

Heilbronn University and Heidelberg University in  
cooperation with Uganda Martyrs University

# Introducing Dynamic Forms in a Client-Server Based Hospital Information System in Uganda

Medical Informatics - Master Thesis

Britta Lohmann

29.02.2016

Supervisor: Prof. Thomas Wetter, Heidelberg University

Co-Supervisor: Prof. Tomas Benz, Heilbronn University

Examiner: Dr. Simon Ndira

## Abstract

eHMIS is a Ugandan Hospital Information System (HIS), which targets the Sub-Saharan market. In its first version all forms were programmed statically and adaptations were done by code modifications. In 2014 the development of a second version of eHMIS based on Java started.

This work aims at introducing dynamic forms to this new version. While forms that are significantly important to the workflow of the application will remain static, others are replaced by forms that are dynamically designed by the user. By that, the application will become more flexible and local and situational tailoring will be possible without inducing extra costs.

In this thesis the design, implementation and testing of dynamic forms in eHMIS is discussed. The architecture is based on the questionnaire resource of FHIR®. The module enables the user to create questions and group them into sections and questionnaires. For each question the type of answer expected and other constraints can be defined. A user interface covering all functions was designed, so that no programming skills are required. In a first step dynamic forms were integrated in the application's workflow for recording symptoms, though other fields of application are possible. For testing, a usability experiment was conducted in Tororo Hospital in Eastern Uganda, using the thinking aloud method. Results were analysed and evaluated to detect usability problems and gain a general impression of user satisfaction.

## **Declaration of Authenticity**

I have written this thesis independently, solely based on the literature and tools mentioned in the chapters and the appendix. This document – in the present or a similar form – has not and will not be submitted to any other institution apart from the University of Heidelberg and Heilbronn University to receive an academic grade.

## **Acknowledgements**

I would like to gratefully acknowledge Prof. Thomas Wetter for all guidance and for providing me an opportunity to go to Africa. Thank you for giving me that chance and for continuously supporting me in numerous Skype discussions and mails.

Furthermore, I wish to express my gratitude to Prof. Tomas Benz, his wife Damalie and her family in Kampala. Thank you for all helpful advices and the great accommodation. Because of you, I had a shelter and a family caring for me the moment I came to Uganda.

Thank you Simon Ndira for inviting me to Uganda. It was a pleasure to work with you and your team. Special thanks to Ben Oduli and Gerald Matege. You helped me finding my way through Kampala and the million lines of code.

Finally I want to thank my parents for always supporting me in my decisions and for encouraging me at all times.



# Contents

<b>1. Introduction</b>	<b>9</b>
1.1. Motivation . . . . .	9
1.2. Objective and Research Question . . . . .	10
<b>2. Background</b>	<b>11</b>
2.1. State of the Art . . . . .	11
2.1.1. Generic Databases . . . . .	11
2.1.2. Programmed Forms . . . . .	12
2.1.3. Dynamic User Interface . . . . .	12
2.2. eHMIS . . . . .	13
2.3. FHIR . . . . .	15
2.4. Thinking Aloud . . . . .	19
<b>3. Results</b>	<b>22</b>
3.1. Requirement Analysis . . . . .	22
3.2. Modelling . . . . .	23
3.2.1. FHIR Resources . . . . .	23
3.2.1.1. Modifications . . . . .	25
3.2.2. States . . . . .	26
3.2.3. Answer Format . . . . .	27
3.2.4. Constraints . . . . .	28
3.2.5. Integration into the Application Workflow . . . . .	29
3.3. Implementation . . . . .	30
3.3.1. Data Layer . . . . .	32
3.3.2. Logic Layer . . . . .	34
3.3.3. Presentation Layer . . . . .	35
3.3.3.1. Validation . . . . .	36
3.4. User Interface Design . . . . .	37
3.4.1. Questionnaire Creation . . . . .	38
3.4.2. Questionnaire Usage . . . . .	42
3.5. Experiment . . . . .	45
3.5.1. Preparation . . . . .	45
3.5.2. Implementation . . . . .	46
3.5.3. Evaluation . . . . .	47
<b>4. Discussion</b>	<b>54</b>

4.1. Discussion of Methods . . . . .	54
4.2. Discussion of Results . . . . .	55
<b>5. Outlook</b>	<b>57</b>
<b>6. Bibliography</b>	<b>58</b>
<b>A. Appendix</b>	<b>60</b>
A.1. Use Case Descriptions . . . . .	60
A.2. Logic Layer Code Example . . . . .	72
A.3. Presentation Layer View Example . . . . .	75
A.4. User Interface Screenshots . . . . .	77
A.5. Documents of the Experiment . . . . .	78
A.6. Analysis of the Experiment . . . . .	83

# List of Figures

2.1. eHMIS feature overview [29]	15
2.2. FHIR resource categories	16
2.3. FHIR Patient resource defined in UML[11]	17
2.4. FHIR Patient resource sample data defined in XML[10]	18
3.1. Use case diagram of the creation of a dynamic questionnaire	23
3.2. Use case diagram of the application of a dynamic questionnaire	23
3.3. FHIR Questionnaire resource [30]	24
3.4. FHIR QuestionnaireAnswer resource [31]	25
3.5. Questionnaire data structure	26
3.6. eHMIS application structure	31
3.7. Questionnaire entity classes	32
3.8. Layout structure	38
3.9. Screenshot of the groups' section	39
3.10. Screenshots of the question creation form	40
3.11. Screenshot of preview function	41
3.12. Screenshot of feedback list	42
3.13. Screenshot of an expanded feedback	43
3.14. Screenshots of a dynamic questionnaire	44
3.15. Computer experience of subjects	48
3.16. Duration of the experiment compared to years in profession	48
3.17. Time taken for the first compared to the fifth question	49
3.18. Answer types used	50
3.19. General understanding of answer types	51

# List of Tables

2.1. Static tables that contain structural information . . . . .	11
3.1. Questionnaire constraints . . . . .	29
3.2. Usability problems . . . . .	51

# 1. Introduction

## 1.1. Motivation

Electronic forms have been used for a long time and proved to be an effective interface design, especially for complex data capture.[1, 2, 4] As most people know paper-based forms, it is easy for them to become acquainted with the electronic counterpart. Another advantage is the possibility of structuring tasks with which the user can comply. Besides forms can be implemented in an economic way.[3] However, electronic forms come with their own challenges: On the one hand they can be overloaded. Too many fields complicate navigation and distract the user although most of the fields might not even be applicable in the given situation.[4] On the other hand, forms suffer from missing flexibility, which increase the risk of later code changes and extra costs.[5]

Among others these reasons motivated the research on different kinds of dynamic forms. Meaning of the term varies and systems differ in usability, database structure and their ability of processing. If forms are created dynamically, the application is incapable of computing and processing that data by itself. Instead the user has to define his<sup>1</sup> own dependencies and computation rules. That is why complex processing always comes along with a more complicated handling of dynamic forms.

The application eHMIS faced similar problems. eHMIS is an electronic health record system, which was developed by Simon Ndira within the scope of his dissertation.[6] It is applied in different hospitals in rural and urban Uganda. Data capture was done using forms, which were implemented statically in the first version. In use, the development team received many change requests, which had to be satisfied. Every modification, even a small one like adding an extra field to a form or removing an unnecessary field had to be implemented by a programmer. Since the need of changes can have several reasons, e.g. new regulations by the government, internal restructuring, personal liking etc. this was seen as an ongoing issue.

Currently a new version of the application is being developed, which will face identical challenges. A hospital information system like eHMIS is based on sophisticated processing regarding the management of patients and their diagnoses, medications etc. Thereby a completely dynamic design becomes impractical. Instead a mixed approach is thought of. While forms that are crucial for the workflow of the application remain static, others can be replaced by dynamic forms. Those forms can be created and modified by users and change requests that have to be handled by the development team decrease.

---

<sup>1</sup>In favor of readability the masculine gender will be used throughout this text.

## 1.2. Objective and Research Question

Can dynamic forms be integrated into a hospital information system in a developing country in sub-saharan Africa, so that adaptations do not consume additional resources? This is the research question which this work will focus on. Before an answer can be given the following objectives have to be achieved:

- Implementation of dynamic forms
- Integration into the clinical workflow of eHMIS
- Design of a user interface, that does not require programming skills
- Application in an Ugandan hospital

In a work environment where these goals have been achieved, the application can be tailored locally and situationally without any additional programming effort.

## 2. Background

### 2.1. State of the Art

Alongside with other techniques, electronic forms are an established design for data entry in human computer interaction, which enable the user to work effectively on complex data.[1, 2] Forms imply an expedient structure and are familiar to almost all users. Besides they don't require advanced hardware or software.[3] Yet, form-based interfaces suffer from missing flexibility when requirements or situations change.[5] Fields may become unnecessary and a distraction to the user. Restructuring at this point is time consuming and costly.[4]

To overcome these challenges, concepts of dynamic forms or dynamic questionnaires were developed. These terms are not clearly defined and used in a wide range. The following section presents different technical approaches to give the reader an overview of the topic.

#### 2.1.1. Generic Databases

During the implementation of a system that uses dynamic forms for data entry the data that will actually be entered into the system cannot be predicted. Dynamic forms automatically lead to dynamic data which then require a dynamic database to be composed. In China Shao-Zhong & Tao (2010) realised a browser/server-based application that combines the static with a dynamic database. A conventional basic database contains all user information including credentials. For the dynamic database a generic approach was chosen. Two static tables (see table 2.1) are forming the foundation.

With these basic structures, tables and items of the database are modelled and then created by the system. The user does not face any structural restrictions. A preprogrammed interface enables the user to manage the database and to modify the data. No

Field	Type and length	Field	Type and length	Index
Table_Id	int(4)	Item_Id	int(4)	pk
Table_Name	varchar(50)	Table_Id	int(4)	pk
Table_Use	varchar(2)	Item_Use	varchar(2)	
Table_Description	varchar(50)	Item_Type	varchar(10)	
		Item_Name	varchar(10)	
		(Item Description field)		

Table 2.1.: Static tables that contain structural information

programming skills are required.

The main obstacle of generic databases is business logic. Since the type of data stored is unknown to the system, hardly any computations are possible. All data is presented in tables and allows only basic data management. In addition, database queries cannot be optimized and complex requests or cascading operations cannot be executed efficiently. [5]

### 2.1.2. Programmed Forms

As early as in 1995 DYNAMIC FORMS, an extensible, programmatic framework, was published. Forms are created by using an object-oriented text-based form description language (FDL). The language doesn't need to be compiled and changes are adapted immediately. Editing can be done in a text editor. Alternatively, DYNAMIC FORMS provides a user interface to create and modify forms, although not all functionality of the FDL is supported. A form consists of different fields which are defined by properties. More complex items, like buttons for single or multiple choices can be integrated. Moreover dependencies can be determined that enable auto-computing. The system supports ordering by drag'n drop, an automatic layout mechanism, collapsable sections as well as guidance tools, like checklists for the user. All captured data is stored as String values. DYNAMIC FORMS is well suited for rapid prototyping because it allows easy, real-time adjustment of forms. The ability to define complex dependencies, computations and constraints (e.g. for visibility of fields) makes it a powerful tool for form design. Unfortunately, the user interface only allows access to parts of this functionality. For advanced tasks a programmer is needed, who is familiar with the FDL and the programming language C.[4]

DYNQUEST is a modern example of programmed forms. It is an open source application designed for researchers to create (online) surveys. The basic idea is to link HTML pages dynamically so that the content of the following page depends on answers given on the current page. No user interface is provided. The user creates the different HTML pages which contain the fields for data capture himself. Subsequently, the user defines conditional links between pages inside the HTML tags. Finally the model algorithm of DYNQUEST assembles all files to a dynamic questionnaire. Similiar to DYNAMIC FORMS, data is stored as String values in a text file. As stated before, basic programming skills as well as knowledge of the operating system are required for the correct configuration of the application. At this point the authors suggest the conjunction with other software (e.g. an HTML editor) to simplify use.[7]

### 2.1.3. Dynamic User Interface

Chen et al. (2009) take a completely different approach with focus on the design of forms. During their research in East Africa they found that too little attention was given to the design of forms which leads to decreased data quality. Most of the existing standards and techniques to guarantee a suitable design and ensure data quality are expensive and by



that not applicable in a resource-poor environment. Driven by this impression, a system called USHER was developed, which dynamically optimizes the user interface. Based on statistical data modelling, appropriate forms and constraints are created. The layout as well as the order of fields are optimized automatically. Moreover, USHER provides user feedback in terms of hints, warnings, auto-completion etc. Comments of other users on specific fields can also be displayed. All functionalities aim at increasing data quality at entry time. Before the system can start operating, training data is needed to train the statistical model as core of the application. In contrast to the projects presented before, USHER doesn't involve a dynamic database. Instead, all data that can be captured is predefined.[8]

## 2.2. eHMIS

The Health Management Information System (HMIS) was published by the Ugandan Ministry of Health. HMIS is a monitoring and reporting system which provides various information about all healthcare levels and their performance. Its main objective is to support the ministry as well as other stakeholders in analysing and improving management of the Health Sector in Uganda.[25] Based on HMIS, which is exclusively paper-based, Simon Ndira decided to develop eHMIS, an "electronical Healthcare Management Information System".[26]

The first version of the application is based on a LAMP<sup>1</sup> or rather WAMP<sup>2</sup> architecture and offers different functionalities: The centrepiece is the electronic health record allowing the documentation of all information regarding the patient, his illnesses and treatments. Beside that, the system provides an ancillary module and basic support infrastructure, e.g. reporting and data backup. eHMIS was deployed in Tororo Hospital and Mifumi Healthcenter in Eastern Uganda as well as in Mengo Hospital in Kampala, which cooperated during the development process. While in 2015 it is still running in Mifumi and certain departments of Mengo (e.g. the eye clinic), Tororo temporarily abandoned the usage of an EHR system because of unstable power supply.[27] However, in the past years, weaknesses of the approach became visible: On one side, the operation in business revealed the lack of a complete accounting module. On the other side, the development in PHP turned out not to be very efficient because a proper IDE providing tools for debugging, refactoring etc. was missing.

In 2014 a new implementation of eHMIS based on Java began. The main objective of the new version is the creation of a modular, extensible and configurable application that combines maintainability and interoperability by dint of the usage of standards and established technologies. eHMIS is built as a web application with a three-tier architecture. Thereby, the software doesn't need to be installed on client-side and can be called via browser. Besides, this approach enables future cloud solutions. The three tiers are the data access layer, the logic business layer and the presentation UI layer. Inside these

---

<sup>1</sup>LAMP: Linux, Apache, MySQL, PHP

<sup>2</sup>WAMP: Windows, Apache, MySQL, PHP

layers different technologies are applied. Data access is realised as object-relational mapping with Hibernate. The UI layer consists of various frameworks, standards, languages and libraries. The interface is designed with HTML5, CSS3 and JavaScript, extended by jQuery. Additionally, Thymeleaf is used as a template engine. Behind the scenes the MVC<sup>3</sup> structure of the Spring framework is applied. Finally, there is ModelMapper, which converts data objects coming from the data access layer to view objects. It is required that all integrated technologies be well established and open source. The idea behind is to simplify familiarisation for new developers and the integration of other open source software like DHIS2, which is already scheduled. To maximise interoperability with different EHR systems, the data is structured internally according to FHIR® (see section §2.3). FHIR itself then encourages the adoption of medical standards.[28]

In the workflow of eHMIS, every patient and staff member is always assigned to a location. A location might be the reception, different clinics or the pharmacy. For staff, the different locations are also associated with different access rights. Typically, a patient visits the reception first and gets registered. In the system, an encounter is created for him. An encounter is equivalent to a case, a visit or multiple visits, if they are associated. Afterwards, he might be sent to a specific doctor. In eHMIS his location is changed and he gets cued at the new location. The doctor can do the examination, order for tests etc. The resulting diagnosis is recorded as a condition, which is linked to the patient and the current encounter. Then, prescriptions, billing etc. are processed in a uniform manner.

eHMIS offers a wide range of functionality; the different features are shown in 2.1, although not all of them are realised yet. The first release is planned for the year 2016 in Mifumi Healthcenter. Not all functionality will be enrolled in this release, though a second and third one is intended. The software will be published as freemium model. While EHR components are provided for free, money-related modules, like accounting, will come with costs.

