. Bekijk nu de acties (in de Log) die zojuist zijn uitgevoerd.

6. Verander de namen van een gebruiker van het systeem
   a. Verander Joep Weijers in Joepie Weij
   b. Bekijk of de veranderingen hebben plaatsgevonden
   c. Verander de gegevens terug.

7. Voeg een nieuwe gebruiker toe, met uw eigen gegevens.
   Noteer hier uw eigen gegevens:

   Gebruikersnaam: …………………………………..

   Wachtwoord: …………………………………..

   Wachtwoord Digitale handtekening: …………………………………..

8. Bij de BMF MDS review van 20 juni 2009; voeg de door u zojuist gecreëerde gebruiker toe als ‘clinician’.

9. Log uit en log opnieuw in met uw nieuwe gegevens (zie punt 7)

10. Ga naar BMF MDS review van 20 juni 2009.
    a. Kies patiënt zoals toegevoegd bij punt 2 en vul een aantal vakjes van het pre-crf formulier in.
    b. Onderteken het formulier.

11. Probeer het zojuist ingevulde pre-crf formulier te wijzigen.

12. Log uit en log wederom in met:
    a. Gebruikersnaam: ***tudelft***
       Wachtwoord: ***kamerplant***

13. Verplaats de patiënt toegevoegd bij punt 2 van de BMF MDS review van 20 juni 2009 naar de review van 11 December 2011.

14. Controleer of de patiënt werkelijk verzet is.

15. Zet de patiënt weer terug naar de review van 20 juni 2009 .

16. Verwijder deze patiënt uit de review

17. Log uit

Bedankt voor het testen van onze applicatie!

Opmerkingen:

Appendix I
Alpha test results

At the end of the project, three alpha tests were done by three of the employees at SKION. The employees were lead through an action pattern that should be basic when using the application. The results of the three alpha tests are gathered below.

- Layout
    1. The 'add patient' link should be on the left.
    2. The time in the Edit and CentralReview screen should be between brackets.
    3. Make more use of headers to improve usability.
    4. Improve the position of the buttons.
    5. The forms make use of too many radio buttons.
    6. There should be an overall improvement on feedback on actions performed by the user.
- Usability
    1. Improve the speed of the gathering of data about patients.
    2. What will happen when different patients have the same initials?
    3. What will happen when 1000 of patients have to be retrieved from the ProMISe database.
- Bugs[1]
    1. Assertion error in User.php.
    2. Layout issue in patient.php.

[1]Have already been fixed

Bibliography

[1] *Object-Oriented Software Engineering.* McGraw-Hill, 2005.

[2] *Software Testing and Analysis: Process, Principles and Techniques.* Wiley, 2007.

[3] Martijn Tijsma. Digital review, March 2009.