MedForm

Utilizing the Web for Dynamic Clinical Trials

David Breneman

2.1

2004

Abstract

As scientific research advances so does the number of medical surveys that are conducted. Presently most such surveys, so called clinical trials, are conducted using paper-based questionnaires. It is expensive and time consuming to distribute, administrate and then compile the results of such surveys.

This report documents the work of a project which aims to investigate the possibility of using a web-based system for creation, administration and distribution of surveys for clinical trials, and also for collecting data from the users of the system and compiling the results in a form that is directly usable in further research using tools for statistics and analysis.

The project has been executed within MedView at the Department of Computing Science at Chalmers University of Technology, in cooperation with the Department of Endodontology and Oral Diagnosis at Göteborg University.

It has been proven possible to implement a flexible and user-friendly system that can be successfully operated by anyone with a moderate level of computer experience and an Internet-connected computer that has one of the standard web browsers installed. The system also has several other advantages over the paper form equivalent since the interface can be adjusted depending on who is using the system. Questions can change depending on previous user input or even be invisible until they become relevant to the user.

It is still too early to know if the system will gain widespread acceptance for use in the daily work in the health care industry, but the response so far has been positive.

Sammanfattning

I takt med att forskningen går framåt ökar även antalet medicinska undersökningar. I dagsläget sker sådana undersökningar, så kallade kliniska prövningar, nästan uteslutande med hjälp av pappersformulär. Det är både dyrt och tidskrävande att distribuera, administrera och sedan sammanställa resultatet av sådana undersökningar.

Den här rapporten beskriver arbetet i ett projekt vars mål är att undersöka möjligheten att använda ett web-baserat system för att skapa, administrera och distribuera formulär för kliniska prövningar, för att sedan samla in och sammanställa data från användare av systemet på en form som sedan direkt kan användas för forskning med hjälp av statistik- och analysverktyg.

Projektet har genomförts inom ramen för MedView på institutionen för datavetenskap på Chalmers tekniska högskola, i samarbete med avdelningen för endodonti med oral diagnostik på Göteborgs universitet.

Det har visat sig att ett flexibelt och användarvänligt system, som inte kräver mer än någorlunda god datorvana hos användaren och en dator med Internet-anslutning och en vanlig web-läsare, är möjlig att både implementera och använda. Det har dessutom flera andra fördelar över pappersbaserade undersökningar som till exempel att det går att variera utseendet beroende på vem som använder det, frågor kan förändras baserat på tidigare indata eller vara osynliga tills de blir aktuella för användaren.

Huruvida verktyget kommer att accepteras för användning i det dagliga arbetet inom sjukvården återstår att se, men det preliminära omdömet verkar vara mycket positivt.

Preface

This report documents my Master's Thesis in Computer Science and Engineering. It was carried out during the summer and fall of 2004.

I would like to thank my supervisor Olof Torgersson for all the help and advice I have received during the course of this project. I would also like to thank Mats Jontell for the constructive criticism he has provided.

- *David Breneman*

Table of Contents

1. Introduction

MedView[1] is a joint project initiated in 1995 by members of the Department of Computing Science at Chalmers University of Technology and the Department of Endodontology and Oral Diagnosis at Göteborg University. The MedForm project has been executed as a part of MedView.

The thesis work has been supervised by Olof Torgersson, Assistant Professor at the Department of Computing Science at Chalmers University of Technology. Olof is one of the driving forces on the MedView team who has conducted research concerning formal handling of patient information[1][2][3][4][5].

Mats Jontell, Professor and Senior Dental Officer at the Department of Endodontology and Oral Diagnosis at Göteborg University is also a member of the MedView team. He has provided feedback, ideas and suggestions during the design, development and user testing phases of the project.

The overall aim of the MedView project is to develop models, methods and tools to support clinicians in their diagnostic work. The project portfolio includes applications for collecting and visualizing patient data.

At this point in time, there is not an effective and easy to use way to collect information, irregardless of available hardware and software, location or computer savviness. The method currently used involves using a program that must be installed on the clinician's computer.

Whenever data is to be gathered from patients or untrained medical staff members standard paper-based forms are used. Once the forms have been submitted, someone must convert the information to digital form by hand before it can be processed.

That is where this project comes into play. The goal is to create a web-based system which will allow users to easily create and manage questionnaires for data collection. Both aspects, creating questionnaires and entering data into them, will only require access to the Internet (or a local server) and a modern web browser, such as Mozilla, Safari or Microsoft Internet Explorer.

Hopefully the system will, besides saving time and money, enhance the experience of creating and performing clinical trials.

2. MedView

The MedView project was initiated in order to support evidence-based oral medicine. The fundamental goal in the MedView project is to provide a formalization of clinical examination data and clinical procedures. A formal definition of the data structure allows collected patient information to be viewed, analyzed or processed automatically.

MedView focuses on the question: How can computer technology be used to manage clinical knowledge in everyday work such that clinicians more systematically can learn from the gathered clinical data?[4] A number of different tools are provided allowing knowledge acquisition, knowledge generation, visualization and analysis of data, as well as knowledge sharing.

The formal declarative model constitutes the main governing principle in MedView, not only in the formalization of clinical terms and concepts, but in visualization models and in the design and implementation of individual tools and the system as a whole as well.

2.1. Structure

The atomic data unit in MedView is an *examination*. Each examination is a set of *terms*. A term is defined as either a set of other terms or a set of values. Table 1 gives an example examination.

Table 1. Sample Examination

Term		Definition
Examination	=	{Patient-data, Anamnesis, Diagnosis, ...}
Patient-data	=	{Patient-code, Age, Born, ...}
Anamnesis	=	{Medication, Allergies, Smoke, Alcohol, ...}
Patient-code	=	{"1234567890"}
Age	=	{"43"}
Born	=	{"Sweden"}

2.2. Images

Each examination can also contain an arbitrary number of images. Images are referenced using a term such as *Photos* defined as a set of paths to the image files.

2.3. Data Storage

Examinations are stored in a format designed specifically for the MedView project. Detailed information about the format can be obtained from the MedView Data Storage Standard Description[6], and a sample output file from Appendix A.

2.4. Form Descriptions

The structure of the examinations used to collect data is stored using an XML standard. The possible types of values, and actual possible values, for each term are defined by two files, termDefinitions and termValues respectively.

A sample input for a form question could be:

```
<INPUT>
  <TERM>Born</TERM>
  <TYPE>single</TYPE>
  <DESCRIPTION>...</DESCRIPTION>
```

```
</INPUT>
```

A corresponding (in this case redundant) entry in termDefintions would be:

```
$Born single
```

And the term's entry in termValues would be:

```
$Born
Europe
Asia
North America
South America
Africa
...
```

The apparent redundancy concerning information about stored terms is due to the fact that the applications in the MedView portfolio do not all use the same standard to store term description information.

2.5. Applications

The MedView system consists of a number of different Java-programs for gathering, displaying and visualizing information. Many of them are used daily by professionals at the Department of Endontology and Oral Diagnosis where a large database of patient information has been gathered.

2.5.1. MedRecords

MedRecords allows a user to associate values, such as a patient's personal data, health information, diagnosis and so on with corresponding terms to create an examination.

The form-like interface allows the user to choose values from a predefined list or to enter new variable values as text, making data input quick, easy and reliable. The user may also include digital images in an examination. The collected data is stored in tree files in a predefined file structure.

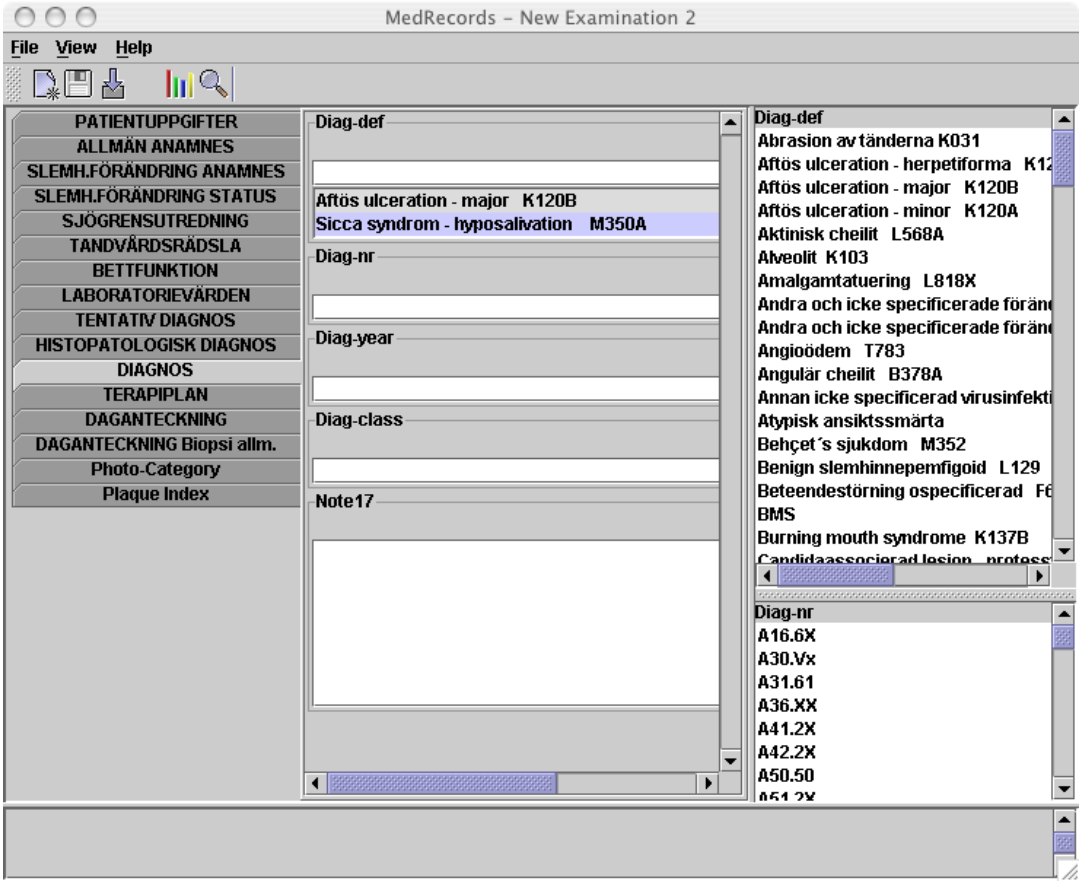


Figure 1. MedRecords in action.

2.5.2. FormEditor

The form structure for each examination type used by MedRecords is defined by the XML format described above. Such XML files can be created and edited with FormEditor.