---

<sup>3</sup>MVC: Model, View, Control

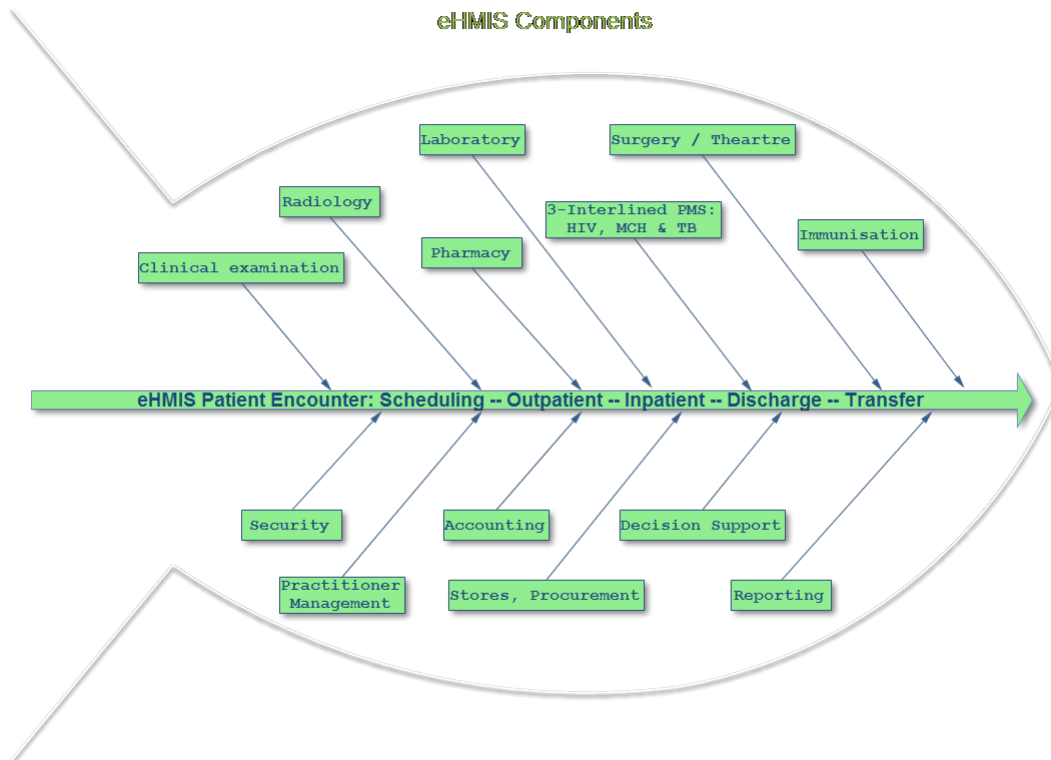


Figure 2.1.: eHMIS feature overview [29]

## 2.3. FHIR

The “Fast Healthcare Interoperability Resources” (FHIR®) is a standard originated from HEALTH LEVEL SEVEN INTERNATIONAL (HL7). It is an open source project and has since Oct. 24th been released as DSTU 2<sup>4</sup>. [9] FHIR provides a framework for the realisation of computer based information systems in healthcare and hence has a strong focus on implementation. The idea is to allow institution-spanning cooperation by defining a common data structure and an API to exchange information between different healthcare facilities. Besides, this approach also covers the integration of mobile devices. FHIR can be used in different settings; in addition to clinical care, public health and clinical trials, administration and finances are considered as well. The use of FHIR in veterinary care is possible, too. FHIR strives to increase the interoperability, extensibility, flexibility and scalability of information systems in healthcare. To achieve these goals, other established standards are applied. Among others, the framework makes use of XML and JSON objects and includes the definition of a RESTful API. Alongside, FHIR encourages the use of different healthcare standards and nomenclatures like HL7 or SNOMED. [10]

<sup>4</sup>Draft Standard for Trial Use

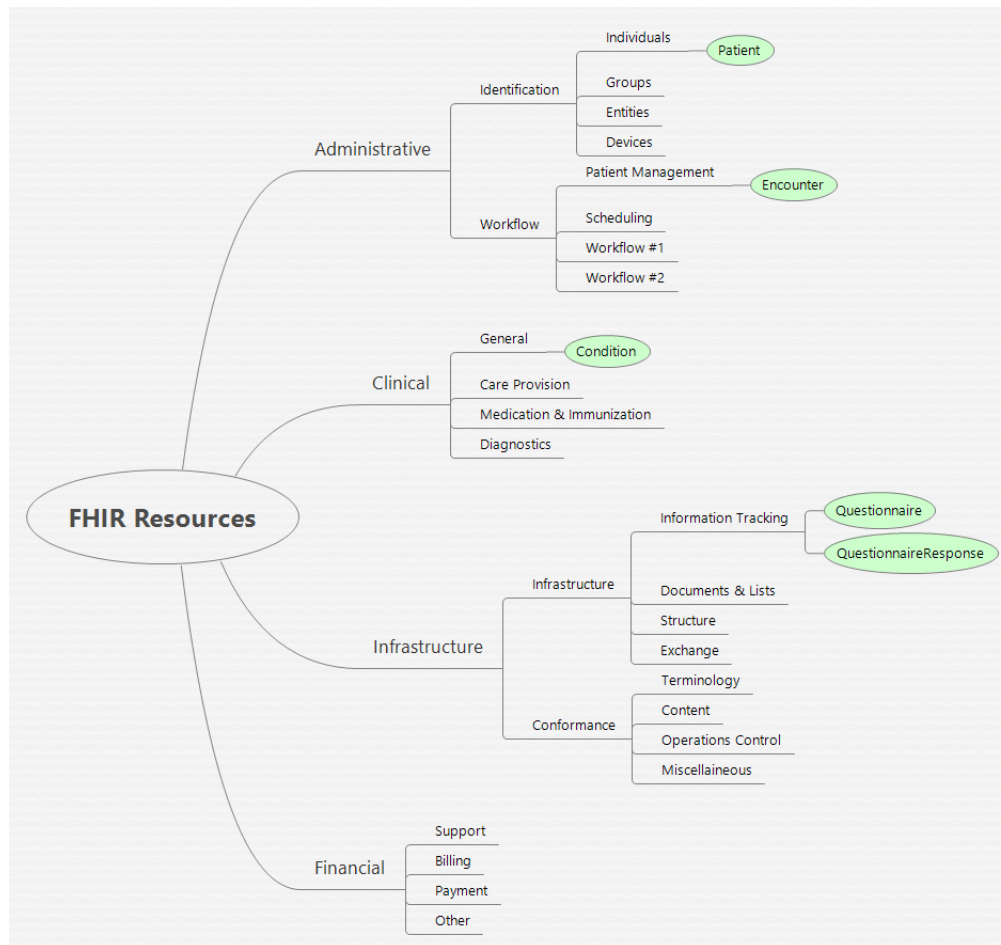


Figure 2.2.: FHIR resource categories

The principal components of FHIR are called “Resources”. A resource can be defined as a building block of associated information. Resources are able to reference each other and thus form complex information sets to work on. They can be categorized as shown in figure 2.2.

The diagram reveals some example resources like Patient, Encounter or Questionnaire. Every resource consists of a unique identifier, necessary meta data, a human readable description and the data which is capsulated by this resource. Besides, every resource can be extended to fit the needs of its environment. Figure 2.3 illustrates the formal definition of the Patient resource, while 2.4 shows the sample data of a Patient resource as XML document.[9]

Resources can be exchanged by an API, which allows direct access to them. According to FHIR the interface shall provide CRUD<sup>5</sup> functionality and additionally the possibility

<sup>5</sup>CRUD: Create, Read, Update, Delete

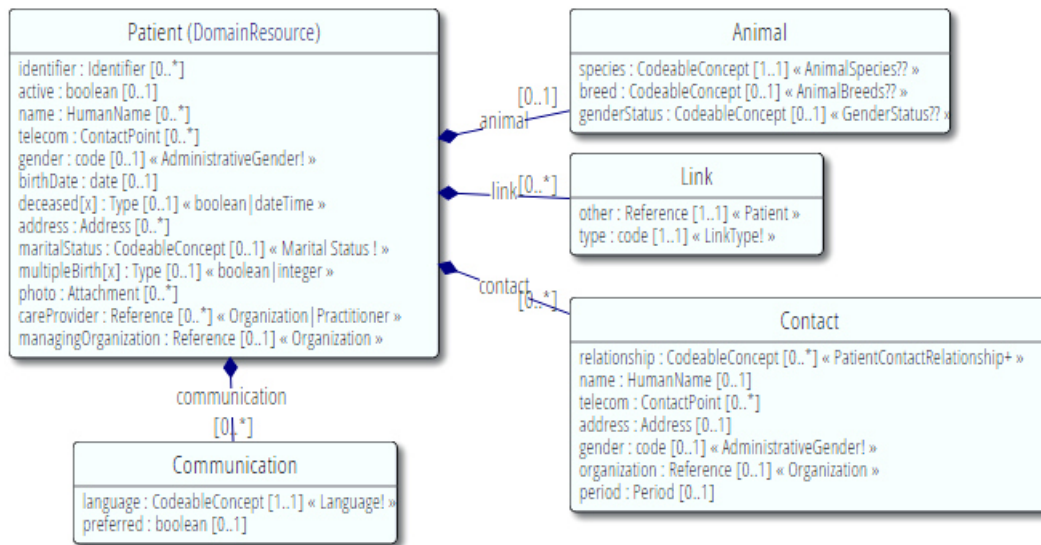


Figure 2.3.: FHIR Patient resource defined in UML[11]

to process search, history, transaction and self-defined operation requests. Alternatively FHIR also describes other methods of communication like messaging.[9]



Figure 2.4.: FHIR Patient resource sample data defined in XML[10]

## 2.4. Thinking Aloud

The term 'Thinking Aloud' refers to a type of experiment which basically includes a subject who is verbalising his thoughts while working on a task. A moderator introduces the experiment and the task and supervises the execution. If necessary, the moderator encourages the subject to continue talking, but refrains himself from any commentary or help otherwise. During a session, a video or audio tape is recorded, which can be analysed afterwards. Initially, the purpose of thinking aloud experiments was to understand the cognitive processes of problem-solving. Problem-solving is done when several mental steps are taken to find an answer to a question. Usually, these steps involve constructing a solution out of existing and new information and justifying this solution against possible alternatives. By analysing this process, researchers try to gain insight into human reasoning, the origin of differences between individuals and the cause of errors. Today, thinking aloud is widely spread and its applications range from the exploration of learning processes to reading comprehension. Moreover the resulting psychological model can serve as a basis for the construction of expert systems. Expert or knowledge-based systems aim at solving problems by following the same steps an individual would do.[12, 19]

Originally, the thinking aloud method derives from the field of psychological research. It succeeded the introspection method, which stated that events happening inside a subject's mind can be observed the same way as events in the outside world. In contrast to introspection, thinking aloud only refers to the contents of working memory, which can be verbalised and avoids any interpretation by the subject. The verbal protocols are treated as data that can be analysed by various researchers and enable an objective evaluation. The thinking aloud method has a long history as it was already conducted in 1945 by Duncker and in 1946 by de Groot.[13, 14] Its breakthrough was made in 1972 by Newell and Simon, who successfully constructed a detailed model of memory retrieval by using thinking aloud.[15] In the literature, both are often named as the inventors of the current technique. The realisation of expert systems using the thinking aloud approach began in the 1980s.[16]

The outcome of a thinking aloud experiment are video or audio tapes. Before analysis can be done, a psychological model needs to be defined. It serves as theory and can be verified or refined by the upcoming protocol analysis. The model describes the process of problem-solving and is based on psychological theory that is applied on a task analysis, which consists of the ideal way to perform a task and possible alternatives. By that, the model predicts the different steps someone will take when performing a task. To compare the model with the actual results of the experiment, all records have to be transcribed first. Afterwards, the text is segmented into single utterances and coded. The coding scheme relates the psychological model to the protocols on the basis of verbalisation theories. To adapt this general coding scheme to the concrete task of the experiment, it is refined by analysing approximately two pilot protocols. Basically, coding means that every segment is classified according to the type of phrase (e.g. question, guess) and its topic. Finally, the coded protocols are aligned with the psychological model. All coded segments are associated with the different steps of the model. If the protocols show

missing, additional or a different order of steps, the model can be revised.[17]

In addition to the original method, the thinking aloud approach has been modified in various ways to suit different research purposes. Reicks et al. (2003) used thinking aloud to explore the decision-making process of low-income African Americans while shopping groceries. The protocols were coded with regard to kinds of food and reasons for buying. Instead of aligning them to a psychological model, the coded segments were sorted by argument and weighed against each other to determine their importance.[18]

Another field of application is usability testing, where a subject is asked to think aloud while performing a task on a computer or prototype. A user interface, especially the informational aspects, should be structured according to the mental processes of a user. Thinking aloud can be applied to reveal existing discrepancies. Since usability testing does not aim at evaluating cognitive processes but at discovering problems that the user encounters, a psychological model or coded protocols are not required. Instead, analysis can be done on notes taken during the experiment or while analysing to the records.[19] In an experiment students of computer science with minimal training found 49% of all usability problems using the thinking aloud method with only two to three subjects. For each usability problem, the chance of discovery amounted to 40%.[20] Forgoing the detailed analysis, the implementation of thinking aloud requires less time, labour and training while still detecting a satisfactory number of usability problems. The thinking aloud method can be applied in every stage of software development, which might even be a paper-based prototype. If a cost-saving realisation is possible, an iterative design of the experiment can be implemented.[19, 21, 22] Besides the analysis the moderation can differ from the original approach as well. Depending on the given task, a less strict moderation might be suitable, e.g. to help the subject in certain conditions by providing hints. However, moderation rules should be defined clearly.[23] The main purpose of thinking aloud in this context is the discovery of usability problems. Moreover, someone can gain a general feeling of how users perform tasks and how they feel about the system. In detail, a thinking aloud experiment generates quantitative as well as qualitative data. Time, completion and error rates can be measured to quantify results that allow comparison, though the main focus lies on qualitative data. This includes a list of problems that users encountered. Because of the thinking aloud technique, the reason why a problem occurred can be identified as well. Moreover, even minor problems, which might annoy the user without clearly affecting the results, can be detected. Depending on the number of affected users, problems can be prioritised. To obtain significant results it is required that subjects are representative users in terms of their environment, knowledge and computer experience.[19, 21]

Alongside with the established experimental setup, variants have been developed. One of them is 'Constructive Interaction', which means that two subjects participate in a thinking aloud experiment at the same time. Both are allowed to work together, which enables a more natural verbalisation of thoughts. This setup, for example, is suited for children, for whom it might be difficult to simply follow the instructions. Negative aspects are, that more subjects are needed and different strategies for problem-solving of cooperating subjects can interfere with each other. Another alternative in usability



testing is 'Retrospective Testing'. This method includes showing the recorded video tape of the thinking aloud experiment to the subject after finishing all tasks. By that, more extensive comments can be collected and the number of required test persons decreases. However, the experiment will take much more time for the experimenter as well as for subjects. The 'Coachig method' differs completely from the original setup, because the subject is allowed to ask questions that will be answered by the moderator or an expert. On the one hand this will interrupt cognitive processing and problem-solving and does not reflect reality, where a user is facing the problems alone. On the other hand it might be the only way to generate any feedback in cultures where criticism is not communicated openly, e.g. Japan. Besides, indications of how to improve manuals and training can be gained.[23]

Concluding, the advantages and disadvantages of the thinking aloud method in usability testing should be summarised here. If analysis is simplified, thinking aloud can be applied with low costs. Even with a small number of subjects sufficient information can be generated. Compared to other techniques, thinking aloud does not suffer from retrospective distortion, as for instance post-test questionnaires do. Moreover, the experimenter can observe directly how subjects interact with the system and gain an insight of why errors occur. If a full analysis of data is done, thinking aloud becomes more laborious. Because of the time needed for each subject, only small groups of participants are realisable. Besides, verbalisation of thoughts is very unnatural to people and for some it is very hard to continue speaking. It cannot be excluded that the experimental setup influences the subject's behaviour and cognitive processes, though a distinct effect has not been proven so far. Furthermore, the results of a thinking aloud experiment are mostly qualitative and do not lend themselves for benchmarking and comparison. To balance strength and weaknesses a combination of different approaches is recommended.[19, 21, 23, 24]

## 3. Results

### 3.1. Requirement Analysis

In eHMIS, data is captured using static forms. Such forms are used when a new patient is registered in the system, when a disease is documented, a prescription is made etc. Every field in these forms has a corresponding entry in the database. Since the system 'knows' what data is stored, this method enables complex processing. Correctness of information can be verified, constraints can be checked and computations can be made. However, the most important advantage of this approach is that known data objects can be fully integrated into the application workflow. The system behaviour adapts to the existing information, for example when a known patient comes to the registration and the application displays the existing health record. Besides, the application relates different data objects with each other. A related person belongs to the patient he or she came with. A prescription of a medication can be the result of a condition a certain patient suffers from. All these connections are stored in the database and presented to the user.

Nevertheless, there is also information that only needs minimal processing. Especially, if this data varies a lot, dynamic questionnaires are a suited alternative to static forms. In eHMIS, there are several fields of application: Before tests or treatment are done the physician will ask the patient about his symptoms and health problems. The questions that the physician will ask highly depend on the patient's reason of coming. A pregnant woman might be asked about former pregnancies and complications whereas someone with diabetes might talk about his nutrition. This information consists of subjective descriptions and hardly any processing is done. That is why for these cases dynamic questionnaires can be implemented. Besides, there is also a use for reserachers who want to conduct studies on patients.

The requirements can be split into two parts: First, the creation, second, the usage of a dynamic questionnaire. The creation of a questionnaire implies the creation of questions. For every question, the user should be able to specify a text and the type of answer expected. Answers can be single or multiple choice or an input by the user. Constraints concerning the input, e.g. a data type or a maximum value, can be defined. A question can be flagged as required or optional. Furthermore, it should be possible to group questions and use them in different questionnaires. As long as not in use, everything should be alterable. All requirements of the questionnaire creation are shown as use case diagram in 3.1. A detailed use case description can be found in the Appendix.

The second part deals with the use of a dynamic questionnaire. During the examination of a patient, a health worker should be able to choose one of the completed dynamic

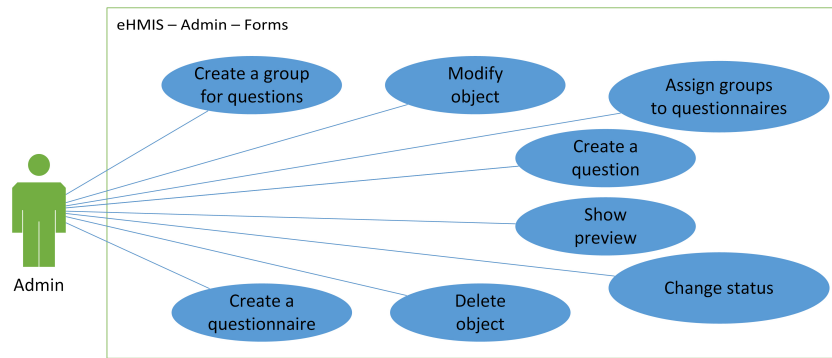


Figure 3.1.: Use case diagram of the creation of a dynamic questionnaire

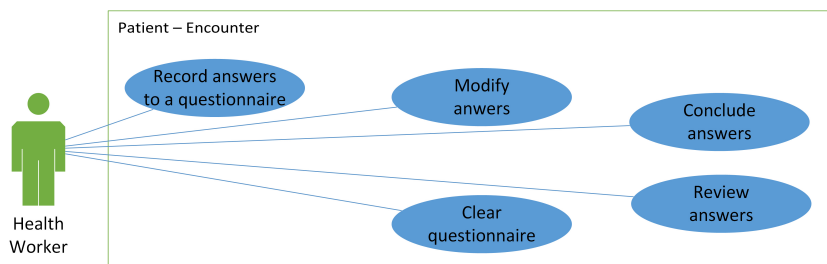


Figure 3.2.: Use case diagram of the application of a dynamic questionnaire

questionnaires. He should be able to record the answers of the patient with the option of modifying or complementing them later. The answers are part of the patient's health record and should be accessible at any time. If a questionnaire was completed or not should be visible for the user. The use case diagram in 3.2 illustrates all requirements (see Appendix for the use case description).

## 3.2. Modelling

### 3.2.1. FHIR Resources

Modelling began by taking a look at the data structures. eHMIS uses a static database, which enables complex processing and business logic. Thereby, a complete generic approach as presented in 2.1.1 is not applicable. The solution has to integrate well with the existing database architecture which is based on the FHIR® standard. In the DSTU2 ballot version, two resources (see section §2.3) for the implementation of a dynamic questionnaire are presented: The first resource allows the definition of questions and describes their grouping into sections and questionnaires. The second deals with the integration into the clinical workflow. The answers given by a patient are associated with his health record as well as with a questionnaire, forming a bridge between the different data. FHIR resources cannot be implemented directly. Instead, they guide the modelling process.

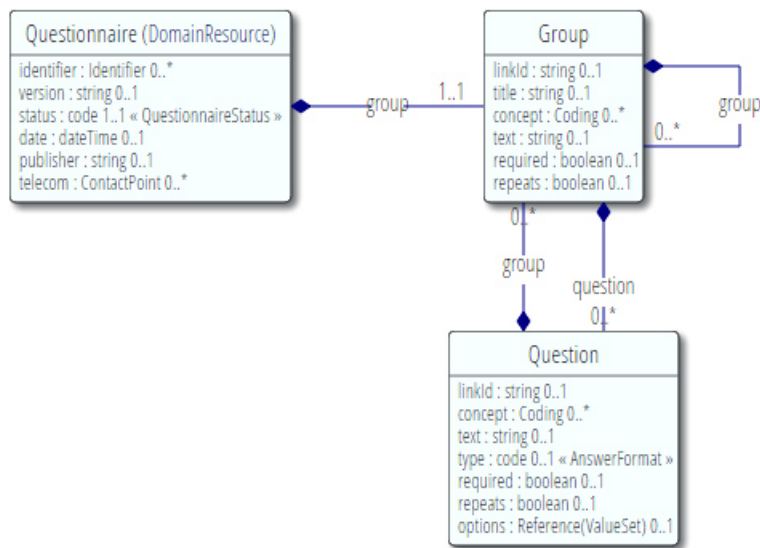


Figure 3.3.: FHIR Questionnaire resource [30]

According to FHIR “a questionnaire is an organized collection of questions intended to solicit information from patients, providers or other individuals involved in the healthcare domain”. [30] Different examples for use are named like medical or social history and clinical research forms. Since these examples match our idea of use, the provided data structure can be applied. [30] The diagram in 3.3 shows the first resource, concentrating on the definition of a dynamic questionnaire.

The architecture consists of three data objects. The questionnaire object mostly contains administrative information, like a version, a publisher and the date of the last change. Besides, a status is mentioned which will be discussed in 3.2.2. The questionnaire is associated with a single group object, which serves as a root group. Groups can be indefinitely nested and enable the user to build a hierarchy of groups and sub-groups within a questionnaire. Groups have a title and a description and can be compared to sections and sub-sections on traditional paper forms. If a group is repeatable, it can occur several times in the same questionnaire. The 'required' field indicates if a group can be skipped. Finally there is the question object. Every single question is encapsulated in one of these objects. The diagram displays a many-to-many relationship which means that a group can have several questions and a question can belong to multiple groups. The text field contains the actual question. Moreover, details concerning the expected answer can be specified. The user can choose between different answer formats, e.g. number, text, choice. If the user decides for single or multiple choice, options can be listed.

As stated before, the questionnaire is not related to patients in any way. This gap is filled by the questionnaire answers resource which provides the structure for recording answers to the questionnaires. It is illustrated by the diagram in 3.4.

The questionnaire answers resource is linked to a questionnaire as well as to an encounter

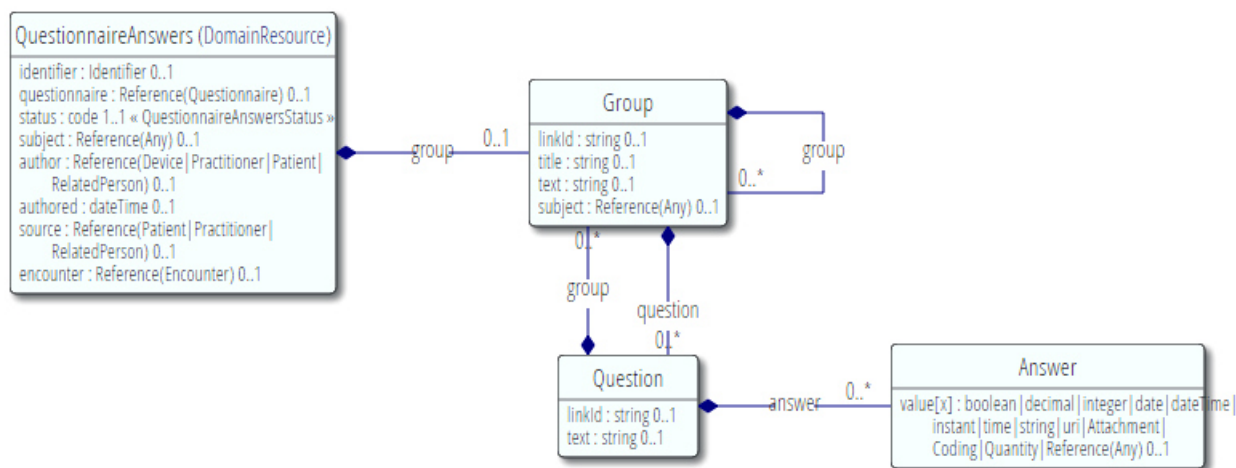


Figure 3.4.: FHIR QuestionnaireAnswer resource [31]

(see section §2.3), which is a central piece of the clinical workflow in eHMIS. Besides, it can be related to three different individuals: The author who records the responses, the source who is giving the answers and the subject who they apply to. However, the differentiation of subject and source is rarely expedient. The group and question object in figure 3.4 are intended to copy the hierarchy that was defined in the questionnaire resource and simply contain an id and a text. Both objects do not contain any extra information. The most important information is stored in the answer object. Every object contains a single answer of a patient to one question. The value type is specified by the answer format of the question object (figure 3.3). Although it cannot be realised just as displayed, a dynamic questionnaire requires this generic part.

### 3.2.1.1. Modifications

The architecture of FHIR turns out to be very flexible and powerful, but this flexibility comes with a high complexity not only during implementation but also in use. A user interface that presents indefinite nested groups and many-to-many relationships between questions, groups and questionnaires might confuse the user and comes with little advantages. For that reason, we decided to simplify the structure of the questionnaire resource. The relationship between questionnaire and group was flattened: A questionnaire can have several groups and each group can still belong to different questionnaires. However, nesting was abandoned. A title and a description field was added to the questionnaire because of the missing root group, which included this information originally. Moreover, every question is only assigned to one group. The new organisation is illustrated in figure 3.5, a detailed class diagram can be found in figure 3.7 on page 32.

The concepts and repeats fields were removed completely, because they seemed unlikely to be used in any context. Neither questions nor groups can occur twice in the same

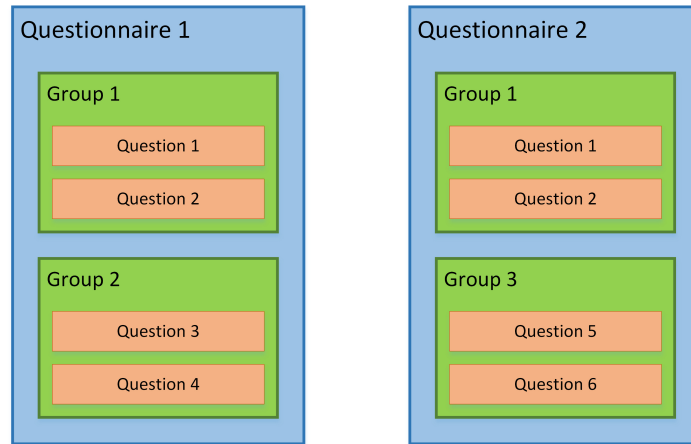


Figure 3.5.: Questionnaire data structure

questionnaire. The concept field allowed a classification, e.g. assigning LOINC codes, which is rarely applicable in relation to questionnaires. Besides, the 'required' field of the group object was omitted as it can be auto-computed by the corresponding field of the question object.

After simplifying the structures, some additional modifications and extensions were made. Similar to the questionnaire, a status field was added to the group objects which indicates the completeness of the group (see section 3.2.2 for details). The QuestionnaireAnswers resource was renamed to QuestionnaireFeedback to avoid confusion with the answer object. In addition, the answers were directly linked to the feedback object instead of unnecessarily copying the hierarchy of groups and questions which are already defined by the questionnaire resource (see figure 3.3 and figure 3.7).

One of the functional requirements was the option to define constraints concerning the answer of a question, e.g. a maximum value. Since FHIR is lacking this concept entirely, the questionnaire resource was modified and a validation object was added. It is associated with the question object and enables the definition of different constraints. Detailed information can be found in section 3.2.4.

### 3.2.2. States

The use case which describes the modification and deletion of a question, a group or a questionnaire seems simple at first. But on the second glance, challenges appear. What if a question is modified after answers of patients have been documented? The correction of a misspelling isn't a problem but what if the meaning is changed? All given answers might become incomprehensible. What happens to given answers if a question is deleted or moved to another questionnaire? Will they disappear? These challenges had to be addressed. Answers given by the patient can contain valuable information and need to be protected against unintended deletion or loss of meaning. However, the option to

modify or delete information is a functionality that cannot be abandoned.

Different states, which were already introduced by FHIR, can help solving this problem. The proposal of FHIR was extended in such a way that not only questionnaires and feedbacks but also groups have their own state. The different states of groups and questionnaires are specified below. Two terms indicate a divergent naming, the first term always refers to the group object.

- Draft
- Review (only groups)
- Active / Published
- Disabled / Retired

Those states work in a similar way. As long as a group or a questionnaire is in draft or review state, modifications and deletions are possible. Questions are seen as a part of a group and can be altered and deleted in the same way. In return, those objects cannot be applied in clinical work yet. Before a group can be assigned to a questionnaire, the status has to be changed to 'Active'. This action is irreversible, an active group cannot return into draft or review state. Moreover, an active group cannot be modified in any way or deleted, including the associated questions. The very same logic applies for the questionnaire. Groups can be assigned as long as the object is a draft. If a questionnaire is published, it is ready for use and neither the questionnaire itself nor its groups or questions can be altered or deleted. If the state of a group or questionnaire is set to disabled or rather retired, it can neither be modified or deleted nor applied anymore. Review is the only reversible state since a group in this state is allowed to change back to the draft state.

The feedback state serves another purpose. It can be on of the following:

- In Progress
- Completed
- Amended

As long as a feedback is in progress, the patient has not finished answering a questionnaire. Missing answers might be complemented. This interpretation differs from a completed questionnaire, where questions without answers might have been skipped on purpose for different reasons. By that, the status of a feedback helps in the interpretation of the patient's answers. A feedback that was completed, yet modified afterwards, will change to the state amended automatically. This behaviour becomes meaningful if different users are involved to call attention to later changes.

### **3.2.3. Answer Format**

A traditional questionnaire often contains different types of questions. Sometimes the subject is requested to just write down an answer, sometimes he can choose between predefined options. The same is expected of an electronic questionnaire. The system should provide a convenient interface allowing the user to work efficiently and also guiding

him. When the questions are created dynamically, the user has to specify this answer format. In our model, the format belongs to the question object as displayed in 3.3 on page 24. The user can choose between the following predefined values:

- Input
  - Text
  - Number
  - Date
- Single Choice
- Single Open Choice
- Multiple Choice
- Multiple Open Choice

Input questions are those which do not provide any options to choose from, but allow the user to enter his answer by typing it in. Besides, a data type can be chosen and the system will adapt the interface accordingly; a text, a number or a date field will be displayed which forbids any other inputs. Convenient features like a date picker are also possible. For all other answer formats, options can be defined. Single choice means that only one option can be picked whereas multiple choice questions permit several options to be selected. The keyword 'open' signalsises that in addition to the options, a text field will be shown. This field allows the user to write additional comments, which might be an explanation, a supplement or a simple note.

Obviously, this list does not cover all possible answer formats and others might be added as far as needed. However, for a more precise definition of the type of input expected, constraints can be applied.

#### **3.2.4. Constraints**

Validation is a big strength of electronic forms compared to conventional ones. The system can check all inputs of a user and reject invalid answers automatically without any person involved. By that, mistakes caused by accident or a wrong understanding can be avoided. Validation is done by validation rules or constraints which can be defined for any input field. Table 3.1 illustrates all applied constraints depending on the type of data entered.

One of the most important constraints is 'required' which can be used on any type of input. If this constraint is set, a questionnaire cannot be saved as long as the associated field is left empty. For numbers, a minimum and a maximum value can be defined, which prevents unrealistic values, e.g. an age above 150. A minimum or maximum set for a text is interpreted as a boundary respective to the length. A name might have a minimum value of 2 and a maximum of 30 characters. Besides, specific types of text like a mail address can be checked regarding their syntactical correctness. Invalid inputs are rejected. A date can be restricted to the past or the future in case it can be predicted.



Text	Number	Date
Required		
Minimum		
Maximum		
E-Mail		
URL		
		Past
		Future
Regular Expression		

Table 3.1.: Questionnaire constraints

For instance this applies on the birth date, the last pregnancy, the coming appointment etc.

Finally 'Regular Expression' can be understood as an extension mechanism. It allows practically any constraint to be defined, but requires advanced knowledge of regular expressions to be specified correctly. Though its use by clinical staff is very unlikely, it grants the user additional flexibility. The definition of all types of constraints is optional and the functionality of dynamic questionnaires is not affected if constraints are skipped.

### 3.2.5. Integration into the Application Workflow

Dynamic questionnaires can have several fields of application. However, the major part of data capturing is done by static forms to enable a more sophisticated form of processing. Initially, two areas in the clinical workflow were chosen to integrate questionnaires.

After passing registration and triage, a patient will usually meet a nurse or physician who will ask details about his health condition and his reason for coming. In eHMIS this section is called 'Signs and Symptoms' and can be found in the examination tab of the encounter in the patient's health record. The questions asked at this point highly depend on the patient's condition, the clinic he visits etc. Contrariwise two patients might be asked the same questions if they came for the same reason. Especially for cases that occur frequently, a suited questionnaire can be more convenient than a blank space. Obviously, a text area grants the user more freedom in writing than a questionnaire. A questionnaire has the advantage to guide the interview and allow a more efficient documentation. The idea is not to integrate one specific questionnaire but to allow the user to choose one or several among the existing ones.

Alongside with encounters eHMIS supports the definition of care plans. Compared to encounters, care plans are intended to persist for a longer period of time. They might be used to supervise a pregnancy or monitor HIV or cancer treatment. Those treatment plans often involve routine checkups where the current condition, progress, complications and other relevant information is recorded. Instead of programming static forms for each and every different care plan, dynamic questionnaires can be applied.

Dynamic questionnaires have to be used carefully, because the lack of information processing can cause negative consequences. If I request all medication of a patient from the application, medication that was mentioned in a questionnaire will not be returned. However, there are still fields of application left, e.g. research, where integration of dynamic questionnaires can be enlarged.

### 3.3. Implementation

eHMIS is a client-server application. More precisely it is browser-server based which means that the whole system is solely running on the server and can be accessed via a browser on the client side. The system is divided into three parts: the data layer, the logic and the presentation layer. Furthermore a subdivision into core, administrative, clinical, financial and infrastructure packages can be seen. This structure originates from the FHIR® standard. An overview including key classes and packages is shown in figure 3.6.

If a user accesses eHMIS with his browser, a request will be sent to the server. This request will reach the appropriate controller class in the presentation layer. The Form-Controller for instance is responsible for all requests concerning dynamic questionnaires. In the next step, the controller will either call the logic or the data layer depending on the type of request. A GET request is meant to retrieve information without causing any changes in the database. It is processed directly in the data layer by one of the repository classes. All other requests are forwarded to the service classes in the logic layer. In this layer, all parameters are validated first and bad requests are rejected. If testing is passed and all dependencies are considered, one of the factory classes in the data layer will be called and changes will be saved to the database by using the respective entity classes. The result of the request will be sent back to the controller in the presentation layer. The controller will return a visual feedback to be displayed in the client's browser.

A new feature always affects all layers which will be addressed one after another in this chapter.