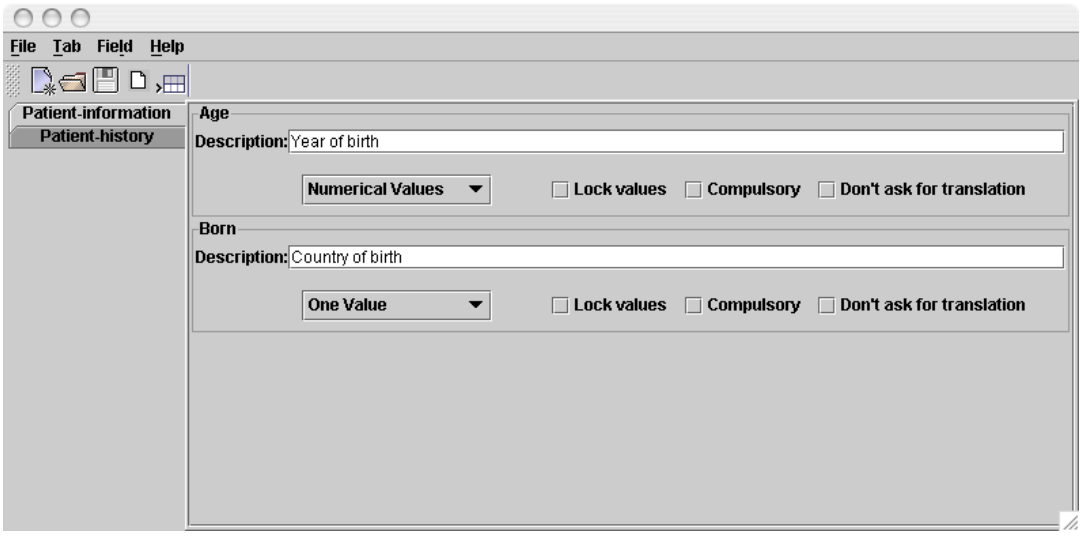


Figure 2. FormEditor in action.

2.5.3. MedSummary

MedSummary uses the output from programs such as MedRecords to present the information, including any associated images, in one of several different human-readable forms.

For example, the data can be displayed as medical records intended to be read by the patient. An alternative representation of the same data may be more appropriate for viewing by other doctors. Such a representation can be easily achieved by simply choosing a different translator.

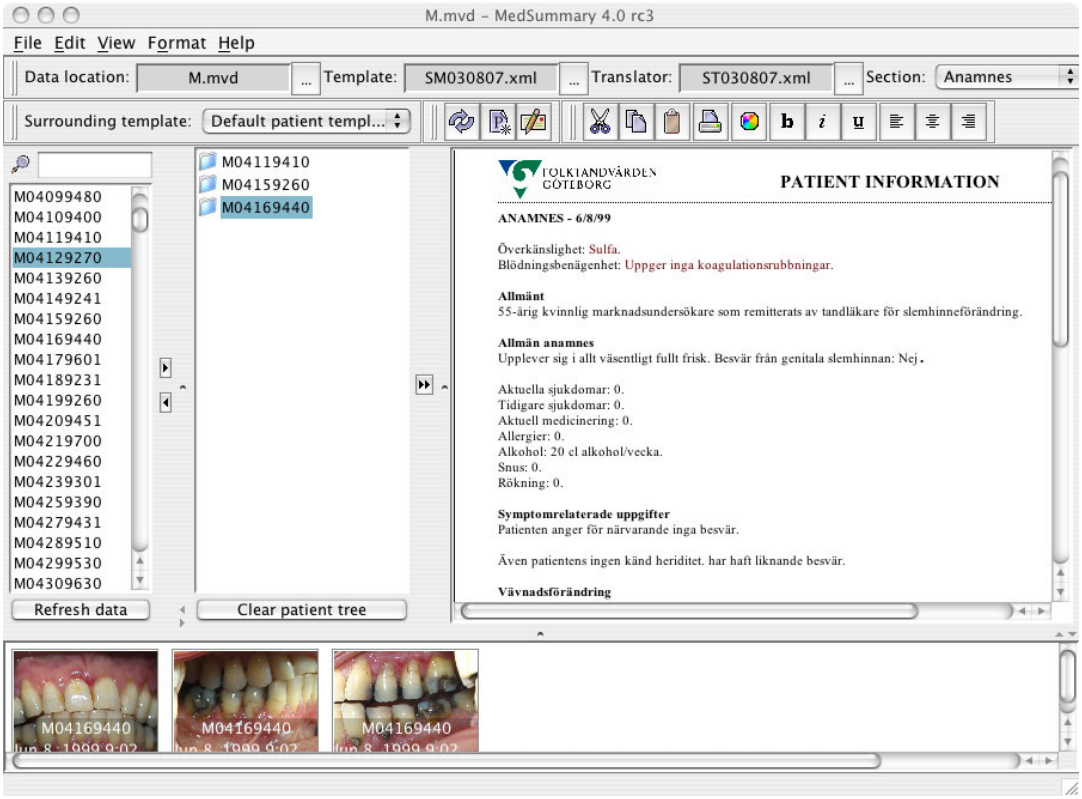


Figure 3. MedSummary in action.

2.5.4. MVisualizer

MVizualizer can be used to analyze a large number of examinations stored in the MedView data storage format visually in an intuitive drag-and-drop user interface.

Different kinds of graphical presentations, such as diagrams and scatterplots, can be used to represent data, allowing the user to visually observe trends or connections that may otherwise be difficult to discover.

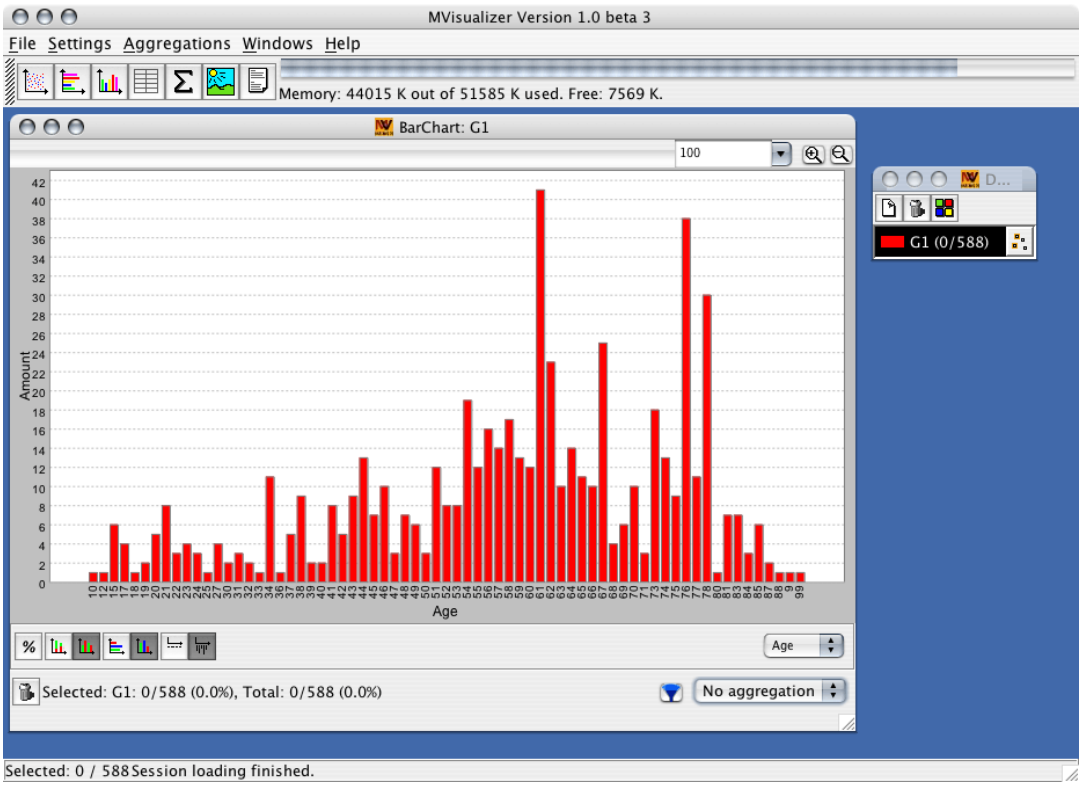


Figure 4. MVisualizer in action.

3. The MedForm Objective

The goal of this project is to develop a system allowing medical personnel, and possibly even the patients themselves, to enter patient data into the MedView database, using standard web-pages. This is important since many users are intimidated by being required to install and learn how to use new applications, but are becoming increasingly accustomed to running bank errands, checking time tables and so on, on the web.

An intuitive, form-like interface, based on a standard scripting language such as HTML, must be created, since requiring special plug-ins or browsers would defeat the purpose of the project.

The form interface must let the user input data as quickly and accurately as possible, without handicapping the number of types of questions that may be asked. It must also be obvious to the user how each type of question is to be answered.

Possible enhancements that would supply the user with added value over that provided by current paper-based forms must be explored and tested. If the tests are successful, the new features should be included for integration into the system.

A suitable form description standard, possibly much like the XML standard used by MedRecords, must be developed. An editor allowing the form administrator to create and edit forms online must be implemented.

Data collected by the system must be made available to other components of the MedView system, such as MVisualizer or MedSummary.

In every way possible, the web-based system must integrate well with other applications in the MedView portfolio.

4. Preparations

4.1. Limitations

4.1.1. Time

The time span for the MedForm project is from March to October of 2004. The bulk of the development work will be done during the period May-September. This constraint will of course limit the number of features that can be implemented and tested.

4.1.2. Scope

The goal of this project is to create a usable environment allowing medical personnel, and possibly even patients to enter data for later use. The forms must also be created, so a simple web-based editor will be created as well. However, the editor itself is outside the scope of this project.

4.1.3. Resources

The extent of the project will be limited by the available hardware and software. A development machine running MacOS X and any freely available software will be used, as well as a web server running a default installation of Debian Linux, that has been provided by the Department of Oral Medicine.

The server is shared with other projects, so some constraints will be in effect concerning what can be done with it. For example, some software may not be entirely up to date since other projects may only run in a stable manner with an older version.

4.1.4. Security

Since handling of medical data is of an inherently sensitive nature a discussion concerning data security and data integrity is in place.

Though security-related issues within the MedForm system will be addressed as they arise it is assumed that the responsibility to uphold functional firewalls, controlled system access, secure data transfer, data backup and so on lies chiefly on the administrator of the system on which MedForm is installed.

4.1.5. User Testing

Since most of the development will be done during the summer, when most of the concerned medical professionals will be on vacation, the test group available for evaluation of the application will not be excessive.

4.2. Target Audience

The system is aimed at medical personnel with a certain degree of training and experience using the system. However, the system will be self-explanatory for anyone with a sufficient level of computer literacy.