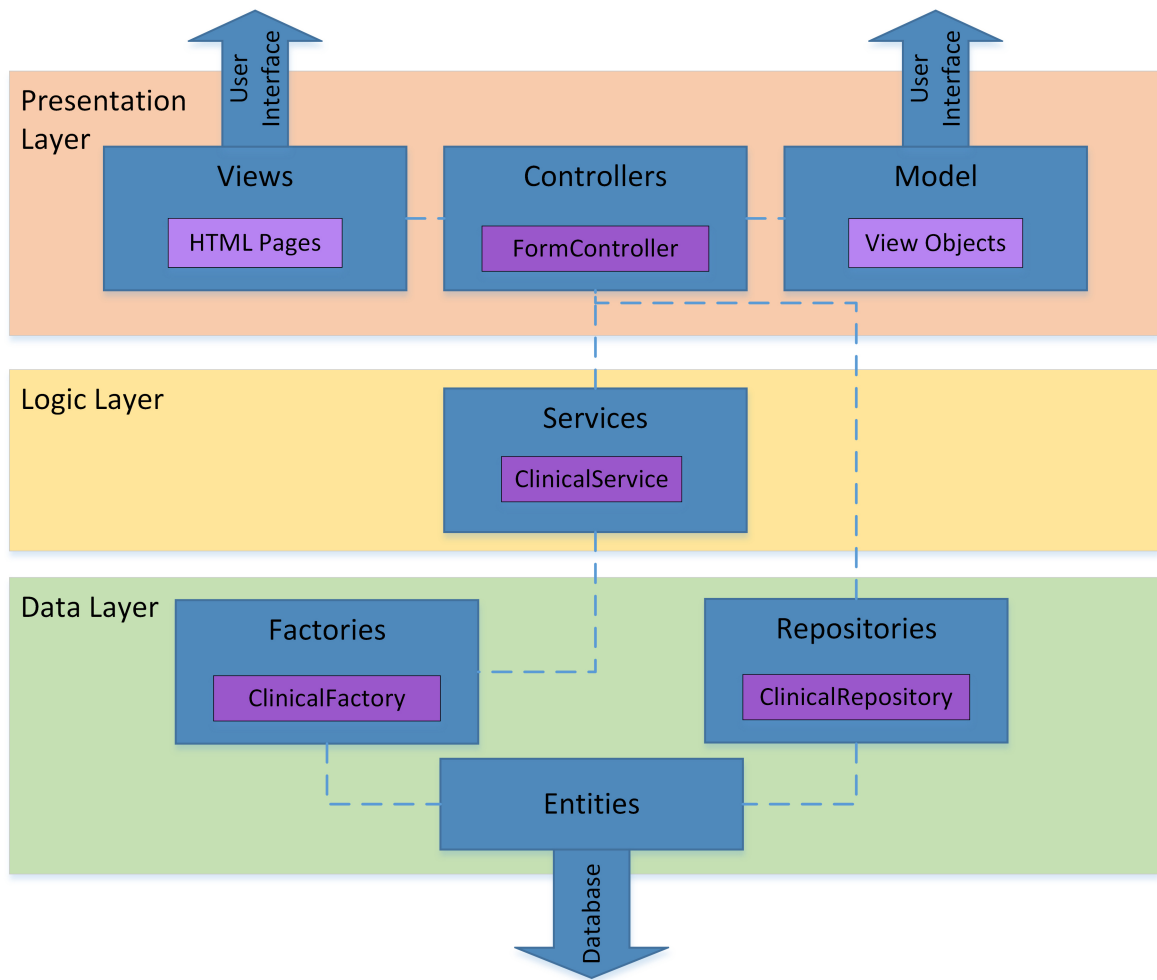


Figure 3.6.: eHMIS application structure

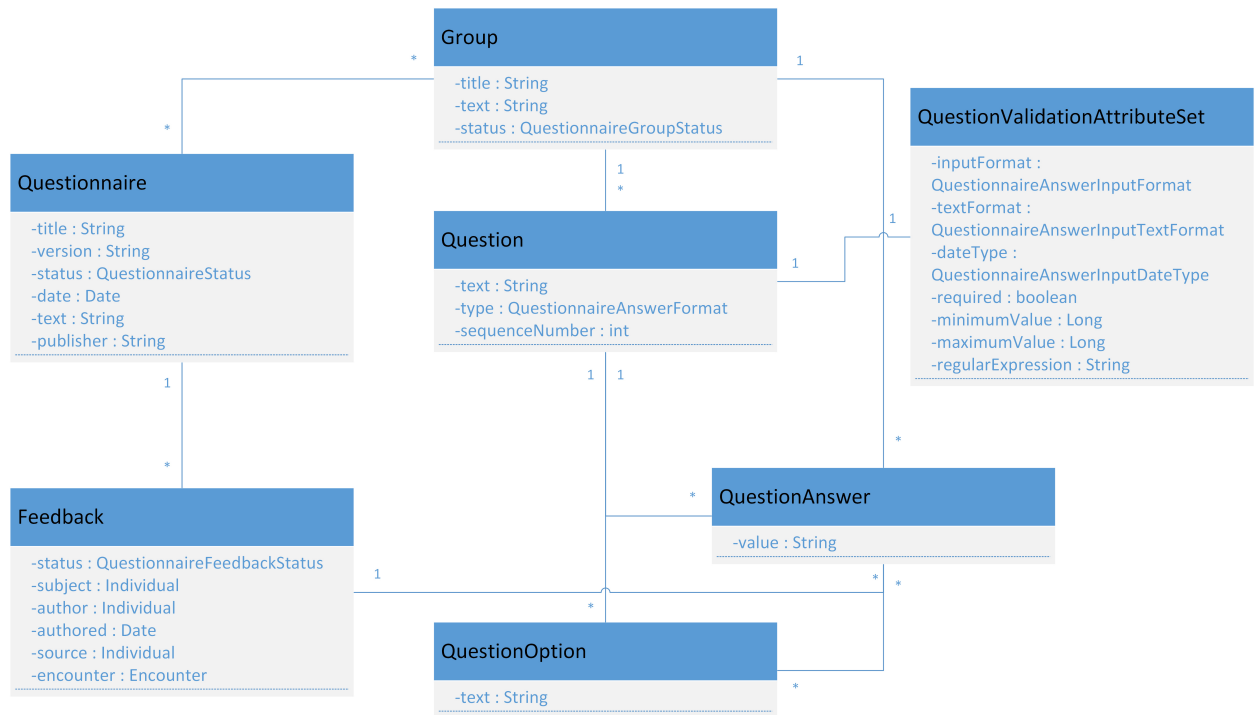


Figure 3.7.: Questionnaire entity classes

### 3.3.1. Data Layer

The database forms the foundation of eHMIS. It is generated from entity classes by Hibernate. Entities are annotated java classes. Every class is converted into a database table and every field into a column of this table. Among other things, the annotations serve the definition of relationships between tables. The class diagram in figure 3.7 displays all entities of the dynamic questionnaire. For a clear arrangement, methods were omitted and naming was simplified (e.g. QuestionnaireGroup renamed as Group). The class diagram is similar to the FHIR resources (see figure 3.3 and 3.4 on page 25). As described in section 3.2.1.1, some changes were adopted: The relationship between question and group was implemented as one-to-many so that every question belongs to a single group. Besides, the self-referencing of groups was left out which prevents nesting. The feedback object was directly related to the questionnaire and two additional classes were added: The QuestionValidationAttributeSet class includes all validation rules concerning a single question. Hence it is a one-to-one relationship. If a single- or multiple-choice question is created, every defined option is stored as QuestionOption. The object solely contains the option's text. Furthermore, all fields that can adopt only predefined values are implemented as enums. This applies to the different status fields, the answer format in question and the input validation formats of the QuestionValidationAttributeSet. Finally, the value field of QuestionAnswer needs to be discussed, which differs from the

description of FHIR. The standard defines the answer value as a generic field, which can adopt any data type. Since conventional databases don't support these kind of fields, it was declared a String type instead. Strings offer the highest flexibility as they can store not only text, but numbers, dates, boolean values etc. Though conversions become necessary (see section 3.3.3.1 for details).

In the application's workflow, entities are created by the factory classes which also belong to the data layer. The created entity objects are already connected to Hibernate. By that, changes of the object can be stored into the database using a simple 'save' command. An example can be seen below.

```
QuestionnaireFeedback questionnaire_feedback =  
  
    this.getDatabase().loadDataFactory(CoreFactory.class).  
    setResource(QuestionnaireFeedback.class, id);
```

The feedback object inherits from the resource object which needs to be respected during creation. In `setResource()`, a generic load method is called which will either create a new instance or retrieve the object from the database depending on the specified id. The new object will be returned to the logic layer.

Next to factories, the repository classes are an important part of the data layer. They are responsible for accessing data from the database but without providing any functionality of modification. The following method returns a sorted list of all questions of a specified group.

```
public List<QuestionnaireQuestion> getQuestionnaireGroupQuestions(int groupId) {  
  
    ExpressionTree group_questions = new ExpressionTree(ExpressionOperator.AND,  
        "group.id", BinaryOperator.EQUAL, groupId);  
    group_questions.setSortMap(new HashMap<>());  
    group_questions.getSortMap().put("sequenceNumber", SortOperator.ASCENDING);  
    return this.getDataObjects(QuestionnaireQuestion.class, group_questions);  
  
}
```

As displayed, composing of SQL statements is not necessary. Instead, a generic `ExpressionTree` is created. The passed parameters contain the selection criteria. The definition of a `SortMap` allows ordering of results. In the given example, all objects are returned in an ascending order according to their sequence number. In the last line, the required class is defined. All parameters are passed via the generic method `getDataObjects` to Hibernate and a list is returned. The simple access of information stored in the database does not require any advanced business logic. That is why, in contrast to factory classes, repositories are usually called directly from the presentation layer.

### 3.3.2. Logic Layer

The logic or business layer is the smallest module with only eight classes. The core are the services which are subdivided into administrative, clinical, core, financial and infrastructure services. Since the questionnaire belongs to the clinical resources according to FHIR, its features are integrated into the `ClinicalService` class. Services handle all requests that lead to a change in the database, e.g. the creation of a new object, modifications or assignments. The sole exception is deletion which has no advanced implementation and is handled by Hibernate directly. Persisting a change is proceeded in several steps: First, parameters are checked. If the request is valid, the corresponding entity object will be accessed and the changes are adopted. If necessary, values are auto-computed and complemented. In the last step, dependencies need to be updated, created or deleted. If all data is consistent, the modifications are saved. A small example is shown below, a more complex one can be found in the appendix.

```
public QuestionnaireGroup saveQuestionnaireGroup(int id , String title ,
String text , QuestionnaireGroupStatus status) {

    QuestionnaireGroup questionnaire_group = null;

    if ((id == 0 && status == null) || (id != 0 && (questionnaire_group =
this.getBusiness().getDatabase().select(
    QuestionnaireGroup.class , id)) != null &&
    (questionnaire_group.getStatus() == QuestionnaireGroupStatus.DRAFT ||
    questionnaire_group.getStatus() == QuestionnaireGroupStatus.REVIEW))) {

        if (status == null) {
            status = QuestionnaireGroupStatus.DRAFT;
        }
        questionnaire_group = this.getDataFactory().
            setQuestionnaireGroup(id , title , text , status);

        if (questionnaire_group != null) {
            this.getBusiness().getDatabase().save();
        }
    } else {
        return null;
    }
    return questionnaire_group;
}
```

This method is called whenever a group is created or modified. First of all, the parameters are checked inside the if-statement. If the id is zero and the status is not set, a new object shall be created. Otherwise, the group was already persisted and shall be modified. In this case the 'select'-request has to return the mentioned object. This object has to be

in draft or review state to allow any modification. If one of these tests fails, the method will return 'null' instantly and an error will be displayed to the user. If the parameters are valid, the object is created or rather updated using the factory class of the data layer. If necessary, the status field is initialized before. Finally, the changes are saved to the database and the object is returned to the presentation layer. This method is rather simple because no dependencies have to be considered.

Even if it is small, the logic layer is an essential part of the application as it contains most of the business logic. It can neither be auto-computed nor replaced by any framework or plugin.

### 3.3.3. Presentation Layer

The presentation layer is based on the Spring framework and therefore implements its MVC structure. MVC represents Model, View, Control. Correspondingly the layer consists of controller classes, model objects and views. To implement the dynamic questionnaire all components had to be extended.

Controllers are the core classes of the presentation layer. They are responsible for controlling the process flow. For the realisation of questionnaires, the FormController class was implemented. The controller contains 49 methods whereas 36 of them are bound to an URL. An URL represents a request by the user. This might be the display of all existing questionnaires, the details of a specific one, a form for the creation of a new one or a saving or deleting request. If such a method is called, the controller will retrieve the information from the data layer or rather redirect the request to the logic layer. In the majority of cases, an entity object or a list of entities will be returned. The controller will map the data from the entity to a view object, which is part of Spring's model and will be explained later. Afterwards the object will be bound to a view and both will be returned to the user to be displayed. Concluding, the controller is the interface to all other layers and manages the model objects and views.

The model consists of so-called view objects (VOs). View objects are passed to views and serve as a kind of 'data carrier'. Its fields can be accessed and modified by views on client-side as well as on the server. Their structure is very similar to entity classes, though the typical annotations are missing. From this, it follows that a questionnaire VO, a question VO, an answer VO etc. was implemented. View objects can be related to each other if necessary, although they usually only contain primitive data types. In addition, validation rules can be defined on fields of a view object. This is done by adding annotations to the respective field, e.g. @NotNull. More details regarding more complex validation can be found in section 3.3.3.1.

In contrast to controllers and VOs, views are not Java classes, but HTML pages, which can be displayed by a browser on client side. Usually, the user interface is composed of several nested views. The main menu might be another view than the sub menu, the actual content and the footer. The dynamic questionnaire features 15 views, which are again extending other ones. eHMIS is using Thymeleaf, a template engine that allows more sophisticated functionalities than HTML itself. By that, views cannot only use

nesting, but contain inheritances, loops, if-statements etc. As stated above, views can also access view objects and an input field can be linked directly to a data field of the VO. An example is shown below:

```
<tr data-th-if="{question_status.index} == 0 OR
  *{questions[__${question_status.index}__].groupId} !=
  *{questions[__${question_status.index}-1__].groupId}">
  <td class="font_bold" colspan="2">
    <div data-th-text="{questions[__${question_status.index}__].groupTitle}">
      &nbsp;
    </div>
  </td>
</tr>
```

This extract describes a conditional table row of a questionnaire, which will display the group title, if the beginning of a new section is reached. It is part of a loop over all questions of that questionnaire. 'data-th-if' represents an if-statement in Thymeleaf. The code checks whether the current question is the first one - the index would be zero - or if the group id differs from the group id of the previous question. If one of the statements is fulfilled, an extra table row is added. The group title is retrieved from the data field 'groupTitle' of the VO object and can be displayed.

Views do not contain any style definitions. The exact appearance of the user interface is defined by stylesheets which are applied on all views of the application and guarantee an uniform look of the user interface. An example for a view and its later appearance in the browser can be found in the Appendix.

### 3.3.3.1. Validation

In eHMIS, validations are found in almost every static form, whereupon an input field being required is the most frequent one. Those constraints are realised by Hibernate annotations. The framework offers various annotations, which are added to the corresponding fields in the view objects. Validation is done on client-side without any involvement of the server. If a user fills out a form and clicks on 'Save', all fields of the form are validated right away in the browser and an error is displayed if validation fails. Unfortunately this approach does not work for dynamic questionnaires. The answer of every question is stored in the value field of the corresponding answer VO. Since for every question different constraints can be defined, the field cannot be annotated statically. Instead, a dynamic validation is needed, where constraints are set during runtime. Fortunately, Hibernate provides the functionality of dynamic declaration of annotations. To realise this feature, the answer VO has to implement the DynamicModelObject interface which comes with two methods: `getValidationType` and `getValidationConstraints`. The latter returns a list of `ValidationConstraints` in which each object represents an annotation. In this method, all defined constraints are checked and the list is assembled accordingly. Afterwards, it is passed to the `FieldElementValidationProcessor` class, which forwards the constraints to the Hibernate framework.



The `FieldElementValidationProcessor` plays a key role in input validation and can also be used to overcome another challenge. As mentioned before, the answer value is defined as `String` internally. This is unproblematic, unless it is passed to a view. The user interface should provide an appropriate input field to enter an answer depending on the data type, e.g. a date picker for questions that expect a date as answer. This can be achieved by implementing the other interface method: `getValidationType`.

```

@Override
public ValidationType getValidationType ()
{
    if (questionType == QuestionnaireAnswerFormat.INPUT) {
        if (questionValidationSetInputFormat ==
            QuestionnaireAnswerInputFormat.TEXT) {
            return ValidationType.TEXT;
        } else if (questionValidationSetInputFormat ==
            QuestionnaireAnswerInputFormat.NUMBER) {
            return ValidationType.NUMBER;
        } else if (questionValidationSetInputFormat ==
            QuestionnaireAnswerInputFormat.DATE) {
            return ValidationType.DATE;
        }
    }
    return ValidationType.TEXT;
}

```

First the question type is checked, because a single or multiple choice question does not require a data type. Next the `inputFormat` of the `QuestionValidationAttributeSet` is retrieved. Since the `Question` and the `QuestionValidationAttributeSet` entity have a one-to-one relationship, they were combined as an answer view object in the presentation layer to simplify access. This method returns the appropriate data type or `TEXT` as default type which is again passed to the `FieldElementValidationProcessor`. Instead of Hibernate, the data type is processed by Thymeleaf engine.

All constraints are only applied on input fields which means that currently no validation is done on single or multiple choice questions. At this point enhancements are possible.

### 3.4. User Interface Design

eHMIS aims at providing a consistent user interface. Common style sheets, which are used throughout the whole application, maintain the same look for all forms, menus and content fields. Besides, all pages are built in a uniform structure, which is illustrated in figure 3.8. Before the actual implementation, the design was discussed on the basis of paper prototypes.

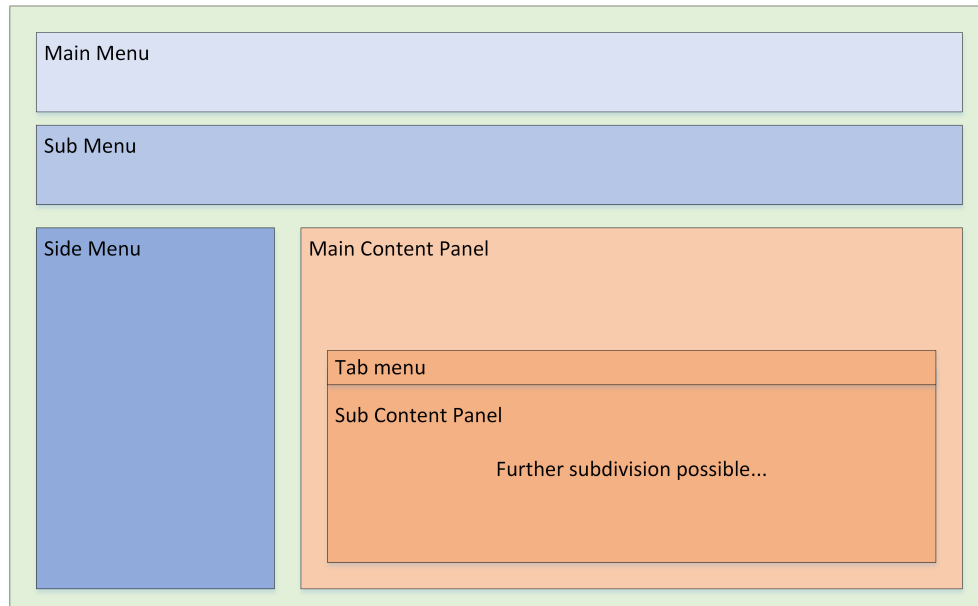


Figure 3.8.: Layout structure

### 3.4.1. Questionnaire Creation

For the creation of a questionnaire, the user has to navigate to the 'Form' section of the 'Admin' menu. In the sub menu, groups or questionnaires can be selected according to their state. Different states might correspond to different areas of responsibility, though that depends on structures of the healthcare facility.

The screenshot in figure 3.9 displays a selected group in draft state. The chosen items in the sub and side menu are highlighted. The main content panel contains all general information of the selected group. Besides, the buttons for modification, deletion, change of status and preview can be found there. The tab menu beneath only includes the 'Questions' tab. In the sub content panel, a table of all questions of that group is shown. For every question, its number, text, answer type and all defined validations and options are listed. Moreover, the functionalities of adding, editing and deleting of questions are situated there.

If the user selects 'Add' or 'Edit' in the 'Questions' sub content panel, a form in a separate window appears. The fields of the form automatically and immediately adapt to the answer format and the input type, which the user chooses. The different appearances of the form are illustrated in figure 3.10<sup>1</sup>. A red star marks the field as required.

Neither the form nor the representation in a table convey the final look of the dynamic questionnaire to the user. That is why a preview of groups and questionnaires can be opened, which displays the questions how they will look like when the questionnaire is

<sup>1</sup>The functionality to specify a sequence number was realised after the end of the experiment.

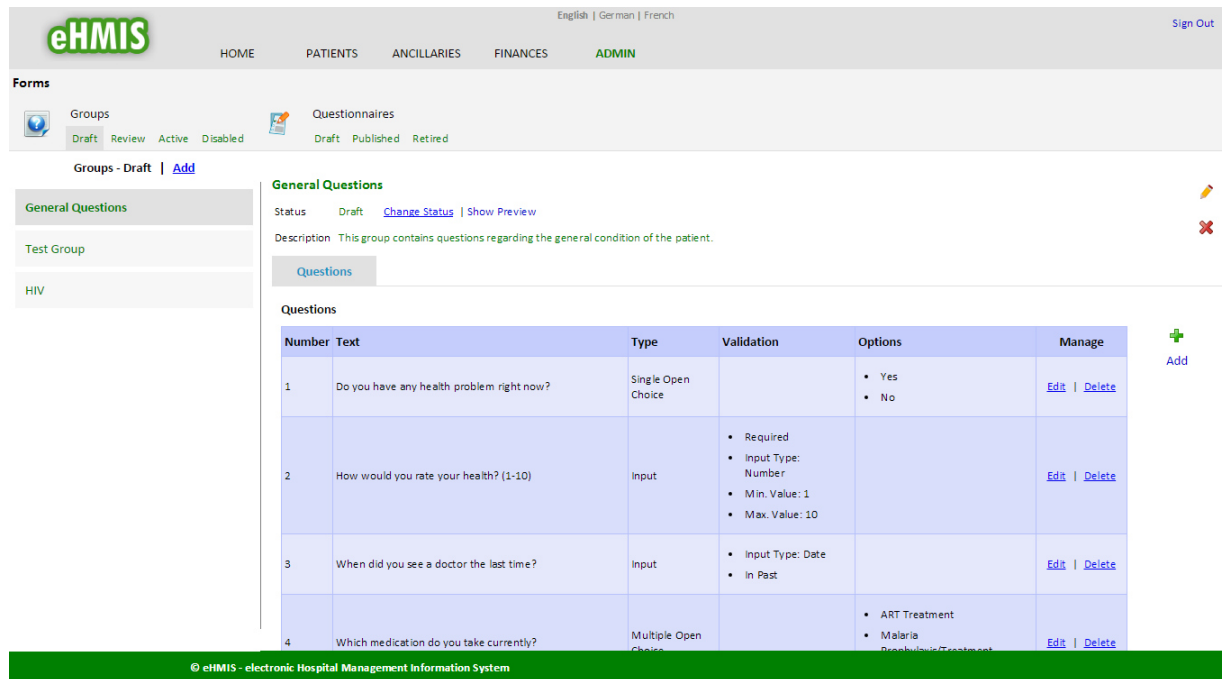


Figure 3.9.: Screenshot of the groups' section

in use (see figure 3.11). The preview contains the questions and input fields to enter or options to select an answer. The chosen input type is considered and, depending on the browser, a date picker is available for the correspondent fields.

The look and structure of the 'Questionnaire' section is very similar to the one of 'Groups'. Instead of 'Questions' a 'Groups' and an 'Other Information' tab can be found below the main content panel. If the user selects 'Groups', a list of all assigned groups including their title, status and description is displayed. A form containing a selection list allows the user to manage the assigned groups. 'Other Information' will unveil another sub sub menu for the definition of contacts and identifiers regarding the questionnaire. Additional screenshots can be found in the Appendix.

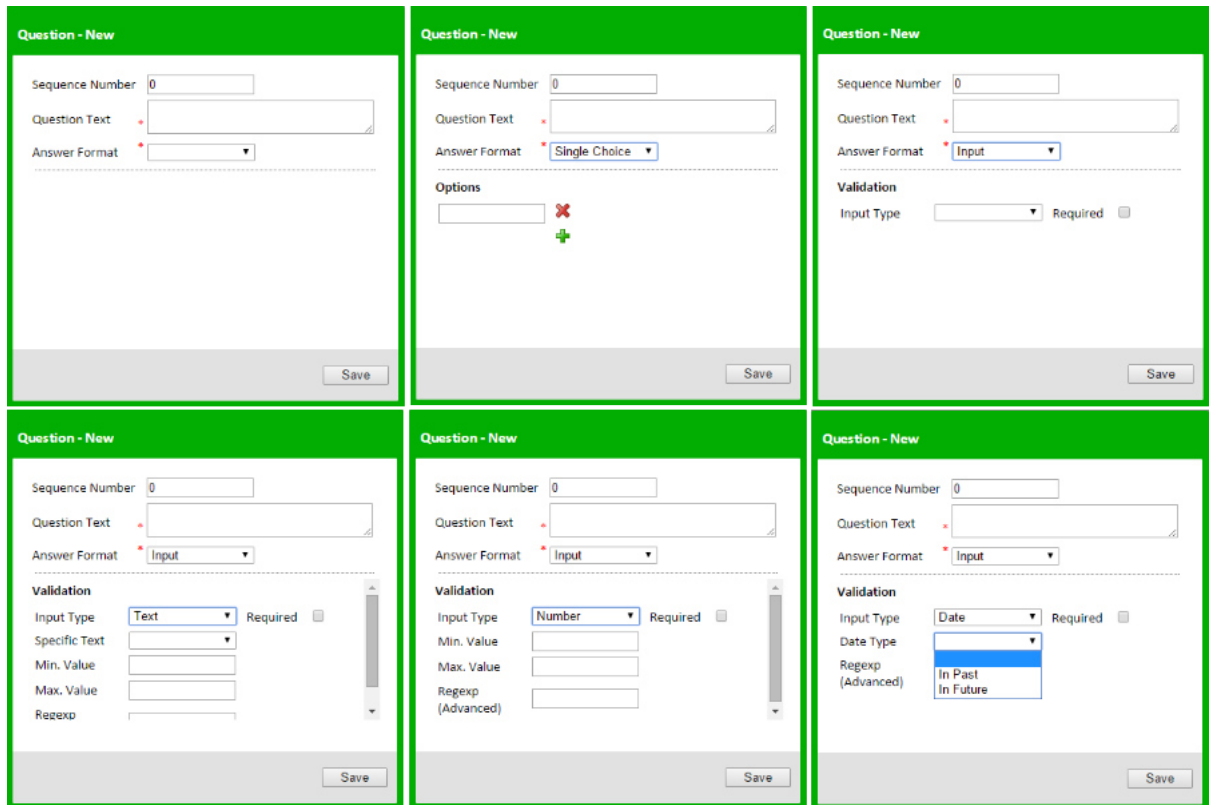


Figure 3.10.: Screenshots of the question creation form

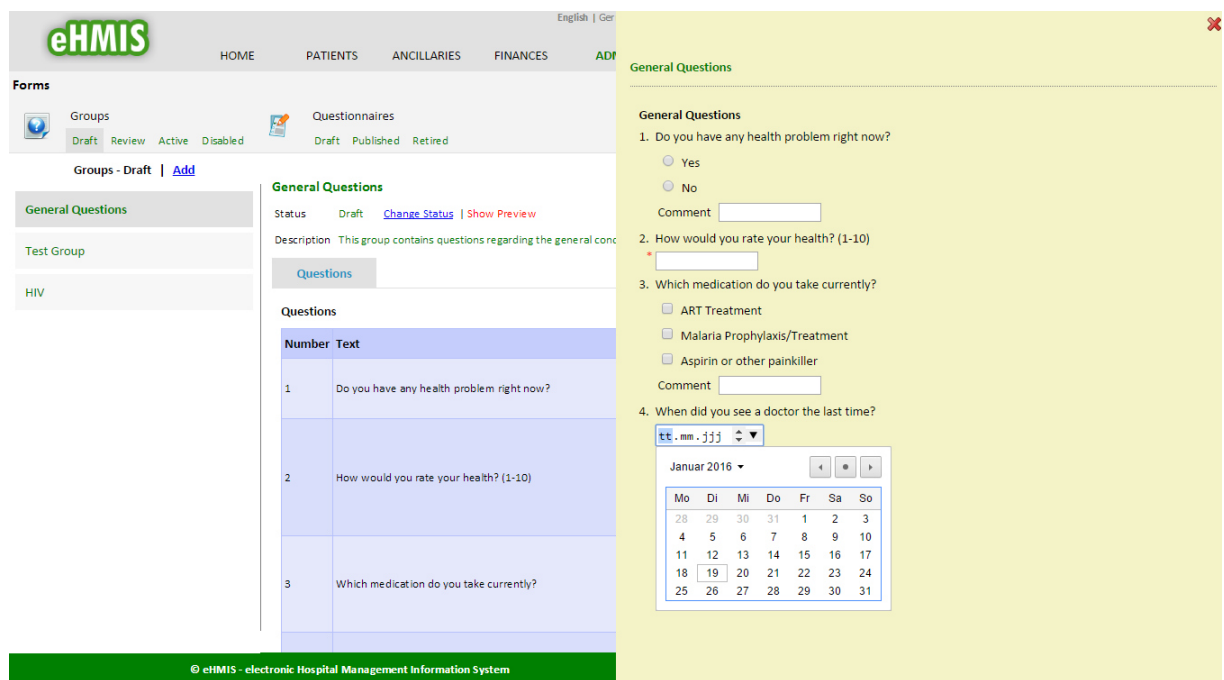


Figure 3.11.: Screenshot of preview function

The screenshot shows a patient record for 'Doe John' (Male, Active). The 'Encounters' tab is selected, and the 'Examination' sub-tab is active. Under the 'Signs and Symptoms' section, there is a list of questionnaires:

Questionnaire Name	Last Modified	Actions
test	01/14/2016	More Details, Edit, Delete
General Questionnaire	01/19/2016	More Details, Edit, Delete

Figure 3.12.: Screenshot of feedback list

### 3.4.2. Questionnaire Usage

A dynamic questionnaire can be used once it is published. One possible application is during the examination of the patient. For this purpose, the user navigates to the patient's record first and selects the 'Encounter' tab. The user can choose one of the existing encounters and subsequently open the 'Examination' tab. The section 'Signs and Symptoms' provides a list of all answered, partly answered and started questionnaires which are related to the chosen encounter. As shown in figure 3.12 the list contains the name and the date of the last modification for each questionnaire. On the right side, a button for modification and deletion can be seen.

If the user clicks on "More details", the corresponding list entry expands, as illustrated in figure 3.13. The upper part names author<sup>2</sup> and status of the questionnaire, which can be changed via 'Change status'. Below, all questions and the corresponding answers of the patient are displayed. Given answers can be modified or complemented by clicking on "Edit".

To record the answers of a questionnaire, the user selects "Add" at the right top of the section and chooses one of the existing questionnaires. Afterwards, a form with the dynamic questionnaire is displayed instantly (see figure 3.14). It contains all questions and the defined options to choose from or an input field. A single choice question provides radio buttons whereas a multiple choice question has check boxes. Open choices have an additional text field for comments. Input fields only allow the specified input type text, number or date. After the user has entered all answers, he clicks on 'Save'. The fields are

<sup>2</sup>At the end of development, the implementation of user and role management was not completed yet. Hence no author is displayed.

General Questionnaire - 01/19/2016  

More Details

Author | Status In Progress [Change Status](#)

**Answers**

---

**Answers** [Edit](#)

1. Do you have any health problem right now?
  - Yes

Comment Flu
2. How would you rate your health? (1-10)  
8
3. Which medication do you take currently?
  - Aspirin or other painkiller

Comment not frequently
4. When did you see a doctor the last time?  
2015-02-20

Figure 3.13.: Screenshot of an expanded feedback

validated and an error is shown if a field does not satisfy the requirements. Otherwise, the answers are saved and displayed.

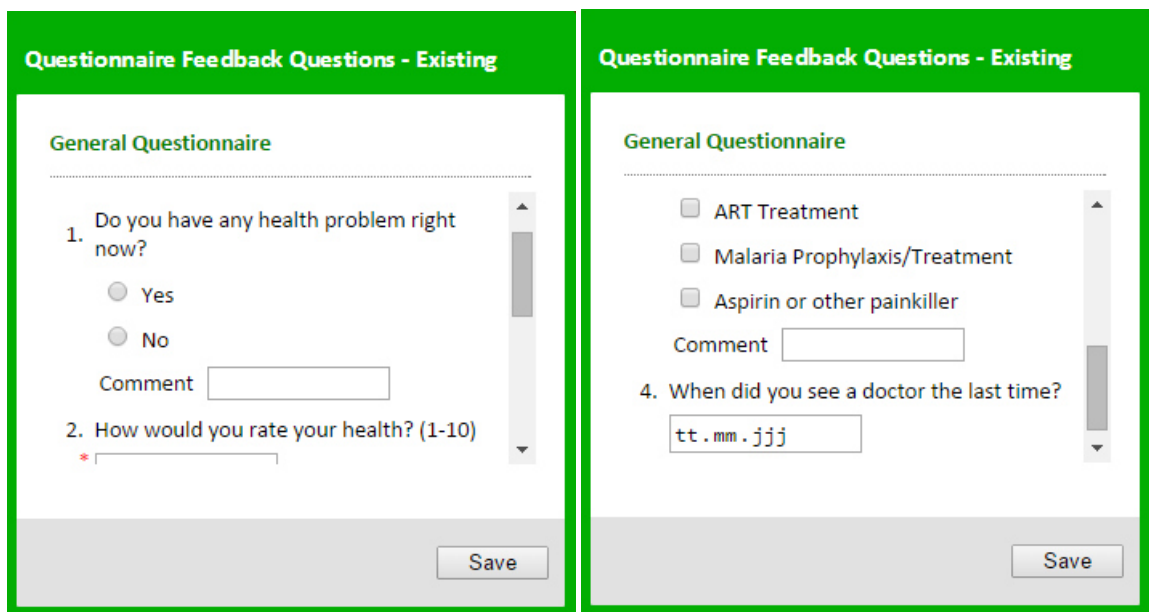


Figure 3.14.: Screenshots of a dynamic questionnaire



## 3.5. Experiment

### 3.5.1. Preparation

After the technical realisation, an experiment shall prove the usability of the new functionality. Usability is a pre-condition for successful adoption of the module and by that an essential subgoal for the longterm objective of enabling users to adapt the application to local and situational needs by themselves. The experiment shall allow a general assessment of the usability as well as encourage improvements. In detail, the following questions shall be answered:

- Is the feature usable by the target group in the target environment?
- Which functionalities are missing?
- Which difficulties do users have? Which weaknesses in the interface and the workflow can be uncovered?
- What impression about the general satisfaction of the users can be gained?

In the past, eHMIS was deployed in different hospitals, most of which were located in rural regions. For the experiment, Tororo Hospital in Eastern Uganda was chosen. On the one hand, the hospital meets the target group as it used to have the first version of eHMIS for some time and might deploy it in future. On the other hand, the size of the hospital allows the recruitment of a sufficient number of subjects. Different professions of the participants are sought after. In routine operation, dynamic questionnaires might be created by IT officers or comparable staff. But it cannot be excluded that nurses or people with other healthcare professions might be prospective users. Moreover, a basic computer knowledge is required. This includes the operation of a laptop with keyboard and mouse or touchpad as well as a general understanding of user interfaces, e.g. the function of a button compared to an input field. Prior experiences with eHMIS are not needed.

For the experiment, the Thinking Aloud method was chosen because of its easy feasibility. It requires only few equipment and is time-saving for test persons. This is particularly important as the experiment will be conducted on a working day. However, the method comes with its own challenges. To gain evaluable results, the test subjects have to participate very actively and adopt the unfamiliar behaviour of pronouncing their thoughts. To minimize the risk of failure, a preliminary experiment with only two test subjects will be set up. Subsequently, adjustments can be done if necessary. Another challenge is the communication of the method to the subjects. For that reason, a video was shot. The video lasts approximately three minutes and shows a tablet on which Matatu - a famous and simple card game - is played, while thinking aloud, which enables the subjects to listen to the thoughts of the player. In addition to the abstract explanation, this video will serve as a demonstrative example.

The subjects will be asked to create a dynamic questionnaire. This includes the creation of a group, the definition of questions and the activation of that group by a status change. Following this, a questionnaire is created, the group is assigned to that questionnaire and the questionnaire is published. The subjects shall create five questions and will be

encouraged to try different answer types. The amount of five questions was chosen so that, on one side, the test person doesn't lose motivation, but, on the other side, has the possibility to try different configurations and gain some routine. The first question shall be fixed. Concerning the other four, the subjects are either allowed to think of their own or to take predefined ones. The predefined questions have to meet several demands: Obviously they should include different answer types. To be as close to reality as possible, they should be part of an existing and established questionnaire. Besides, the content shouldn't be too specific to enable an easy comprehension. Based on these requirements, the introduction part of the WHOQOL-HIV BREF<sup>3</sup> was selected. The document that was handed over to the subjects can be found in the Appendix. It contains the following questions:

- What is your gender? Male / Female
- How old are you? (age in years)
- How is your health? Very poor / Poor / Neither Poor nor Good / Good / Very Good
- Do you consider yourself currently as ill? Yes / No
- If there is something wrong with you, what do you think it is?