4.3. Requirements

The following requirements must be met by the final product:

- The system must be independent of computer hardware and software platform, and must work with any reasonably new and reasonably standards-compliant Web-browser such as Mozilla, Apple Safari or Microsoft Internet Explorer.

- The client-side portion of the system must be implemented using standard HTML, CSS and JavaScript only. No installation or configuration should be required of the user, other than a standard Web-browser and server access via a standard Internet connection.
- The system must, to the extent it is possible, use the data formats provided by the MedView-project for specification of a form and also for the information resulting from a form being filled.
- The system must allow an arbitrary number of content creators.
- Each content creator must be able to manage an arbitrary number of questionnaires.
- Each questionnaire must be able to have an arbitrary number of users.
- The application must be usable by any somewhat computer-literate user with little or no documentation or training.
- The system must, to the extent it is possible, utilize techniques that provide form users with a superior user experience compared to that of the paper form equivalent.
- The application must conform to good human-computer interaction practices.

5. Method

5.1. Design

A user-centered design process would have been preferred. However, since the group of actual users is very small, and the amount of time they can spare for application testing is even smaller, an alternative method will have to suffice. The design process consist of short intervals, which will last between one and three weeks.

Each interval will focus on a specific problem which will be solved, implemented and then tested for stability, usability and consistency. If the results are deemed satisfactory, the next problem or area of concern will be tackled. If the results are unacceptable, that particular interval will require one or more additional iterations.

5.2. Development

There are no restrictions concerning the software and development models to be used, as long as they do not contradict the requirements of the project. However, when possible, standards, technology and software that is freely available will be prioritized over alternatives that are less free.

Also, standards, languages and protocols that are in use within the MedView project will be prioritized over other alternatives if there is not a very good reason to choose the alternative.

5.3. User Testing

Since a user-based iteratory method of design and development cannot be used, a somewhat handicapped version of it will be utilized.

Each iteration will be tested on a small number of individuals that are not involved with the MedView project in any way. This is not necessarily unfavorable to the project, since a more varied user group hopefully will result in a system that is attractive to a wider user base. Based on the feedback provided by the different user tests, the functionality and interface will be continually revised and refined.

A small number of more formal user tests, with a predefined set of questions to be answered by a somewhat larger group of users, will also be performed.

Major milestones will be scrutinized by MedView staff, and final release candidates of the system will be tested on actual oral medicine personnel.

6. Related Work

Computer-based forms have been around at least since the invention of the graphical user interface, if not even longer.

Several efforts have been made to produce applications that will automatically generate a form based on some type of template. Some of them have even attempted a dynamic approach, allowing forms to change based on various variables, such as previous user input or changes in the current environment. An even smaller subset try to provide one or both of these capabilities for use on the web.

The following section describe the different applications and technologies that are available today that may have some significance for the progress and success of the MedForm project.

6.1. Interactive HTML Documents

Interactive HTML[14] is a browser-neutral system that combines the most common types of services used in Web browsers such as plug-ins, Java-style applets, and JavaScript-style scripts into a uniform architecture.

6.2. Active Forms

Active Forms[15] presents dynamic forms using Tcl/Tk applets and the SurfIt HTTP/HTML browser implementing functionality such as conditional presentation of blocks of fields and minimal non-redundant data entry.

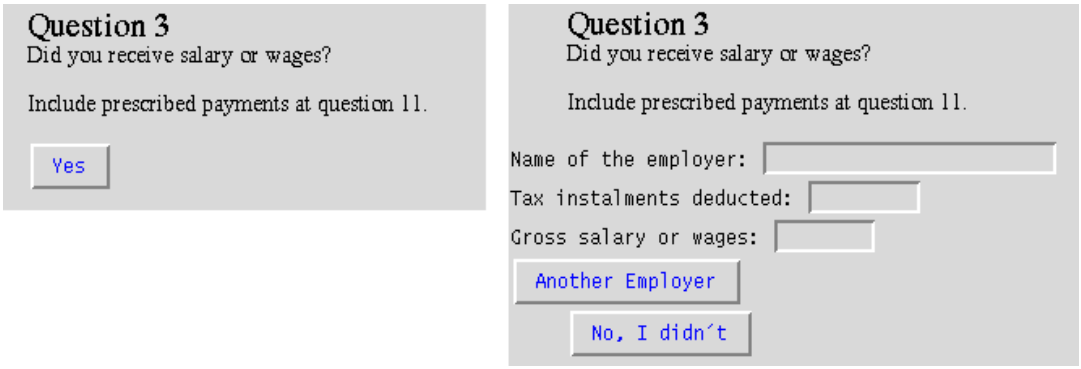


Figure 5. Active Forms before and after clicking the "Yes" button.

6.3. NLMenu

The NLMenu System[9] is a menu-based approach to natural language understanding, allowing users of the system to input data using menus that change dynamically, based on previous input.

Since all input is chosen from menus that are system-generated, every possible input can be understood by the system, thus giving a 0% failure rate. A companion system that can automatically generate interfaces to relational databases has also been discussed.

6.4. Adaptive Forms

Adaptive Forms[10] is similar to NLMenu, implemented using text-input areas instead of menus. The system uses a context-free grammar, similar to BNF, to encode dependencies of the input.

Input fields that have been filled are collapsed. Fields yet to come are filled if the input can be unambiguously decided based on previous input. Otherwise a selection of applicable values are presented.

Adaptive Forms have been used for specification of air campaign objectives, creating sentences such as “stop advance of invading forces outside of weapons range of the oil fields during phase 2”.

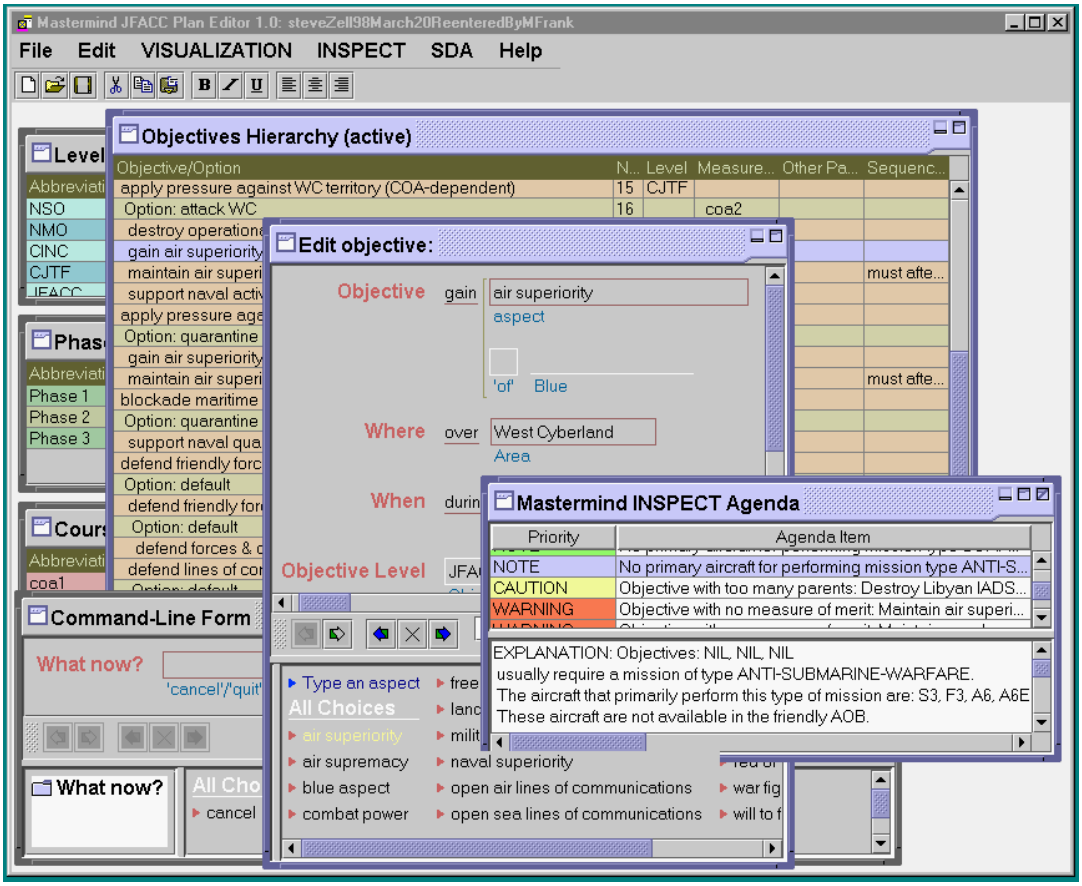


Figure 6. Screen shot of Adaptive Forms.

6.5. Dynamic Forms

Dynamic Forms[12] provides the capability to change, hide and expose fields in a form based on user input using a form description language. The structure consists of sections and subsections so that organization and navigation of the information can be achieved effectively.

Key features, besides dynamic field visibility and layout based on user input, is early validation of field values, computation of field values and repetition of fields.

6.6. Web Dynamic Forms

Web Dynamic Forms[13] provides the functionality of Dynamic Forms[12] for use on the Web as Java applets. This allows an interactive form to communicate in the background with Web servers and exist inline with other HTML objects.

Figure 7. Web Dynamic Forms with some collapsed and some expanded sections.

6.7. Intelligent Interfaces for Web-Based Clinical Trials

The system developed during the D3 project Intelligent Interfaces for Web-Based Clinical Trials[7], and improved during the summer of 2003 by Claes Westlin and David Breneman, has been used as the starting point for MedForm. This system is discussed in detail in the System Analysis section.

6.8. XIML

The eXtensible Interface Markup Language[16] is an attempt to standardize the representation of a graphical user interface for use on different types of devices such as a desktop, a PDA or the Web.

Since is a realistic assumption that MedForm could be deployed on other types of presentation software than standard web browsers, it may be a good idea to have some sort of standardization concerning the user interface built in right from the start.

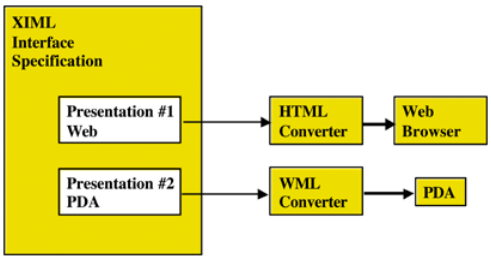


Figure 8. An example of how XIML could be used to generate content for two different types of devices.

6.9. XForms

XForms[17] is W3C's specification of web forms that can be used with a wide variety of platforms including desktop computers, hand helds and information appliances.

They are similar to HTML forms, but allow decoupled data, logic and presentation, validation, required fields, support for structured form data, multiple forms per page, and pages per form, support for suspend and resume and seamless integration with other XML tag sets.

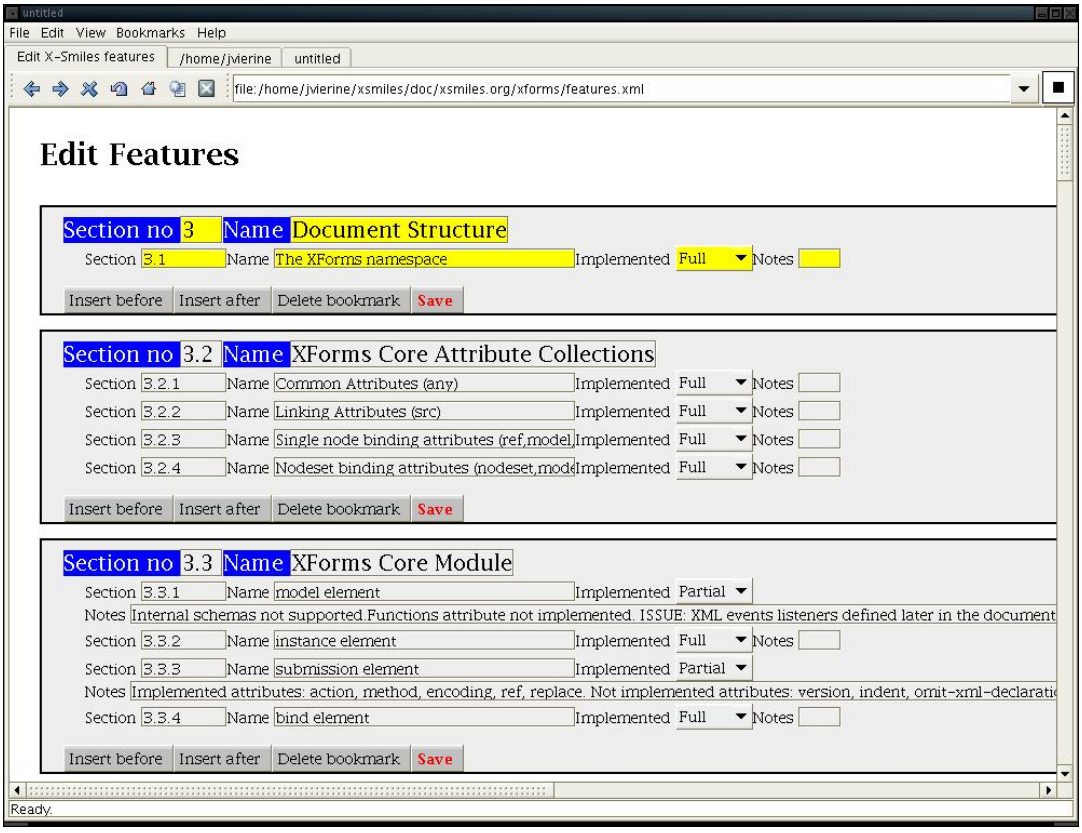


Figure 9. X-Smiles, a browser with XForms support, written in Java.

7. Analysis

A number of paper forms currently used for clinical trials were studied in order to define a reasonable set of question types to be able to represent in the web based equivalent. None of them posed any particular problems, but some required a somewhat modified approach.

For example, a common practice while producing paper forms is to ask the user to list a number of items, such as medicines currently being used. The reasonable method in a computer based setting is to allow the user to choose values from a list, and only type in a new value if the desired one is not available in the list. This eliminates spelling errors and ensures that the answers will have a format that can easily be compiled and used for further research.

All of the systems described in the above section are more or less relevant and applicable. However none of them come even remotely close to being compliant with the list of requirements. Most of them require extra client-side software, which makes them unsuitable. The grammars and languages used by the systems are far too complex to be usable in this context.

However, many of the presented ideas, especially the ones concerning dynamic and adaptive forms for the Web, will be considered for incorporation into the project.

7.1. XForms

A functional XForms implementation would be exactly what would be required to make the implementation of MedForm a relatively painless effort. Sadly, there is for the time being no support at all for XForms in any of the major browsers.

Getting the various applications that do exist to work is not easy, and doing something even remotely usable appears to be next to impossible, at least in combination with the available hardware and software.

7.2. XIML

Standardizing user interaction data is a good idea. However, for now it does not appear to be what the MedForm project needs, as it would add another layer between the form description language and user interface.

It should be just as easy to generate form presentations directly from the MedForm source files. Besides that, the XIML Forum is still refining the specifications of the standard. Once the standard has matured, and gained acceptance in the industry, it will merit another look.

7.3. Intelligent Interfaces for Web-Based Clinical Trials

The system is a good starting point for MedForm. However it does have several obvious flaws.

First of all, the XML format used for defining questionnaires does not follow any type of standard, so it cannot be used by any of the other applications in the MedView portfolio. The format itself is not suited to handle the tree-like structure in which information within the MedForm project is stored. Besides that, it does not separate the information from the graphical layout, with tags for position, page breaks and so on.

The data collected by the system is stored in a database in a flat structure which basically is an array with the question ID used as a key to access the answer (value). This makes it virtually impossible to share the information with other MedView applications, which generally handle data stored as tree files.

The editor, used to create the XML files, is based on the presentation of a question, not what type of question it is. This makes the editor rather complex, forcing the user to decide, for example, if the values to a single choice question should be presented as a drop-down menu or as a series of radio buttons.

The application solicits input from the user via a well-known forms paradigm that emulates their pa-

per equivalent. The user is assumed to be familiar with these and it is well suited for the input of simple attribute-value data. However, the user input may or may not be applicable depending on previous user input.

In paper-based forms, this is solved by marking questions "fill out only if you answered no to the previous question." Most electronic forms suffer from the same problem. The interface becomes cluttered, making the form difficult to understand and the questions difficult to answer.

8. Design

Due to the iterative nature of the development process the design stage was revisited several times after the original design had been specified.

8.1. Preparatory Design

A complete reevaluation of the ideas that the Intelligent Interfaces for Web-Based Clinical Trials project was based on lead to a set of goals to be implemented.

The different question types available were reviewed, some were enhanced and some were discarded. The mechanism for choosing the presentation type for each question was moved from the content creator to the system itself, based on variables such as the number of available alternatives.

The XML format was redesigned based on the XML format used by MedRecords, with certain additions. The question types "vas", "text" and "image" were added, as well as the required forminfo tags, in principle unchanged from the XML format used by Intelligent Interfaces for Web-Based Clinical Trials. Hopefully, this standard will be adopted and developed further within the MedView project. The MedForm DTD can be used for validation of MedForm XML. A complete DTD and a sample file has been included in Appendix A.

A module for generation of tree files, the data storage format currently in use within MedView, was designed to replace the current database format. It was constructed to follow the structure of the questionnaire defined by the XML file.

A set of design decisions were made concerning the user interface, in order to guide the implementation of the different question types.

- Clutter should be kept to a minimum by hiding elements that the user does not need access to at the moment
- Whenever possible, the user should be able to choose a value, as opposed to typing a value
- All actions must be reversible, allowing the user to make mistakes
- Each element type must be easy to understand and use without prior experience

Language support was added, making it easy to translate all static text appearing in the system, such as the labels on buttons, instructions, and so on, to other languages.

A new view, allowing the user to see the questionnaire divided into several sections, one for each category, was designed. The user may choose to switch between this new view and the original one, with the entire form on one page, at any time while answering a questionnaire.

8.2. Iteratory Design

Some sort of prototype or testing scenario was designed for each of the original design decisions. In some cases the functionality had to be completely implemented before it could be effectively tested. If the design was found to meet the requirements it was kept (or fully implemented) but if it did not, the decision had to be reevaluated and allowed another cycle in the development process.

Based on conclusions from the user testing some parts of the system were redesigned completely. One such detail was the user preferences subsystem that also handled the administrative tasks such as keeping track of which users are affiliated with a certain form. This change dismissed the need for a database altogether, significantly extending the applications portability. The preference files and file structure will be discussed in detail in the Implementation section.

Some of the other details that were changed based on user tests were the manner in which term values are ordered, how the web page behaves when new values are added and how the compilation if the entered data is displayed before submission.

8.3. Dependency Rules

One of the most important issues to be dealt with was the concept of dependency within a form. Such dependencies should relieve the user of the cognitive stress induced by the equivalent of paper forms' constructs such as "If you answered X to question Y, please continue to question Z." Since a computer display can be updated dynamically, there is no reason to use instructions of that sort.

A significant amount of the design effort was the creation of the dependency rules. Many rewrites of the specification were required, and several prototypes tested, before a satisfactory combination of generality, simplicity and dynamicality could be achieved.

Ideas such as allowing rules to substitute a question's current set of term values for another, allowing a question to appear or disappear based on if a certain question had been answered or not, or allowing the visibility of previous questions to toggle the visibility of following questions were evaluated.

It was discovered that the only type of dependency that would be truly useful in many different types of forms is a rule that would command the visibility of other questions based on the current answer(s) to the current question. The details of the final dependency rule system are discussed in greater detail in Implementation section.

Any given question can of course have an arbitrary number of rules, allowing potentially large sets of questions to have their visibility toggled at once.

When a certain question's visibility is set to false, all questions that have become visible due to the first question's dependency rules will become invisible as well. This works in a recursive manner, allowing entire sub trees of questions to become visible or invisible at once.

9. Implementation

Large portions of the system were implemented in a trial-and-error fashion, allowing frequent revisits to the design stage based on the results of user testing.

9.1. Adaptation of Existing Code

As a first step, with the Intelligent Interfaces for Web-Based Clinical Trials system as a starting point, most of the code was rewritten, basically implementing the existing functionality, but with radically different code under the hood. This was necessary because of the inherently different way in which MedForm is intended to function.

Many graphical user interface enhancements were added, such as a new set of icons, a logout button and tweaked style sheets with more appropriate spacing, fonts and colors.

The server-side code of the system was written entirely in PHP[18]. PHP was chosen because of its open-source licensing and general availability for virtually any platform. It is also extremely well-documented and widely used. Besides that, much of the code of previous work that has been reused was written in PHP.

Code that allowed entered data to be saved in the form between pages and sessions was added, as well as the summary page allowing the user to review his or her data before finally entering it into the system.

The current database in use is MySQL[19]. All code for data processing is stored in a single PHP file that can easily be adapted to use a different database, or even entirely different methods of data storage, such as a collection flat files.

Once version 1.0 was released for testing of the new and improved XML format, tree file generation and form editor, development of new functionality could begin. A list of requested features was organized, allowing time for user testing after each implementation.

Some of the major improvements included:

- The possibility for form users to add new term values to the current list of values
- The possibility for form users to upload one or several images for inclusion with the resulting data
- The possibility for form creators to mark certain input fields as "mandatory" and others as "optional"
- Dynamic updating of the form based on previous input
- The possibility for form creators to edit XML code and termValue definitions directly

9.2. Data Formats

The MedForm system currently depends on three data formats. The form description format holds the structure of a form. A second file accompanies it, keeping track of the term values that the various terms defined in the form may assume. Finally, the MedView data storage format is used for saving gathered information to disk.