In addition to the predefined questions, further materials were prepared. A short questionnaire was written to collect some personal information about each subject. This information includes gender, profession and years of experience in it. Besides, the experience with computers and the average usage of them is asked. A question concerning the age was omitted on purpose because in Ugandan culture the age is regarded as sensitive information and asking might be considered as rude, especially by women. Furthermore, a document containing short instructions was prepared. It illustrates the structure of a questionnaire and its groups and questions and summarises the necessary steps for its creation. Moreover the answer formats are listed and explained to guide and help the subjects during the experiment. Finally a usability questionnaire was created, which can serve as alternative in case the Thinking Aloud method cannot be applied. This questionnaire adopts the 'System Usability Scale' (SUS) to measure usability. It consists of ten statements and a 5 point Likert scale, which allows the subject to agree or disagree with these statements. It was published by the Digital Equipment Co Ltd. and is considered simple and reliable, which was the main reason for its selection.[32]

All mentioned documents can be found in the Appendix.

### 3.5.2. Implementation

The experiment was implemented in Tororo Hospital in a separate conference room. In addition to the subject, the moderator and the chief developer of eHMIS were present, though he mostly stayed in the background. The role of the moderator was performed by myself. Unfortunately a few disturbances from other people couldn't be avoided, but were reduced as much as possible. The experiment was conducted on a laptop with a

---

<sup>3</sup>WHO Quality of Life Questionnaire (brief version) with adaptations concerning people with HIV

local installation of eHMIS. A mouse was provided. The subjects were called in one after another.

First, all subjects were asked to fill out the questionnaire concerning their personal information. Afterwards, the moderator presented the experiment by adhering to the following procedure:

- Introduction of himself
- Explanation of the background of the experiment
- Explanation of the Thinking Aloud method including a presentation of the video
- Presentation of a completed dynamic questionnaire
- Assignment of tasks
- Explanation of the structure of a questionnaire
- Short demonstration of the required steps for the creation of a questionnaire
- Handover of the predefined questions and short instructions
- Reminder of Thinking Aloud including motivation

Following recording was started and the experiment began. All subjects were allowed to ask questions, which was done very frequently. Whenever the subject became quiet, the moderator tried to remind him to continue talking. Besides, the test person was motivated to try different answer types. If answer types were completely misunderstood, the moderator suggested the preview function to point the subject to his mistake.

After the subject completed all tasks the moderator presented the completed questionnaire in use. The experiment was ended and a small money present was given to each participant. Editing time took around 30 minutes per test person.

On the first day, the preliminary experiment with two participants was conducted. The results were promising and only some small modifications were adopted. The instruction to mandatorily take the first predefined question was abandoned because it seemed too restrictive and disturbed people in their planning of the questionnaire. Besides, mandatory fields and answer types were explained in more detail as the first subjects encountered difficulties in understanding the concepts. The main experiment, which considered these changes, was conducted on the next day and included eight subjects.

### **3.5.3. Evaluation**

Ten subjects participated in the experiment whereof nine were female. All of them worked at Tororo Hospital in medical or administrative professions and were following a different occupation, e.g. as nurse, medical social worker or record assistant. Experience in this profession ranged from 4 months to 30 years with an average of 9,133 years. In the beginning, all test persons were asked how they rank their computer experience and how many hours they spend on the computer per week. The answers are displayed in figure 3.15. Except two, everyone stated to be “medium” experienced. However, answers to the second question show more variations and might be more accurate. A table containing all details about the participants can be found in the Appendix.

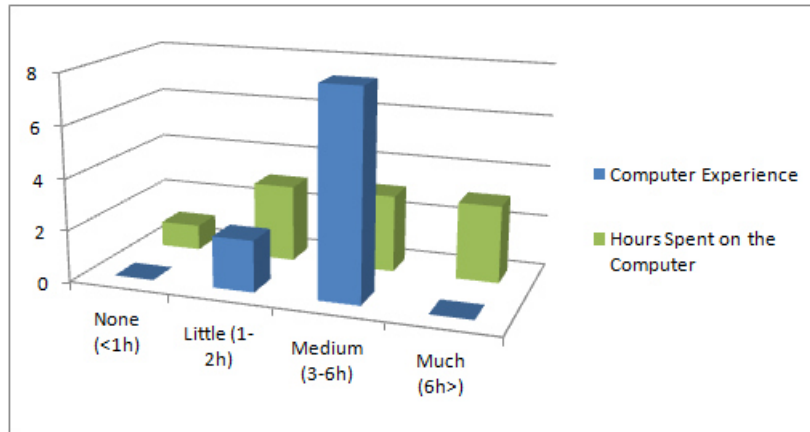


Figure 3.15.: Computer experience of subjects

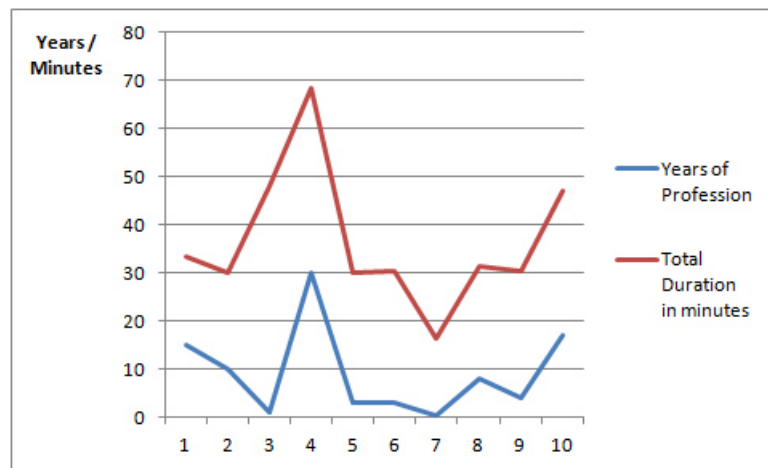


Figure 3.16.: Duration of the experiment compared to years in profession

For every participant, quantitative data was recorded and compared. This includes total duration of the experiment, time taken for the first and the fifth question and the usage of different answer types, constraints and other functions as well as the general understanding of them in the beginning compared to the end.

First, the overall length of the experiment should be discussed here. The measured period includes a final presentation of the created questionnaire but omits the introduction and instructions given in the beginning. In average, subjects took 37,03 minutes to complete all tasks with a minimum of 16,35 and a maximum of 68,37 minutes. As displayed in figure 3.16, duration is proportional to the years that people worked in their profession. Because of the cultural background, age of the participants was not asked, but this result leads to the assumption that the time needed might depend on the age rather than on the professional experience.

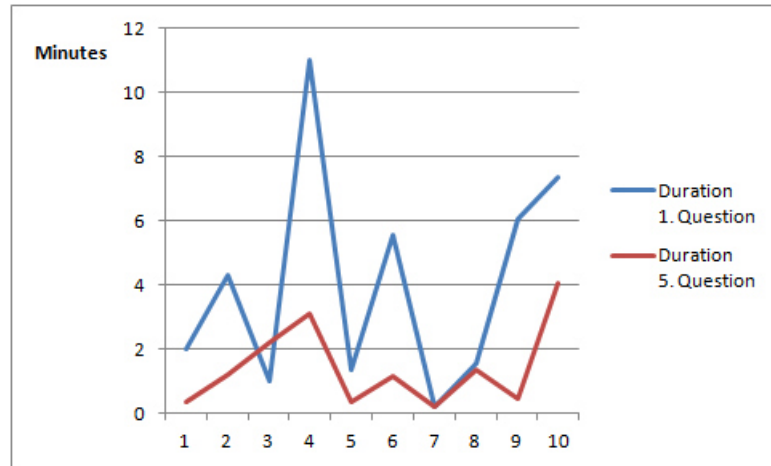


Figure 3.17.: Time taken for the first compared to the fifth question

Beside the total duration, the time that subjects took to create the first question was compared to the time for the fifth one (see figure 3.17). For the first question 20 seconds to 11 minutes were needed. In contrast test persons took 20 seconds to 4,08 minutes for the fifth question. The average decreased from 4,04 to 1,45 minutes. All participants increased in speed except one, whose last question was more complex and contained options in opposite to the first one. These results imply that little practice is already sufficient to understand the workflow.

In total 54 questions were created. The diagram in figure 3.18 shows the different answer types that were used. More than half of all questions were declared as 'Input'. 16 'Single Choice' and 3 'Multiple Choice' questions were created whereof 3 were open choices. Multiple Choice questions were only used by those who wanted to try out the answer type, so this type might rarely be applicable. All others were utilised to construct real life questions. All participants decided to define their own questions and at most one question was taken from the provided example sheet. This result indicates that the possible answer types satisfied the needs. Five subjects used the optional functionality of specifying constraints. The option to put a minimum or maximum value was adopted four times, two test persons marked a question as required and one declared a date to be in the past. The constraints 'String Type' and 'Regex' were not inserted. Summarizing, there seems to be a field of application for constraints, but their correct application require some understanding. Additionally, everyone was pointed to the 'Preview' functionality to check the correctness and visualize the questionnaire, but only one participant used this functionality on his own account.

Based on their errors and questions, the subject's general understanding of the answer types was rated. In this case, an error was defined as an action of a test person that resulted in an unexpected behaviour of the system. Comprehension was assessed as 'not at all', 'medium' or 'complete' in the beginning and the end of the experiment. The results are illustrated in figure 3.19. For most of the participants, the different answer

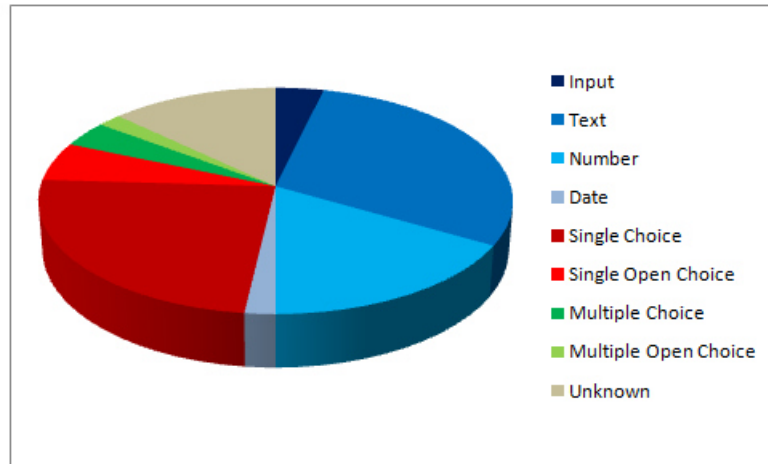


Figure 3.18.: Answer types used

types were not intuitive and additional explanations were needed. In the end, three completely understood the types and acted accordingly. Half of the subjects reached a sufficient understanding of those types, which they used, while two remained uncertain up to the end. Interest and the way each of them got involved with the different answer types turned out to be a major factor of success at this point. A correlation between comprehension and computer experience could not be found.

Next to the quantitative part, the main results of a thinking aloud experiment are qualitative data. For each subject, all errors (for definition see above) and comments on the software were collected. Besides, all questions were considered. The data was grouped according to topic and the underlying cause with the aim of identifying existing usability problems of the software. The results are summarised in table 3.2, only problems which affected more than one person were included.<sup>4</sup>

Deletion became a big issue. One subject accidentally deleted his questionnaire and had to recreate all questions. A second one would have done the same, but was warned before doing so. These incidents illustrate the importance of a confirmation dialogue, which has to be displayed and confirmed before deletion is performed. An integration in the whole application should be considered because of the high risk of permanent data loss. Beside that, three participants complained about the missing functionality of ordering questions, especially because modified questions are moved to the end automatically. That is why an implementation of ordering for questions as well as groups is highly recommended.<sup>5</sup>

During the experiment not only missing functions but also unfavourable functionality was detected. Closing a form by clicking in empty space is a very convenient feature for experienced computer users, as it provides a quick alternative to the 'Close'-button. However, in the experiment it turned out to be problematic for novice users, who tend to

<sup>4</sup>The number of affected users can be understood as a minimum value of those who expressed their problem.

<sup>5</sup>This problem was partly addressed, so that a basic ordering of questions is already possible today.

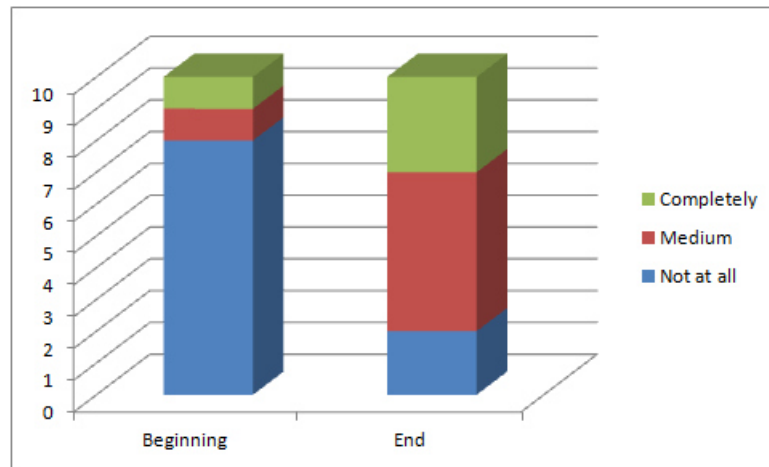


Figure 3.19.: General understanding of answer types

Category		Problem	Affected Users
<i>Missing functionality</i>		No check before deletion of an object	2
		No ordering possibility	3
<i>Unfavourable functionality</i>		Form closes when clicking back or reloading page	2
		Click in space closes form	2
<i>Comprehension problems</i>	<i>Concept Level</i>	Questionnaire structure not understood	5
		Status not understood	2
	<i>Operation Level</i>	Navigation unclear	10
		One question/option per field not clear	4
		Required mark (*) not intuitive	4
	<i>Text Comprehension Level</i>	Terminology not intuitive	10
Input Types not intuitive		8	

Table 3.2.: Usability problems

click and by that close a form unintentionally. Thereby, data that was entered into the form and not saved yet was lost. To prevent this, removing the functionality should be considered. A related problem is the reloading and going back or forward functionality of the browser which can result in the same negative consequences when called unintentionally. Since these functions are not part of the application, they can and should not be disabled easily. Instead a warning message can be implemented, that appears if data was entered and one of these actions is called before saving.

Because of the thinking aloud method not only functionality but also comprehension problems could be evaluated. Although explained in the beginning, many subjects faced problems in understanding the internal structure of the questionnaire as well as the status concept. This includes the construction of questionnaires out of groups, which then contain the actual questions. For the user it means that questions cannot be added to a questionnaire directly and a group should be created first. The status marks the state of an object and defines if it is a draft, in use or retired. Changing the status from 'Draft' or 'Review' to another state will cause the object to become unmodifiable, which implies that the correct sequence of actions is required to successfully create a questionnaire. The experiment showed that a more detailed explanation, training and manual is needed. Examples should be shown so that users can connect these new concepts to their existing knowledge. For example, completed groups and questionnaires in the system can be compared to traditional paper-based forms with different sections. The same should be applied to the status, so that users can gain an overview over the different states and their meaning.

For all participants, navigation was very difficult. The user interface of the form section of eHMIS contains three menus and additional buttons and links. Although naming of these items should be clear, all test persons were confused in finding the places to perform the intended actions. Though this was a problem in the experiment, it will probably not occur after implementation in daily practice. The menu structure and navigation in eHMIS is consistent throughout the whole application. After users have become familiarised with this general layout, navigation will become more apparent and intuitive. The same reasoning applies for required fields, which are consistently marked with a red asterisk. If they are left blank, an error is displayed and the form cannot be saved. It can be expected that even a little orientation will be sufficient for users to adopt these concepts. In addition to that, the way questions and options are created was misunderstood by four subjects, who tried to define several ones in the same input field. In the first place, this can be traced back to little experience with computers and electronic (online) forms. To prevent this mistake, a note or a tooltip could be added to highlight the fact that only one question or rather option is expected. Moreover, the system could check for indicators of that error like multiple questionmarks and '/' or other separators in an option field and display a warning in that case.

Finally text comprehension was a problem for all participants. A lot of questions about fields, their meaning and what to enter were asked. For instance, some of these were: 'Who is the publisher?', 'The version means what?' or 'What is the description about?'. Since all these basic questions about terms can be answered in a short and simple way,



tooltips are suitable to solve this issue. Tooltips can contain a short explanation to help the user to understand the context and meaning of different fields. Besides, less important fields could be hidden and only expanded if necessary. This would decrease the number of shown inputs, making it easier for the user to overlook even a more complex form. Both changes would also affect the distinction of mandatory and optional fields in a positive way. If optional fields are hidden, required ones would come to the fore and tooltips could also state if a field was mandatory or optional. As mentioned before, many subjects misinterpreted the answer types as well, even though a short introduction was given. On the one hand, this means that a more detailed explanation including example questions should be provided. On the other hand, the experiment also indicates that even small practice leads to a much better understanding.

The following list should summarise the ideas for improvement which were discussed above.

- Ordering functionality
- Confirmation message before deletion of objects
- Removal of form closing functionality on click in space
- Warning message for updating or leaving page if unsaved data was entered
- More detailed explanation including examples for questionnaire structure, status concept and answer types
- Automatic check for multiple questions or options in a single input field
- Tooltips to define terms, distinguish between optional and mandatory fields and highlighting that one question/option per field is expected
- Folding forms to hide less important fields

In addition to usability problems the participants' impression of the application and their opinion was of high interest. Three subjects commented negatively on the software. Two of them stated that they will need more training, which corresponds to the existing comprehension problems. One stated "So you just talk to your computer? Where would be the eye contact?". In contrast five test persons expressed a positive attitude and assessed the handling as not so hard. Three subjects created more questions than necessary and two discussed the content of questions which should be created. One commented "It's complicated to think about some things you want to do. But it's not hard anyway. The only hard thing is to think about what you want the questionnaire to be." In the end, two participants asked if the software might be introduced in Tororo Hospital. In summary a predominantly positive impression could be observed. Though an adaptation phase will be necessary.