9.2.1. Form Description

Forms are described by the MedForm XML format. This format is a slight adaptation of the MedView Form Description Format, adding details such as meta information and dependency rules.

The general structure of a form is roughly as follows:

```
EXAMINATION
        FORMINFO
                AUTHOR
                TITLE
    ...
        CATEGORY
                INPUT
                INPUT
                ...
        CATEGORY
                INPUT
        ...
```

The complete Document Type Declaration for the MedForm XML format can be perused in Appendix A. There is also a sample form available in Appendix C. The format is fairly straightforward and should be sufficiently self-explanatory.

However, there are a few details worth noting. The "input" tag has an attribute named "translatable". This attribute is used by other MedView programs and does not have any significance for MedForm. The "info" tag is intended to allow the form creator to add information such as sub headers and short texts to a form. This may be extended to include images, sound, video, animations other media types. The "dependency" tag is described in detail in its own section.

9.2.2. Term Values

The termValue format is extremely simple. It has already been described in the MedView introduction. A sample termValue file can be studied in Appendix D.

9.2.3. Data Storage

MedForm generates files of the MedForm Data Storage Standard format. Currently, this standard as implemented as a set of Tree files, a MedView specific format. A sample output file is available in Appendix E.

9.2.4. Other Files

Besides the files used to describe forms and their results, a number of other files may be found in the different user's home directories. These include settings and temporary files used to store the current state of a form while it is in use.

The settings file have a simple structure that assigns values to a number variables. A typical creator settings file would look something like this:

```
pass = xxxxxx
usertype = creator
rname = David Breneman
email = david@breneman.se
```

Which variables are present depends on which type of user the file belongs to. Users of type "user" will have files that also include information about which creator has created the account, which form the user is affiliated with, and so on.

9.3. File Structure

Every user of the system has a home directory that is created at the same time as the account. This directory is located in the "user_files" directory in the application root, but the location may be changed by setting an environment variable. The content of the home directory varies depending on what type of user it belongs to.

The following listing shows the contents of a typical content creator account. The three present ob-

jects are the settings file, a library holding the source files for the content creators current examinations and finally a directory that holds a subdirectory for each active examination in which generated tree files are stored.

```
db@db:user_files% ls -l david/
total 8
drwxr-xr-x  25 www  db  850 22 Sep 23:15 forms
-rw-r--r--   1 www  db  215 22 Sep 23:12 settings.txt
-rw-r--r--  14 www  db  476 23 Sep 00:58 treefiles
db@db:user_files% ls -l david/forms/
total 264
-rw-r--r--  1 www  db  24950 22 Sep 22:18 sampleexam.termValues.txt
-rw-r--r--  1 www  db  12098 22 Sep 22:18 sampleexam.xml
...
db@db:user_files% ls -l david/treefiles/
total 0
drwxr-xr-x  4 www  db  136 22 Sep 22:17 sample.mvd
...
db@db:user_files% ls -l david/treefiles/sample.mvd/
total 0
drwxr-xr-x  31 www  db  1054 22 Sep 22:17 Forest.forest
drwxr-xr-x   3 www  db   102 22 Sep 22:17 Pictures
db@db:user_files% ls -l david/treefiles/sample.mvd/Forest.forest/
total 232
-rw-r--r--  1 www  db  918 22 Sep 22:17 1111111111_040528141026.tree
-rw-r--r--  1 www  db  348 22 Sep 22:17 2222222222_040526230354.tree
...
db@db:user_files% ls -l david/treefiles/sample.mvd/Pictures/Pictures/
total 3872
-rw-r--r--  1 www  db   21000 22 Sep 22:17 040526224124.jpg
-rw-r--r--  1 www  db   21000 22 Sep 22:17 040526224414.jpg
...
db@db:user_files%
```

A typical content creator account only holds a settings file and a number of temporary files used by the system to maintain the state of a form while the user navigates it during a session.

```
db@db:user_files% ls -l testuser/
total 8
-rw-r--r--  1 www  db  107 22 Sep 22:47 settings.txt
drwxr-xr-x  5 www  db  170 23 Sep 06:29 tmp
db@db:user_files% ls -l testuser/tmp/
total 24
-rw-r--r--  1 www  db  447 23 Sep 06:29 answers.txt
-rw-r--r--  1 www  db  275 23 Sep 06:29 depcreated.txt
-rw-r--r--  1 www  db  220 23 Sep 06:29 visible.txt
db@db:user_files%
```

9.4. Dependency Rules

One of the last subsystems to be implemented was the methods for handling the dependency rules. This turned out to be a bad decision since many other parts had to be redone in order to accommodate it. Many different solutions were implemented, tested and discarded because they did not work in an acceptable fashion.

Some required the current page to be reloaded, a method most users found very confusing and frustrating. Others would only allow questions on future pages to be updated, and still others required an extremely complex set of rules. All in all, the entire rule handling portion of the code was rewritten four times, and other parts of the system with it, with many more changes to smaller parts of the system.

Here is the XML code used to define the question in the example below:

```
<INPUT type="single" freetext="false" visible="true"  required="true">
  <TERM>Country</TERM>
  <DESCRIPTION>Which country do you live in?</DESCRIPTION>
  <COMMENT>Select one from the list</COMMENT>
  <DEPENDENCY>
    <DEPRULE>
      <DEPVAL>USA</DEPVAL>
      <DEPTERM>City-USA</DEPTERM>
    </DEPRULE>
    <DEPRULE>
      <DEPVAL>Sweden</DEPVAL>
      <DEPTERM>City-Sweden</DEPTERM>
    </DEPRULE>
  </DEPENDENCY>
</INPUT>
```

Each rule simply defines which question (term) should receive status "visible" for a certain chosen value. This method was chosen over other ideas, such as allowing a question to have several possible sets of values, because of its simplicity. This does restrict what can be done in terms of dynamic dependency somewhat, but it allows implementation of most realistic types of forms.

The following set of figures demonstrates how the dependency rules behave in action.

Figure 10. Before choosing a country.

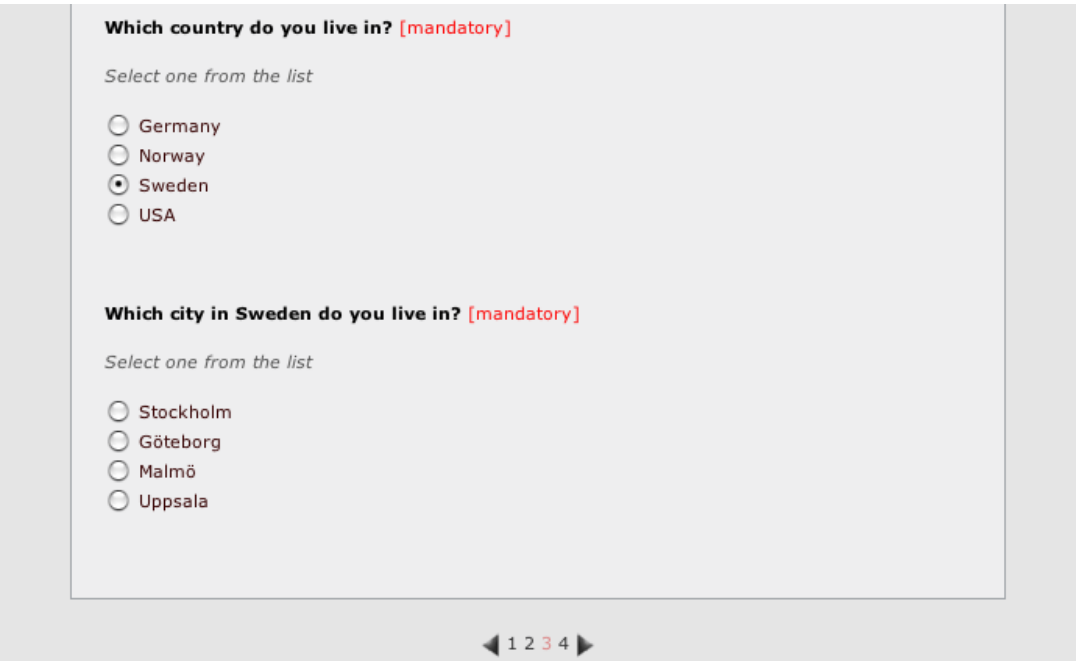


Figure 11. After choosing the value "Sweden".

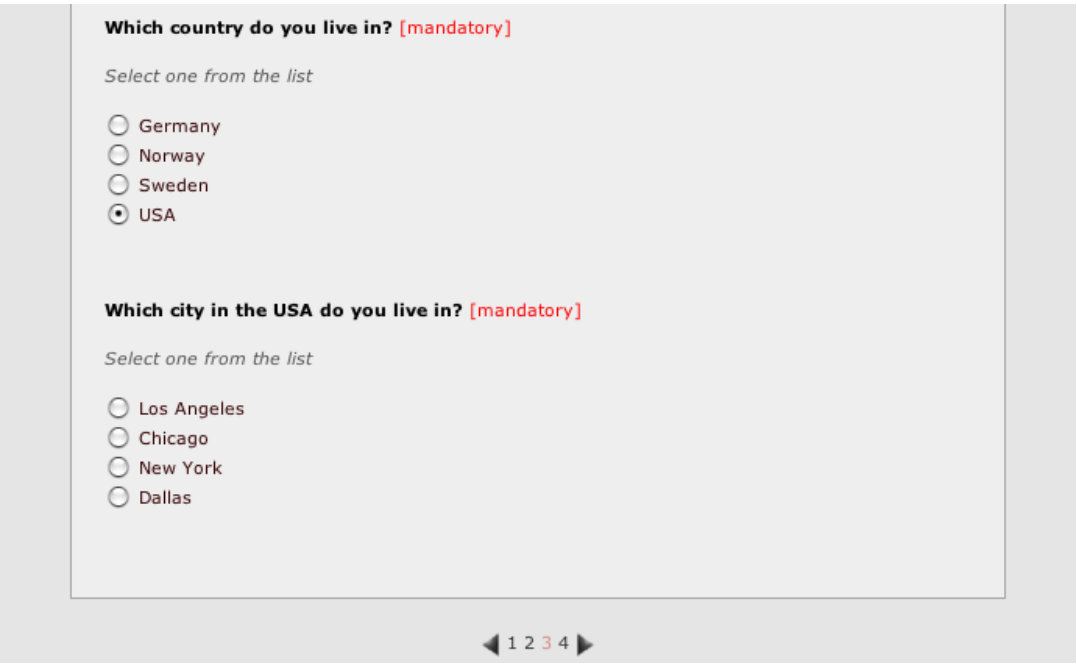


Figure 12. After choosing the value "USA".

9.5. Client-Side Development

The client-side forms and administration tools and editors are presented using nothing but standard HTML, CSS and JavaScript. This posed several interesting problems, mostly based on the fact that the client and server cannot communicate in an effective way during a form session.

One of the most difficult challenges proved to be keeping the entire state of the current form syn-

chronized on the server and on the client, with the HTML form elements in conjunction with JavaScript as the only means of communication. This problem was solved using a large number of hidden HTML form elements that are updated in response to every action taken by the user, and submitted to the server every time a new page is loaded. This includes logging out or changing viewing or language settings.

Another interesting challenge was presented by the fact that JavaScript implementations do not always work as expected on different platforms. For example the following code worked perfectly on every platform, except Microsoft Internet Explorer for Windows.

```
document.getElementById(id).innerHTML = newnode;
```

Changing the code in the following, very minor, way allowed it to work with MSIE as well.

```
var tmpnode = document.getElementById(id);
tmpnode.innerHTML = newnode;
```

Generating the form questions on the fly as requested by different dependency rules also required some imaginative code.

10. Description

The MedForm system is comprised of three modules. The administration interface allows accounts and forms to be manipulated, the editor allows the user to create forms, and finally, the form interface allows users to enter data into the system.

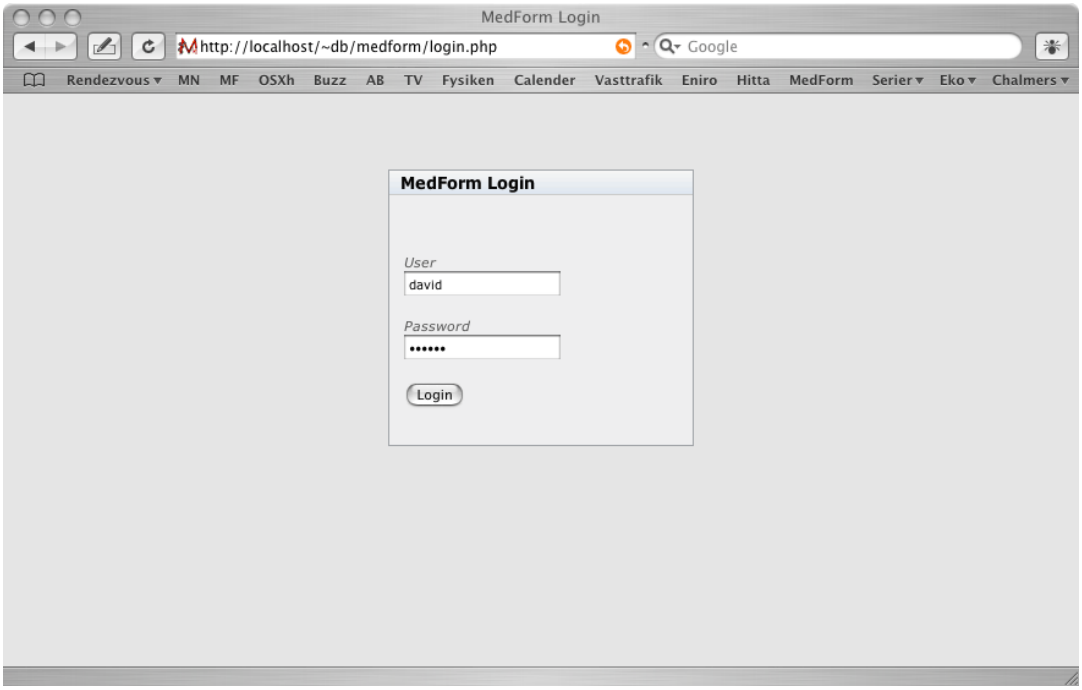


Figure 13. Login screen.

10.1. Administration Interface

The administration interface allows the system *administrator* to create and manipulate an arbitrary number of accounts for *form creators*.

The form creators can use the editor to create an arbitrary number of forms simply by clicking the "New Exam" button.

Once created, an examination can be activated by choosing the appropriate examination from the menu and clicking "Activate". Once an examination has been activated an arbitrary number of *form filler* accounts can be created for users of the examination. This is done by simple entering the name, user name, e-mail address and an optional password for each user.

Clicking the "Download" button will begin a transfer of the entire collection of tree files that has been generated so far for that particular form to the user's hard drive. The files, including any attached images, are archived in the zip format. The same functionality can be accessed by loading a certain URL, allowing programs such as MVisualizer to easily import the data directly for analysis.

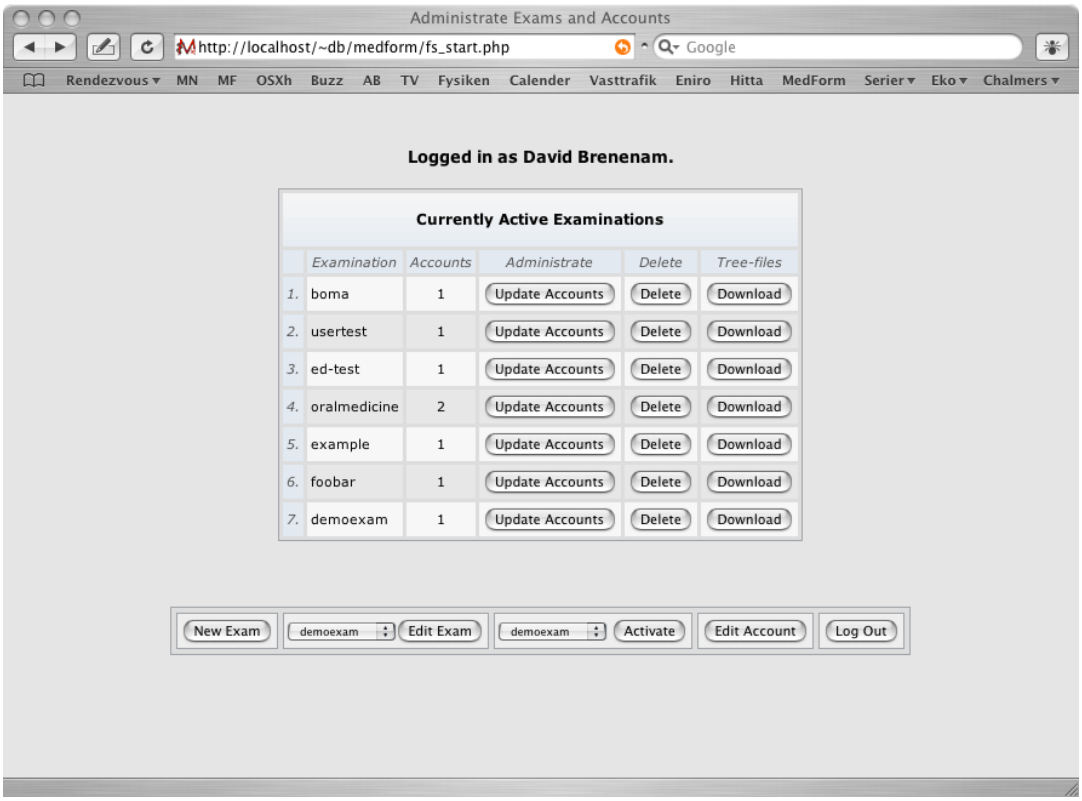


Figure 14. Administrative interface.

10.2. Questionnaire Editor

The form editor is a web-based tool available to form creators. It allows the user to create and edit forms, including the corresponding term values, using a graphical user interface. Form creators can create new categories and add new questions, sub headers and blocks of text to them. Created questions can be moved around or deleted.

The editor currently lacks some functionality, such as the ability to edit questions. Therefore it also allows expert users to edit the XML form source code and the term-value files directly. This feature can also be used to copy and paste forms or lists of values created with other tools, such as FormEditor.

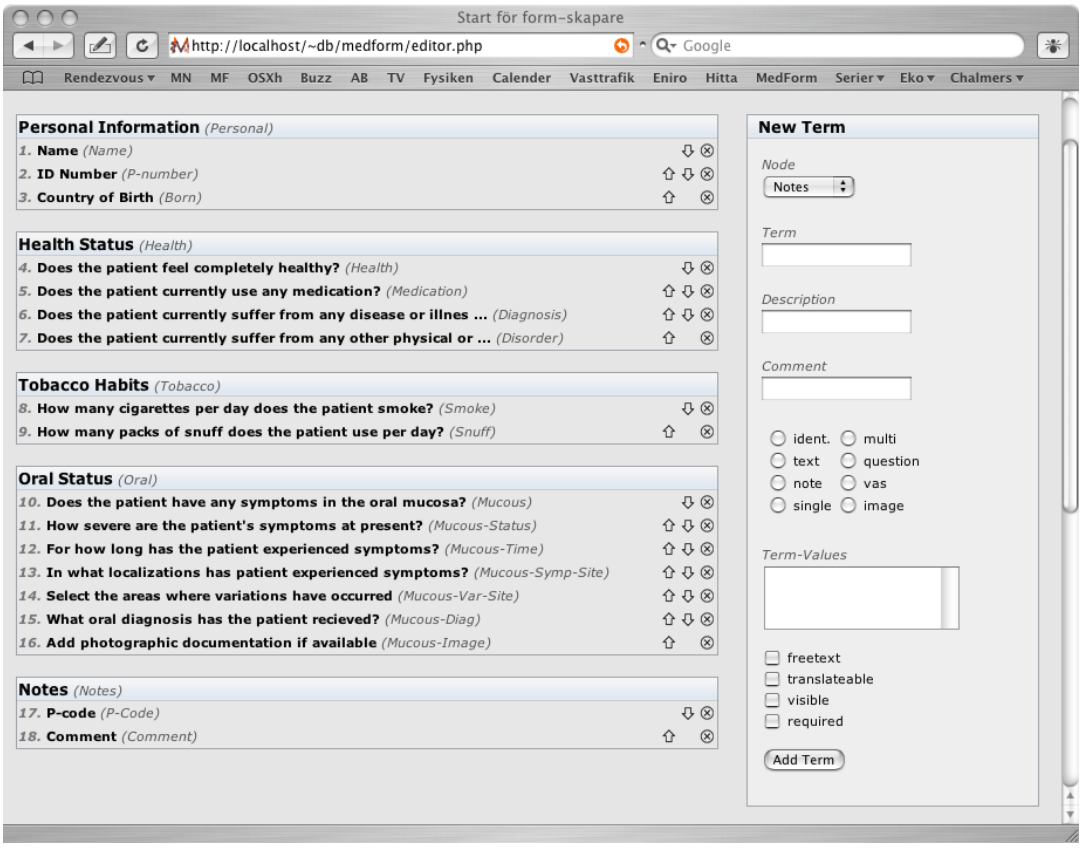


Figure 15. Form editor.