## 4. Discussion

### 4.1. Discussion of Methods

For the field experiment, the thinking aloud method was applied. Positive and negative aspects of its implementation shall be discussed here. In the end a conclusion will be drawn regarding the appropriateness of thinking aloud to evaluate user interfaces in a rural African environment.

Up to the present, development of eHMIS is not finished and therefore the application has not been deployed in any institution yet. Consequently, there are no experienced users who are familiar with the system and thus able to give a qualified feedback. To be applicable on these conditions is one of the advantages of observational approaches. Moreover, thinking aloud proved to be very practical. The experiment was conducted with only one laptop. Neither an internet connection nor any other sophisticated devices, e.g. eye tracking cameras, were required. This is especially important in an environment like rural Uganda, where infrastructure is still poor and hence preconditions like a stable power supply cannot be relied on. Because of the simple implementation, it was possible to conduct the experiment in Tororo Hospital. Depending on their professional background, age and other factors, user groups differ in the way they approach a task and the problems they face. Thinking aloud enabled the involvement of the target group in the experiment, which was of high importance to obtain authentic results. As stated before thinking aloud belongs to the observational studies. These methods include the direct observation of the subjects while conducting the experiments. Although this procedure does not generate numerous quantitative data, significant qualitative data can be gained. Thereby, the thinking aloud experiment in Tororo yielded much more informative value than for example a questionnaire would have. Beside that, the request to think aloud acted as an amplification factor for the subjects motivating them to talk, which resulted in a more extensive feedback and deeper understanding of underlying motives compared to a common observational experiment.

Nonetheless many people find it hard to verbalize their thoughts. Some subjects faced problems when instructed to think aloud and, even though encouraged to talk, kept quiet. Reasons can only be assumed but might lie in unfamiliarity with the process or the cultural background, because especially younger participants contained themselves. In the conducted experiment an explanation and a video demonstrating thinking aloud were provided to the subjects. At this point a small practice session could be supplemented to improve adherence to the method. The lack of understanding what steps to take for solving the task lead to another problem. Instead of handling the task independently and trying out possible solutions, the moderator was frequently asked for support. By

that, processes of problem-solving were merely observable. Uncertainty might be caused by lacking experience in operating laptops and handling unknown applications as well as anxiety of making mistakes. In the beginning, all necessary steps were named shortly and could also be found on the instructions paper. However, a better way of encouraging subjects and conveying the idea of trial and error has to be found.

In summary, the collected data was of medium quality, which would not have been sufficient for a more sophisticated evaluation like modelling cognitive processes. Nevertheless, the experiment was still a success. The qualitative data that was gained enabled the detection of several usability problems, which will result in an enhanced user interface and higher user satisfaction if handled properly. By that, the method proved to be a practical and effective solution in the given setting. In addition, gained experience can help to improve the experimental procedures and increase data quality of future research.

## 4.2. Discussion of Results

At the beginning of this thesis, a research question was asked and objectives were defined. This section will discuss what was achieved and what tasks are still pending.

First, when the project started, there was the abstract idea of dynamic forms. Various concepts were discussed and requirements analysed. Following modelling started. On the basis of FHIR® specification, a database architecture and business logic were defined. Different challenges had to be met: The structure of questionnaires was slightly modified to match the need of a user interface that is simple to use. A concept for deletion and modification of questions, groups and questionnaires was developed and implemented via the status concept. The declaration of an answer type allowed the storage of responses of different data types, though generic types are neither supported by the database nor by eHMIS itself. Moreover, the FHIR's Questionnaire resource was extended to enable the definition of constraints. All features were integrated successfully in the existing application without conflict with established technologies and frameworks. By passing user tests, the first subgoal was achieved.

In the next step, questionnaires were integrated into the workflow of eHMIS. To do so, the 'Signs and Symptoms' section was chosen which is part of the examination area in encounters. It is the place where a doctor or nurse records the current condition and symptoms of a patient before tests and treatment is performed. The questions asked highly depend on the patient's reason for coming, which makes a dynamic approach reasonable. Beside that, questionnaires are also suited to be integrated into the care plan section. Care plans are used to manage long-term treatments regarding chronic diseases, nutrition programs, pregnancies etc, where each of them comes with its particular forms. Database relations and UI of care plans were modelled. However, implementation was not completed, so that questionnaires could not be fully integrated. Yet, dynamic questionnaires were integrated successfully into encounters, which enables their application in the clinical workflow.

An appropriate design is crucial in application development as the user interface plays

an important part in how users perceive and adopt a system in their daily life. eHMIS, including the section for questionnaire creation and usage, is designed in a uniform way so that no additional familiarisation is needed. The interface enables the user to create questionnaires and define individual questions. For each question, an answer type, options, constraints and other details can be defined without having any programming knowledge. By taking all these settings into consideration when displaying the questionnaire, the system becomes more customisable and satisfies the expectations of the beginning. In addition, a 'regex' field was implemented to allow the definition of advanced constraints as regular expressions. Though a dedicated syntax is required to apply this feature, it serves as supplementary extension mechanism. However, the field experiment also revealed some UI weaknesses, that need to be addressed in order to further improve usability.

The fourth goal states the application of the module in a Ugandan Hospital. Since this new version of eHMIS is not deployed in any location yet, a permanent implementation in a hospital could not be realised so far. Instead, a field experiment was designed and conducted with the objective of getting an impression how users interact with the application. Results show that users were able to operate the system and create questionnaires successfully though a short training session seems to be necessary. Because of the thinking aloud method, a deeper insight of how users perceive the system and what problems they face could be gained. To evaluate the actual usage in routine operation, deployment of eHMIS has to be completed first. This is expected for 2016 and will then give new opportunities for research. So far, results of the experiment were promising and allow a positive forecast.

In the end, the overall intention of dynamic forms is to enable local and situational tailoring without additional costs. Dynamic questionnaires, as they were implemented within this project, enable a hospital or another institution to create, modify and adapt forms by themselves without commissioning a software company. However, fields of application are still limited within the software and saving will depend on how frequently dynamic forms are used in daily routine. At this juncture, the question to what extent code changes decrease and costs are avoided cannot be answered. Future work needs to be done at this point to further investigate the effects and provide more accurate measurements.

In conclusion, all sub-goals goals set in the beginning were achieved. Feasibility of dynamic questionnaires was proven and the field experiment yielded predominantly positive results. Since deployment of eHMIS is still pending, analysis of the application in daily work practices could not be done so far, so that at the end of the project an answer to the research question can be foreshadowed, but not confirmed yet.

## 5. Outlook

Although implementation of dynamic questionnaires was completed, many possibilities for extensions and future research remain.

From the technical perspective, a first step would be to adapt the module to findings of the experiment. Furthermore, extensions of the existing implementation should be considered. Dynamic forms can be inserted in many places. Integration in care plans is already planned and might be realised in the near future. Further possible fields of application need to be evaluated in order to enable a wider use within eHMIS. Beside that, it might become possible to extent static forms by adding dynamic fields. This 'hybrid' form would highly increase flexibility of the software, but might limit data processing. At this point, a balance has to be found. In addition, functionality of the questionnaire itself can be increased. For instance, this might include dependencies between questions or a copying functionality. Dependencies can be realised by offering constraints, that allow statements like 'if the answer to this question is 'no', do not display the following question.' Though many use cases can be thought of, this functionality will also induce a more complex user interface. Copying is a convenience function. If a questionnaire was published but needs to be modified, it has to be replaced by a new one. In this case, copying the old one, performing the modifications and publishing it again would be much easier compared to a manual recreation of every question.

Next to technical improvements, additional research regarding qualitative as well as quantitative aspects is needed. After deployment a second usability experiment should be considered to achieve more realistic results. If subjects are familiar to the general layout and forms, the idea of dynamic questionnaires becomes more concrete and understandable. Moreover, additional requirements regarding the use in daily routine can be captured. In consideration of quantitative measurements, the actual use of dynamic questionnaires needs to be evaluated. In this context the question, who is using dynamic questionnaires, for what reason, in which way and how frequently can be addressed. As stated before the research question, which was designated in the beginning, could not be fully answered yet. After deployment of eHMIS necessary code changes and actual costs of local and situational adaptations have to be quantified in order to draw a final conclusion. Finally, in a long-term perspective the benefit of FHIR® for African healthcare systems will be of high interest, not only for researchers and if FHIR proves to be effective and advantageous, a wider distribution might be brought into focus.

## 6. Bibliography

- [1] D. Tsichritzis, "Form management", *Communications of the ACM*, vol. 25, no. 7, pp. 453-478, 1982.
- [2] M. Zloof, "Office-by-Example: A business language that unifies data and word processing and electronic mail", *IBM Syst. J.*, vol. 21, no. 3, pp. 272-304, 1982.
- [3] R. Jeffries and J. Rosenberg, "Comparing a form-based and a language-based user interface for instructing a mail program", *SIGCHI Bull.*, vol. 18, no. 4, pp. 261-266, 1987.
- [4] A. Girgensohn, B. Zimmermann, A. Lee, B. Burns and M. E. Atwood, "Dynamic forms: an enhanced interaction abstraction based on forms." *Human-Computer Interaction*, pp. 362-367, 1995.
- [5] S. Shao-Zhong and O. Tao, "Implementation of dynamic database functions in B/S structure programming." *Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, 2010 International Conference on, vol. 1, pp. 136-139, 2010.
- [6] S. Ndira, "Assessment of the effect of an EHR system in developing countries (Uganda) on data quality and staff satisfaction, as an indirect contributor to the reduction of maternal mortality-a qualitative and quantitative comparative study." 2010.
- [7] J. Rademacher and S. Lippke, "Dynamic online surveys and experiments with the free open-source software dynQuest", *Behavior Research Methods*, vol. 39, no. 3, pp. 415-426, 2007.
- [8] K. Chen, H. Chen, N. Conway, H. Dolan, J. M. Hellerstein and T. S. Parikh, "Improving data quality with dynamic forms." *Information and Communication Technologies and Development (ICTD)*, 2009 International Conference on, pp. 487-487, 2009.
- [9] Health Level Seven International, "Index - FHIR v0.5.0", Hl7.org, 2015. [Online]. Available: <http://hl7.org/fhir/2015May/index.html>. [Accessed: 13- Dec- 2015].
- [10] Health Level Seven International, "Introducing HL7 FHIR", Hl7.org, 2015. [Online]. Available: <http://hl7.org/fhir/2015May/summary.html>. [Accessed: 13- Dec- 2015].
- [11] Health Level Seven International, "Resource Patient - Content", Hl7.org, 2015. [Online]. Available: <http://hl7.org/fhir/2015May/patient.html>. [Accessed: 13- Dec- 2015].
- [12] M. Someren, Y. Barnard and J. Sandberg, *The think aloud method: a practical approach to modelling cognitive processes*. London: Academic Press, 1994, pp. 8-14.

- [13] K. Duncker, "On problem-solving.", *Psychological Monographs*, vol. 58, no. 5, p. i-113, 1945.
- [14] A. de Groot, *Thought and choice in chess*. The Hague: Mouton, 1965.
- [15] A. Newell and H. Simon, *Human problem solving*. Englewood Cliffs, N.J.: Prentice-Hall, 1972.
- [16] M. Someren, Y. Barnard and J. Sandberg, *The think aloud method: a practical approach to modelling cognitive processes*. London: Academic Press, 1994, pp. 29-31,
- [17] M. Someren, Y. Barnard and J. Sandberg, *The think aloud method: a practical approach to modelling cognitive processes*. London: Academic Press, 1994, pp. 37-39,
- [18] M. Reicks, C. Smith, H. Henry, K. Reimer, J. Atwell and R. Thomas, "Use of the Think Aloud Method to Examine Fruit and Vegetable Purchasing Behaviors among Low-Income African American Women", *Journal of Nutrition Education and Behavior*, vol. 35, no. 3, pp. 154-160, 2003.
- [19] C. Lewis, "Using the "thinking-aloud" method in cognitive interface design." IBM TJ Watson Research Center, 1982
- [20] J. Nielsen, *Evaluating the thinking-aloud technique for use by computer scientists*, *Advances in human-computer interaction* (vol. 3). Ablex Publishing Corp., Norwood, NJ, 1993.
- [21] C. George, *User-centred library websites*. Oxford: Chandos Pub., 2008, pp 154-173.
- [22] J. Nielsen, *Usability engineering*. Elsevier, 1994, pp. 18-19.
- [23] J. Nielsen, *Usability engineering*. Elsevier, 1994, pp. 195-200
- [24] J. Nielsen, *Usability engineering*. Elsevier, 1994, p. 223
- [25] Republic of Uganda, Ministry of Health, "The Health Management Information System" Health Unit Procedure Manual, vol. 1, 2010
- [26] H. Ströhle, T. Wetter and D. Schmidt, "Extending a hospital information system with a clinical decision support system for drug interaction", 2015
- [27] eHMIS Ltd., "eHMIS Presentation", 2012
- [28] G. Matege, "eHMIS Software Architecture (Design Patterns - Revision 1)", 2014
- [29] SionaPros Ltd., "eHMIS components", 2015
- [30] Health Level Seven International, "Resource Questionnaire - Content", Hl7.org, 2015. [Online]. Available: <http://hl7.org/fhir/2015May/questionnaire.html>. [Accessed: 15- Jan- 2015].
- [31] Health Level Seven International, "Resource Questionnaire-Answers - Content", Hl7.org, 2015. [Online]. Available: <http://hl7.org/fhir/2015May/questionnaireanswers.html>. [Accessed: 15- Jan- 2015].
- [32] J. Brooke, "SUS-A quick and dirty usability scale." *Usability evaluation in industry*, vol. 189, no. 194, pp. 4-7, 1996.

# A. Appendix

## A.1. Use Case Descriptions

### Questionnaire Creation

Name	Create a group for questions
Brief Description	A group is created to arrange questions. In the questionnaire they'll serve as sections.
Actors	Admin
Preconditions	<ul style="list-style-type: none"><li>• Admin is logged in.</li><li>• The user has navigated to the sub menu Forms.</li></ul>
Basic Flow	<ul style="list-style-type: none"><li>• The user navigates to draft groups.</li><li>• The user clicks on "Add".</li><li>• The user enters a title and an optional description.</li><li>• The user clicks the "Save" button on the form.</li></ul>
Alternate Flow	-
Exception Flows	<ul style="list-style-type: none"><li>• The user doesn't enter a title.</li><li>• The user clicks the "Save" button on the form.</li><li>• An error message is displayed</li></ul>
Post Conditions	A success message is shown. A group is created and its details are displayed.



Name	Create a questionnaire
Brief Description	A questionnaire is created. After assigning groups with questions to it, it can be published and used during an encounter of a patient.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• The user has navigated to the sub menu Forms.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to draft questionnaire.</li> <li>• The user clicks on “Add”.</li> <li>• The user enters a title.</li> <li>• Optionally the user can specify a publisher, a version and a description.</li> <li>• The user clicks the “Save” button on the form.</li> </ul>
Alternate Flow	-
Exception Flows	<ul style="list-style-type: none"> <li>• The user doesn’t enter a title.</li> <li>• The user clicks the “Save” button on the form.</li> <li>• An error message is displayed</li> </ul>
Post Conditions	A success message is shown. A questionnaire is created and its details are displayed.

Name	Create a question
Brief Description	A question is created. It's part of a specified group and will be displayed as a part of it.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• The user has navigated to the sub menu Forms.</li> <li>• The group is still in draft state.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the draft group to which he wants to add the question.</li> <li>• The user opens the question section.</li> <li>• The user clicks on "Add".</li> <li>• Optionally the user specifies a sequence number to order questions.</li> <li>• The user enters a question text and an answer format.</li> <li>• Optionally the user specifies the input type, a required flag and constraints according to the input type (e.g. maximum value, specific text format).</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The answer format chosen is single or multiple choice.</li> <li>• The user specifies any amount of options options.</li> <li>• The user clicks on "Save".</li> </ul>
Exception Flows	-
Post Conditions	A success message is shown. A question is created and displayed in the question section of the group, where it belongs to.

Name	Modify object
Brief Description	Properties of a questionnaire, a group or a questions are modified.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• The user has navigated to the sub menu Forms.</li> <li>• A questionnaire, group or question was created.</li> <li>• The questionnaire or group (to which the question belongs) is still in draft state.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the object in the appropriate draft section.</li> <li>• The user clicks the modify icon.</li> <li>• The user edits the properties in the displayed form.</li> <li>• The user clicks on “Save”.</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The user wants to edit a question and navigates to the corresponding group.</li> <li>• The user clicks on “Edit” next to the question and continues as described as basic flow.</li> </ul>
	<ul style="list-style-type: none"> <li>• The user cancels the task.</li> <li>• No changes are safed, the object remains as before.</li> </ul>
Exception Flows	<ul style="list-style-type: none"> <li>• The user delets a required field.</li> <li>• The user clicks on “Save”.</li> <li>• An error message is displayed. Changes are not saved.</li> </ul>
Post Condtions	A success message is shown. The object was modified. The updated properties are displayed.

Name	Delete object
Brief Description	A questionnaire, group or question is deleted.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• The user has navigated to the sub menu Forms.</li> <li>• A questionnaire, group or question was created.</li> <li>• The questionnaire or group (to which the question belongs) is still in draft state.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the object in the appropriate draft section.</li> <li>• The user clicks the delete icon.</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The user wants to delete a question and navigates to the corresponding group.</li> <li>• The user clicks on “Delete” next to the question.</li> </ul>
Exception Flows	<ul style="list-style-type: none"> <li>• The object cannot be deleted because of existing references to other objects.</li> <li>• An error message is displayed. The object remains in the system.</li> </ul>
Post Conditions	A success message is shown. The object was deleted and is not displayed anymore. A success message is shown.