10.3. Questionnaire

Users provided with accounts by a form creator can log in to the system and are redirected directly to the form connected to the account on log in. The form is presented as a web-page with standard HTML form elements. It also has a header including information about the form such as the title and author's name and contact information. There are also icons allowing the user to toggle the page view and log out.

The navigational bar at the bottom should feel familiar from search engines such as Google, allowing the user to move back and forth between the various pages of the form.

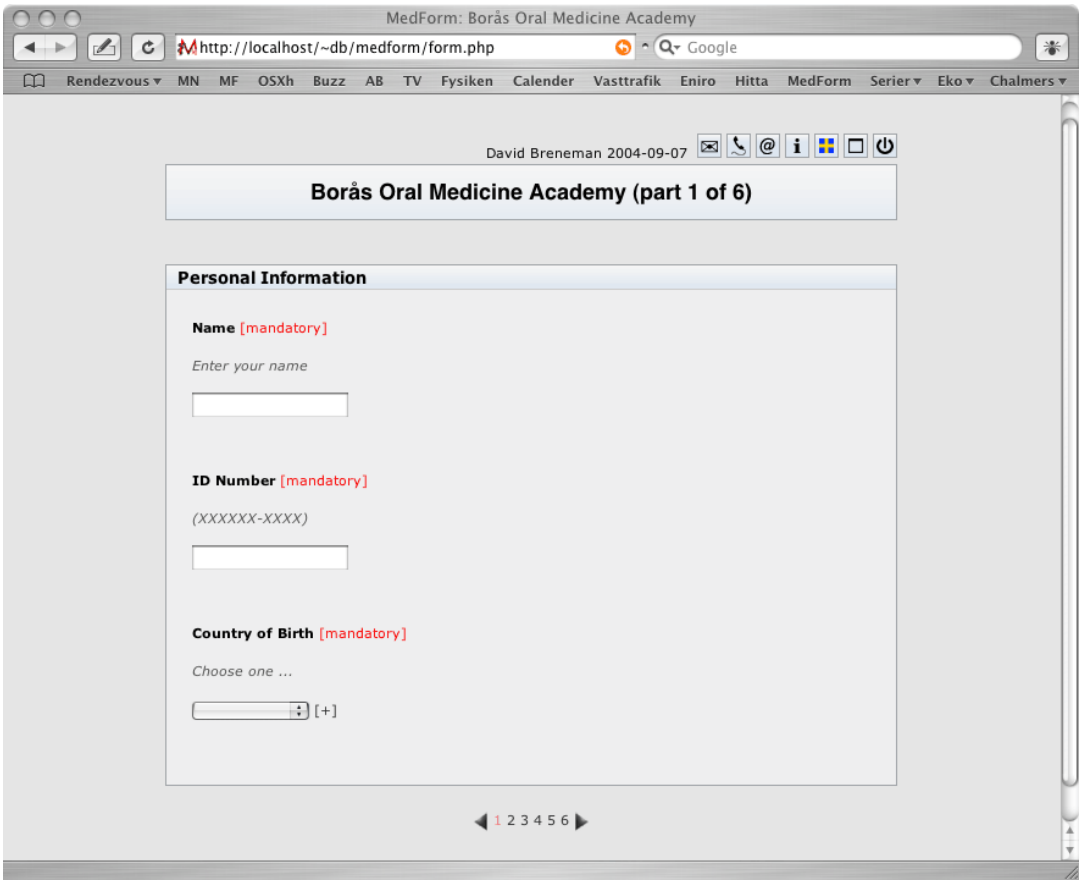


Figure 16. Questionnaire - example of a first page.

The default page view presents the form, divided into sections, with one section on each page. The page view toggle allows the user to switch between the default mode and a view that presents the entire form on one page. Logging out saves the current answers so that the user can continue where he or she left off before logging out.

Which form elements are presented for each question is decided based on two factors, namely, which type of term-values the question has and how many term-values there are.

Single - the user may choose one value from a number of choices.

If the number of term-values is less than 10 a number of radio buttons are shown.

Figure 17. Single - radio buttons.

If the number of values is 10 or greater, the choices are presented as a drop-down menu.

Figure 18. Single - drop down.

Multi - the user may choose an arbitrary number of values from a number of choices. The same method of adding additional values to the system is available, provided the form permits new values to be added.

If the number of values is less than 10, a number of checkboxes will be presented to the user.

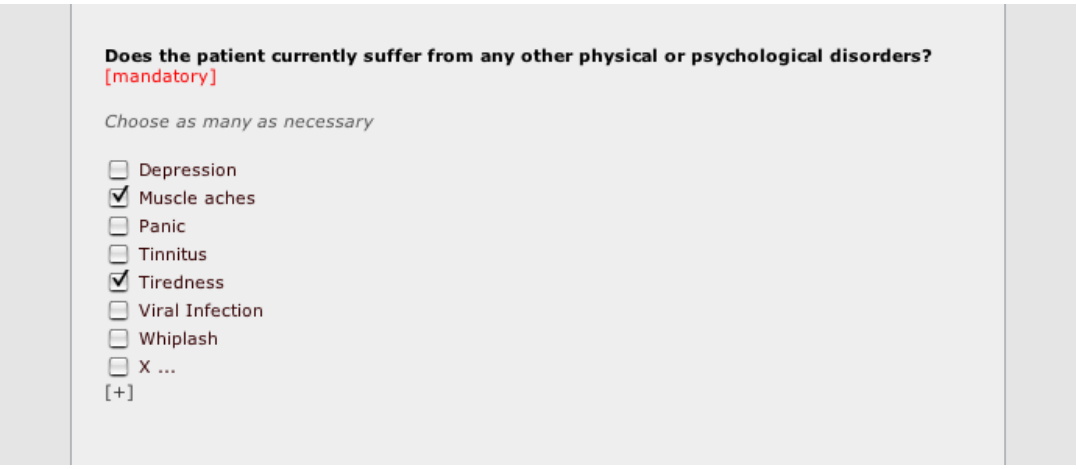


Figure 19. Multi - check boxes.

If the number of term-values is 10 or greater, an input type called look-up-and-add (LUAA) is used. It allows the user to begin typing the desired value. Matches will be highlighted in the left list. Double-clicking a value will copy the value to the list of selected values, on the right. Double-clicking values in the right list removes them from the list.

Figure 20. Multi - LUAA.

Interval - the values are presented as radio buttons representing the intervals between the different values.

The term-values A, B, C and D would generate the interval choices "A - B", "B - C" and "C - D".

Question - A list of values is presented, with the added twist that the user is also presented with the possibility to add a free text value and choose a unit from a menu.

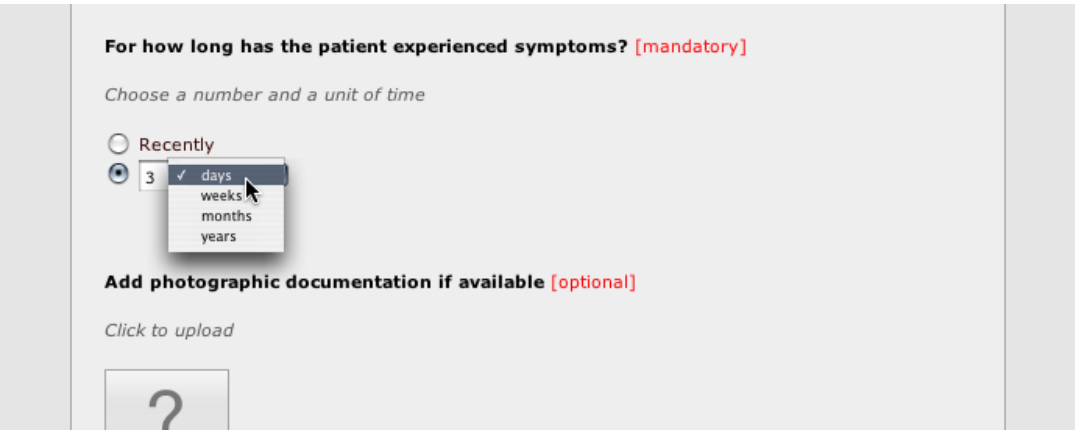


Figure 21. Question

VAS (Visual Analog Scale) - the user is allowed to choose one of 100 values on a non-graded scale.

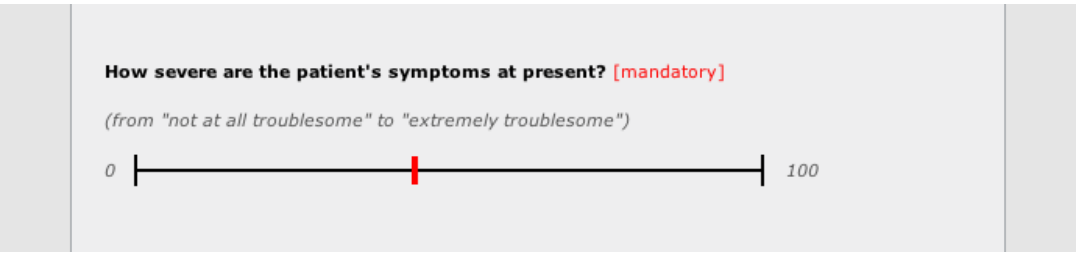


Figure 22. Visual Analog Scale.

Image - the user is presented with the opportunity to upload one or several images to the system. Clicking the question mark generates a pop-up window prompting for an image file. Once the image has been uploaded, it replaces the question mark, and a new question mark appears to the right of the recently uploaded image.

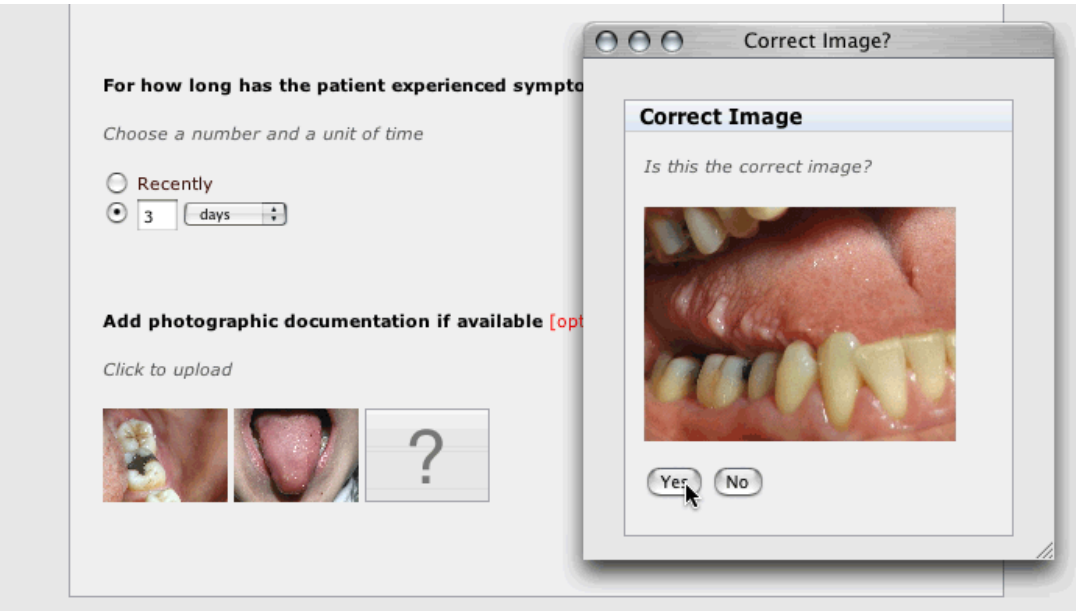


Figure 23. Image upload.