Name	Assign groups to questionnaires.
Brief Description	Groups of questions can be assigned to various questionnaires. A questionnaire consists of all questions of every assigned group. Groups are treated as sections.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• The user has navigated to the sub menu Forms.</li> <li>• A questionnaire and group was created.</li> <li>• The questionnaire is still in draft state.</li> <li>• The group's status was changed to "Active".</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the questionnaire and opens the group section.</li> <li>• The user clicks on "Edit".</li> <li>• The user checks the boxes of groups he wants to have in the questionnaire. It is also possible to remove groups from the questionnaire.</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	-
Exception Flows	-
Post Conditions	A success message is displayed. The chosen groups are assigned to the questionnaire. The questions within the groups are now part of the questionnaire.

Name	Change status
Brief Description	Groups and questionnaires have a status. Before a group can be assigned and a questionnaire can be used, it has to be marked as "Active" or rather "Published". "Disabled" or "Retired" objects cannot be assigned or used anymore.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• User has navigated to the sub menu Forms.</li> <li>• A questionnaire or a group was created..</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the group or questionnaire.</li> <li>• The user clicks on "Change Status".</li> <li>• The user chooses a status.</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	-
Exception Flows	-
Post Conditions	The status was changed and a success message is displayed. If the questionnaire or group was marked as complete, it is not modifiable any more. A published questionnaire can be used to record answers of a patient.

Name	Show preview
Brief Description	A preview of a group or a questionnaire is shown, displaying all questions as they will look like in the questionnaire.
Actors	Admin
Preconditions	<ul style="list-style-type: none"> <li>• Admin is logged in.</li> <li>• User has navigated to the sub menu Forms.</li> <li>• A questionnaire or group was created.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the group or questionnaire.</li> <li>• The user clicks on “Show Preview”</li> </ul>
Alternate Flow	-
Exception Flows	-
Post Conditions	A preview is displayed.

## Questionnaire Usage

Name	Record answers to a questionnaire
Brief Description	A health worker wants to record the answers of a patient to questions of a specific questionnaire.
Actors	Authenticated user (physician, nurse, health worker...)
Preconditions	<ul style="list-style-type: none"> <li>• A patient exists and an encounter is in progress.</li> <li>• A questionnaire was created and published.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the encounter or care plan section of the patient's health record and opens the questionnaire tab.</li> <li>• The user clicks on "Add".</li> <li>• The user chooses a questionnaire and clicks on "Save".</li> <li>• The user starts recording the answers of the patient.</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The user cancels before choosing a questionnaire.</li> <li>• Nothing is saved.</li> </ul>
	<ul style="list-style-type: none"> <li>• The user cancels the recording of answers.</li> <li>• No answers are saved, but the empty questionnaire is still displayed in the health record of the patient.</li> </ul>
Exception Flows	<ul style="list-style-type: none"> <li>• A question that was marked as required was not answered. The user clicks on "Save".</li> <li>• An error message is displayed. The form remains open. The existing answers are not saved.</li> </ul>
Post Conditions	A success message is displayed. A feedback of the patient for the chosen questionnaire was created. Answers were recorded and saved.



Name	Modify answers
Brief Description	If an interview isn't finished or answers have to be modified the user can alter or complement them later on.
Actors	Authenticated user (physician, nurse, health worker...)
Preconditions	<ul style="list-style-type: none"> <li>• A patient exists and an encounter is in progress.</li> <li>• A questionnaire was created and published.</li> <li>• A patient's health record was associated with a questionnaire.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the encounter or care plan which the questionnaire was associated with and opens the questionnaire tab.</li> <li>• The user clicks on "More details" of the questionnaire to display the given answers.</li> <li>• The user clicks on "Edit".</li> <li>• The user modifies or complement the answers.</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The status of the answers is "Completed". Still the user them and clicks on "Save".</li> <li>• The changes are saved and a success message is displayed. The status is changed to "Amended".</li> </ul>
	<ul style="list-style-type: none"> <li>• The user cancels modifying the answers. No changes are made.</li> </ul>
Exception Flows	<ul style="list-style-type: none"> <li>• An answer of a required question was deleted. The user clicks on "Save".</li> <li>• An error message is displayed. The form remains open. Changes are not saved.</li> </ul>
Post Conditions	A success message is displayed. The altered or complemented answers are saved and displayed.

Name	Clear questionnaire
Brief Description	The association between a patient's record and a questionnaire is deleted. That happens if the patient doesn't need to answer the questionnaire or if all answers to a questionnaire should be deleted.
Actors	Authenticated user (physician, nurse, health worker...)
Preconditions	<ul style="list-style-type: none"> <li>• A patient exists and an encounter is in progress.</li> <li>• A questionnaire was created and published.</li> <li>• A patient's health record was associated with a questionnaire.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the encounter or care plan which the questionnaire was associated with and opens the questionnaire tab.</li> <li>• The user clicks on the delete icon.</li> </ul>
Alternate Flow	-
Exception Flows	<ul style="list-style-type: none"> <li>• The object cannot be deleted because of existing references to other objects.</li> <li>• An error message is displayed. The object remains in the system.</li> </ul>
Post Conditions	A success message is shown. The association between questionnaire and the patient's record is deleted, though the questionnaire itself still exists. All answers given by the patient are deleted.

Name	Conclude answers
Brief Description	After all questions have been answered or questions has been skipped on purpose and the interview with the patient is finished the user changes the status of the answers to complete.
Actors	Authenticated user (physician, nurse, health worker...)
Preconditions	<ul style="list-style-type: none"> <li>• A patient exists and an encounter is in progress.</li> <li>• A questionnaire was created and published.</li> <li>• A patient's health record was associated with a questionnaire.</li> <li>• The status of answering the questionnaire is "In progress"</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the encounter or care plan which the questionnaire was associated with and opens the questionnaire tab.</li> <li>• The user clicks on "Change Status".</li> <li>• The user changes the status to "Completed".</li> <li>• The user clicks on "Save".</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The user cancels changing the status. The status remains "In progress".</li> </ul>
Exception Flows	-
Post Conditions	A success message is shown. The status of the answers is "Completed". If answers are changed again the status will change automatically to "Amended".

Name	Review answers
Brief Description	The user can access the answers of a patient from his health record.
Actors	Authenticated user (physician, nurse, health worker...)
Preconditions	<ul style="list-style-type: none"> <li>• A patient exists and an encounter is in progress.</li> <li>• A questionnaire was created and published.</li> <li>• A patient's health record was associated with a questionnaire.</li> </ul>
Basic Flow	<ul style="list-style-type: none"> <li>• The user navigates to the encounter or care plan which the questionnaire was associated with and opens the questionnaire tab.</li> <li>• The user clicks on "More details" of the questionnaire to display the given answers.</li> </ul>
Alternate Flow	<ul style="list-style-type: none"> <li>• The patient has not answered any questions yet. Only the questions will be displayed.</li> </ul>
Exception Flows	-
Post Conditions	The questions of the questionnaire and the answers of the patient are displayed.

## A.2. Logic Layer Code Example

```

public QuestionnaireQuestion saveQuestionnaireQuestion(int id, String text,
    QuestionnaireAnswerFormat type, int sequenceNumber, Integer groupId,
    List<QuestionnaireQuestionOption> options, int validationId,
    QuestionnaireAnswerInputFormat validationInputFormat,
    QuestionnaireAnswerInputTextFormat validationTextFormat,
    QuestionnaireAnswerInputDateType validationDateType,
    boolean validationRequired, Long validationMinimumValue,
    Long validationMaximumValue, String validationRegularExpression) {

    QuestionnaireQuestion questionnaire_question = null;
    if (groupId != null) {
        if (sequenceNumber == 0) {

```

```

List<QuestionnaireQuestion> groupQuestions = this.getBusiness().
    getDatabase().loadDataRepository(ClinicalRepository.class).
    getQuestionnaireGroupQuestions(groupId);

    if (groupQuestions != null && !groupQuestions.isEmpty()) {
        sequenceNumber = groupQuestions.get(groupQuestions.size() - 1).
            getSequenceNumber() + 1;
    } else {
        sequenceNumber = 1;
    }
}
questionnaire_question = this.getDataFactory().
    setQuestionnaireQuestion(id, text, type, sequenceNumber);

if (validationRegularExpression != null &&
    validationRegularExpression.isEmpty()) {
    validationRegularExpression = null;
}

QuestionnaireQuestionValidationAttributeSet validationSet =
    this.getDataFactory().setQuestionnaireQuestionValidationAttributeSet(
        validationId, validationInputFormat, validationTextFormat,
        validationDateType, validationRequired, validationMinimumValue,
        validationMaximumValue, validationRegularExpression);

if(questionnaire_question != null && validationSet != null) {
    questionnaire_question.setValidationSet(validationSet);

    QuestionnaireGroup group = this.getBusiness().getDatabase().select(
        QuestionnaireGroup.class, groupId);

    if (group != null && (group.getStatus() ==
        QuestionnaireGroupStatus.DRAFT || group.getStatus() ==
        QuestionnaireGroupStatus.REVIEW)) {
        questionnaire_question.setGroup(group);
    } else {
        return null;
    }
}

List<QuestionnaireQuestionOption> oldOptions =
    questionnaire_question.getOptions();
List<QuestionnaireQuestionOption> notDeletedOptions = new ArrayList<>();

for (QuestionnaireQuestionOption newTempOption : options) {

```

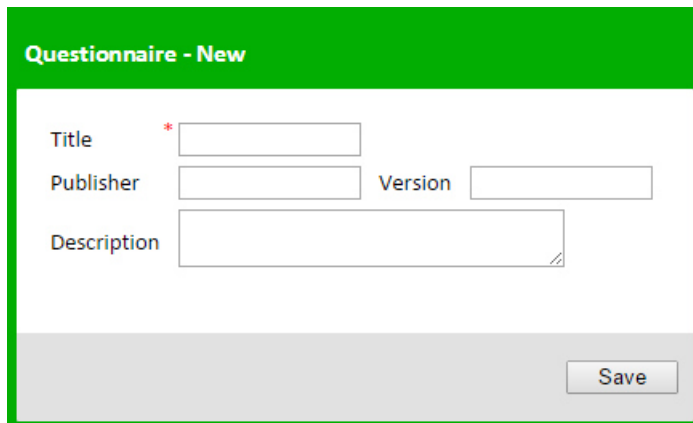
```

        boolean found = false;
        if (oldOptions != null) {
            for (QuestionnaireQuestionOption oldOption : oldOptions) {
                if (newTempOption.getId() == oldOption.getId()) {
                    found = true;
                    oldOption.setText(newTempOption.getText());
                    oldOption.setQuestion(questionnaire_question);
                    notDeletedOptions.add(oldOption);
                }
            }
        }
        if (!found) {
            QuestionnaireQuestionOption newOption =
                this.getDataFactory().setQuestionnaireQuestionOption(
                    newTempOption.getId(), newTempOption.getText());
            newOption.setQuestion(questionnaire_question);
        }
    }

    if (oldOptions != null) {
        Iterator<QuestionnaireQuestionOption> it = oldOptions.iterator();
        while (it.hasNext()) {
            QuestionnaireQuestionOption oldOption = it.next();
            if (!notDeletedOptions.contains(oldOption)) {
                this.getBusiness().getDatabase().delete
                    (oldOption, true, false);
                it.remove();
            }
        }
    }
    this.getBusiness().getDatabase().save();
}
return questionnaire_question;
}

```

### A.3. Presentation Layer View Example



```
<div data-layout-decorator="\${ui.viewPath('formDialog')}"
    data-layout-fragment="inputs">
  <table class="cellspacing_min cellpadding_smaller">
    <tr>
      <td>
        <table class="cellspacing_min cellpadding_smaller">
          <tr>
            <td>
              <label for="title" data-th-text="\#{attribute.title}"&nbsp;</label>
            </td>
            <td colspan="3">
              <input type="text" data-th-field="*\{title}"/>
            </td>
          </tr>
          <tr>
            <td>
              <label for="publisher" data-th-text="\#{attribute.publisher}">
                &nbsp;   </label>
            </td>
            <td>
              <input type="text" data-th-field="*\{publisher}"/>
            </td>
            <td>
              <label for="title" data-th-text="\#{attribute.version}">
                &nbsp;</label>
            </td>
            <td>
              <input type="text" data-th-field="*\{version}"/>
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  <input type="button" value="Save" data-th-field="save"/>
</div>
```

```
</tr>
<tr>
  <td>
    <label for="text" data-th-text="#{attribute.description}">
      &nbsp;&nbsp;&nbsp;
    </label>
  </td>
  <td colspan="3">
    <textarea data-th-field="*{text}">&nbsp;&nbsp;&nbsp;</textarea>
  </td>
</tr>
</table>
</td>
</tr>
<tr>
  <td>
    <input type="hidden" data-th-field="*{id}"/>
    <input type="hidden" data-th-field="*{date}"/>
    <input type="hidden" data-th-field="*{status}"/>
  </td>
</tr>
</table>
</div>
```



## A.4. User Interface Screenshots

English | German | French Sign Out



**eHMIS** HOME PATIENTS ANCILLARIES FINANCES ADMIN

Forms

Groups Questionnaires

Draft Review Active Disabled Draft Published Retired

Questionnaires - Draft | [Add](#)

**Test**   


Publisher: Britta Lohmann | [View More](#) | [Show Preview](#)

Status: Draft [Change Status](#)

[Groups](#) [Other Information](#)

**Groups** [Edit](#)

Title	Status	Description
General Questions	Active	This group contains questions regarding the general condition of the patient.

© eHMIS - electronic Hospital Management Information System

English | German | French Sign Out



**eHMIS** HOME PATIENTS ANCILLARIES FINANCES ADMIN

Forms

Groups Questionnaires

Draft Review Active Disabled Draft Published Retired

Questionnaires - Draft | [Add](#)

**Test**   


Publisher: Britta Lohmann | [View More](#) | [Show Preview](#)

Status: Draft [Change Status](#)

[Groups](#) [Other Information](#)

**Other Information**

[Contacts](#) [Add](#)

Use	System	Value	Period	Manage
Work	Email	example@provider.de	View	<a href="#">Edit</a>   <a href="#">Delete</a>

[Identifiers](#)

© eHMIS - electronic Hospital Management Information System

**Group - New**

Title \*

Description

Save

**Group Status** Close

Status \*

- Review
- Active

Save

## A.5. Documents of the Experiment

## eHMIS Questionnaire *Template*

- 1 Please enter the first question into the application.
- 2 Please enter another four questions. You might take the ones provided below or create your own.

What is your **gender**?

Male / Female

How old are you?

\_\_\_\_\_ (age in years)

How is your **health**?

Very Poor / Poor / Neither Poor nor Good / Good / Very Good

Do you consider yourself currently as ill? Yes /No

If there is something wrong with you, what do you think it is? \_\_\_\_\_



## eHMIS Questionnaire

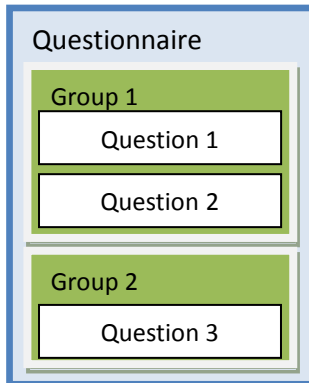
### *Demographics*

1. What is your gender? Male / Female
2. What is your profession? \_\_\_\_\_
3. Since how many years do you work in this profession? \_\_\_\_\_
4. How much computer experience do you have? None / Little / Medium / Much
5. How many hours do you spend on a computer per week?  
(private and professional) < 1 / 1-2 / 3-6 / 6 >

# eHMIS Questionnaire

## *Short Instructions*

### 1 Structure



*A questionnaire consists of various groups, which contain the actual questions.*

### 2 Line of Action

➤ **Create a group**

The questionnaire section can be found under ADMIN → FORMS.

Fields that are marked with a \* are required, all others are optional and can be skipped.

➤ **Add questions to the group**

For every question an answer format has to be specified. An explanation of the different options can be found below:

Answer Format	
Input	Answer is a text, a number or a date (e.g. name, birthdates).
Single Choice	One answer is picked between different options.
Single Open Choice	One answer is picked between different options. An additional comment can be recorded.
Multiple Choice	Multiple answers can be picked between different options.
Multiple Open Choice	Multiple answers can be picked between different options. An additional comment can be recorded.

➤ **Finalize group**

Before a group can be added to a questionnaire, it has to be finalized. This is done by changing its status to "Active". After doing so neither the group nor the questions can be modified anymore.

➤ **Create a questionnaire**

➤ **Add groups to the questionnaire**

➤ **Finalize questionnaire**

As soon as the status of a questionnaire is set to "Published", it can be used.

# eHMIS Questionnaire

## *Feedback*

### System Usability Scale

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	

## A.6. Analysis of the Experiment

Subject	Gender	Profession	Years in this Profession	Computer Experience	Hours spent on the computer
1	Female	Nurse	15	Medium	6 <
2	Female	Medical Social Worker	10	Medium	6 <
3	Female	Anaesthetic Attendant	1	Medium	3-6
4	Female	Nursing Officer	30	Little	< 1
5	Male	Record Assistant	3	Medium	6 <
6	Female	Medical Records Assistant	3	Medium	3 - 6
7	Female	Data Assistant	0,33	Little	1 - 2
8	Female	Secretary	8	Medium	3 - 6
9	Female	Accountant	4	Medium	1 - 2
10	Female	Midwife	17	Medium	1 - 2

Subject	1	2	3	4	5	6	7	8	9	10
Total duration (min)	33:41	30:15	48:13	68:37	30:04	30:55	16:35	31:55	30:50	47:20
Number of example questions taken	0	0	0	1	1	1	0	0	1	0
Total number of questions	6	5	7	5	6	5	5	5	5	5
Time needed for 1st question	2:00	4:30	1:00	11:00	1:35	5:55	0:20	1:55	6:07	7:38
Time needed for 5th question	0:35	1:20	2:20	3:10	0:37	1:18	0:20	1:37	0:46	4:08
Intuitive comprehension of answer type of answer types <sup>1</sup>	0	0	2	0	0	0	0	0	1	0
Comprehension of answer types in the end <sup>1</sup>	2	1	2	1	0	1	0	1	2	1
Answer types used	3	2	5	2	3	3	2	3	3	3
Constraints used	1	0	1	0	1	1	0	0	0	1
Preview used independently	0	0	0	0	1	0	0	0	0	0

---

<sup>1</sup>0 = not at all, 1 = medium, 2 = completely