Text - a small text input field is presented, suitable for short texts such as names or telephone numbers.

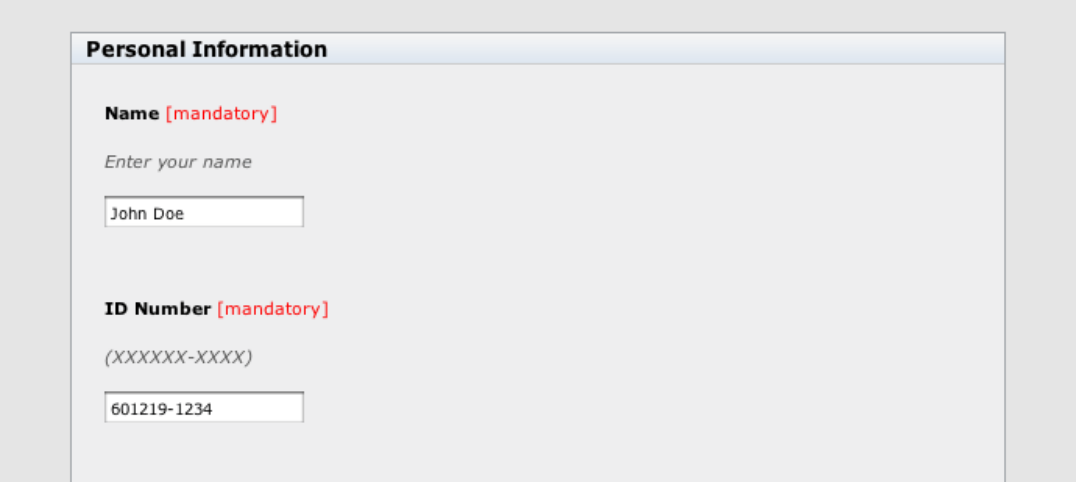


Figure 24. Two text fields.

Identification - identical to the text type, except that the provided value is also copied to the MedView specific identification field in the resulting tree file.

Note - identical to the text input, except that a larger multi-line text box is presented, suitable for longer texts such as a comment to the survey administrator or a description of an applied procedure.

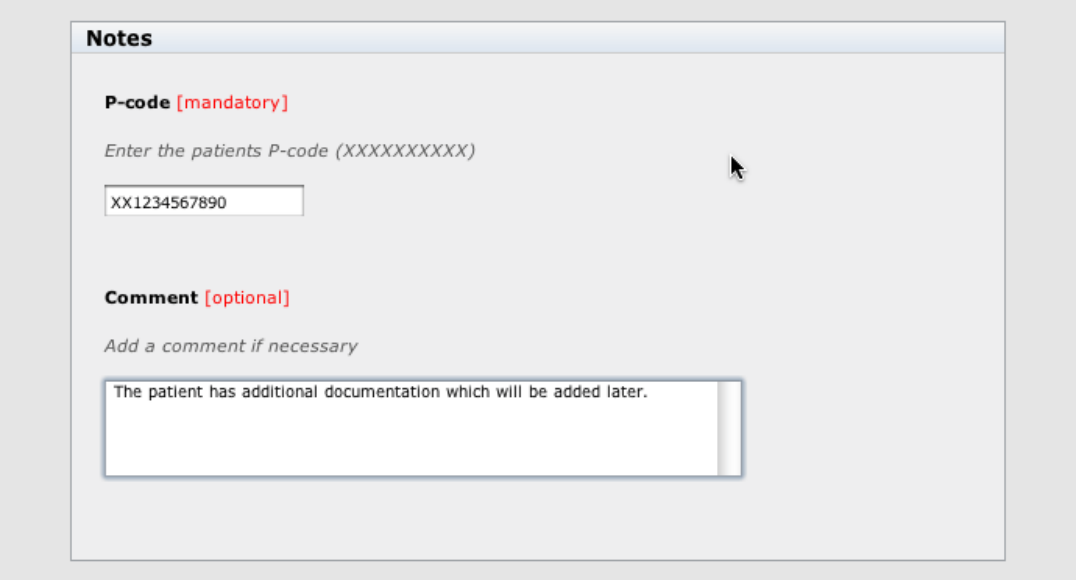


Figure 25. A note and an identification field.

If the user is allowed to enter new possible values into the system, a plus symbol is displayed after the current list of values. Clicking it will present the user with a pop-up window prompting for the new value.

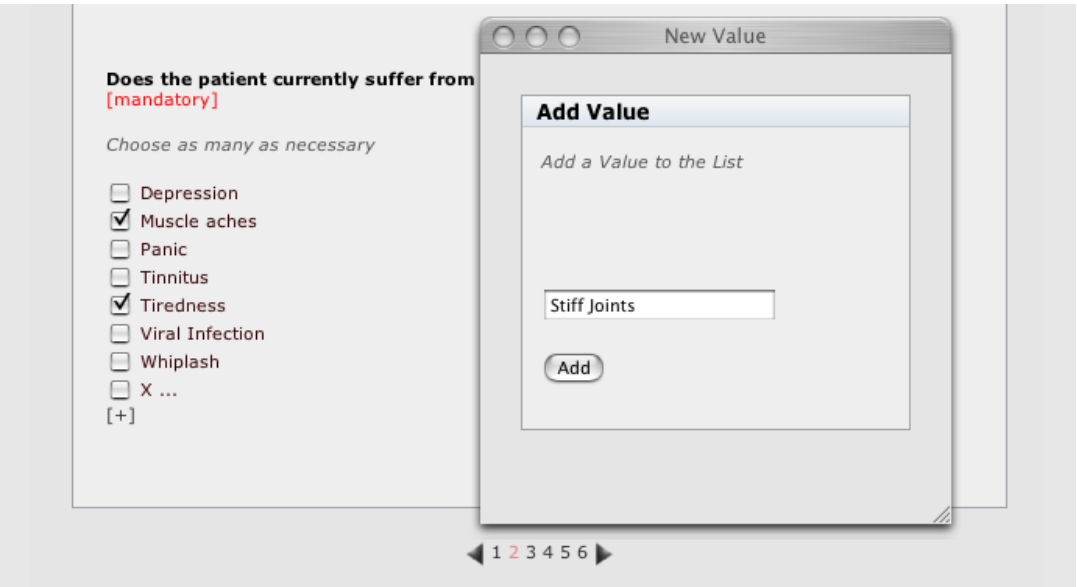


Figure 26. Adding a value to a list of values.

One of the unique features of MedForm is the ability to dynamically display, hide or change questions based on previous input. The following example demonstrates how answering "Yes" to a question will present several new questions related to the one that was just answered. The corresponding code can be perused in Appendix C.

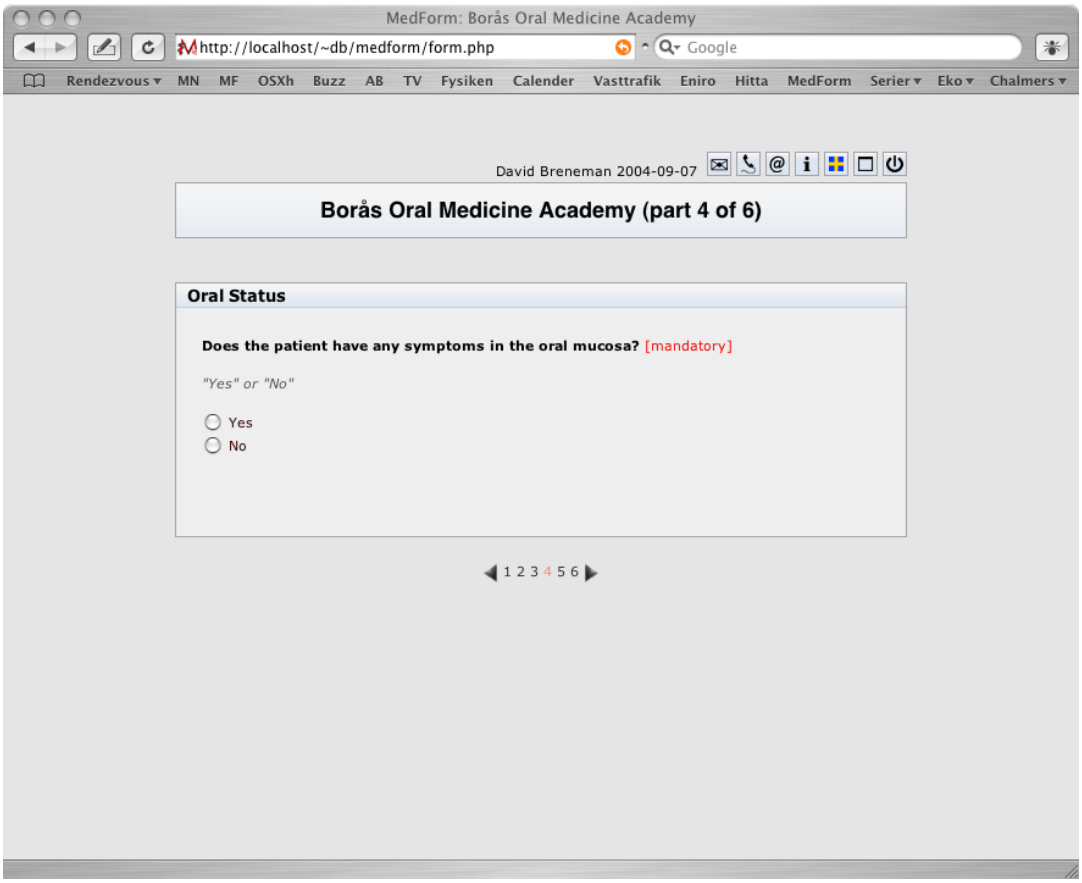


Figure 27. Before dependency rules are activated.

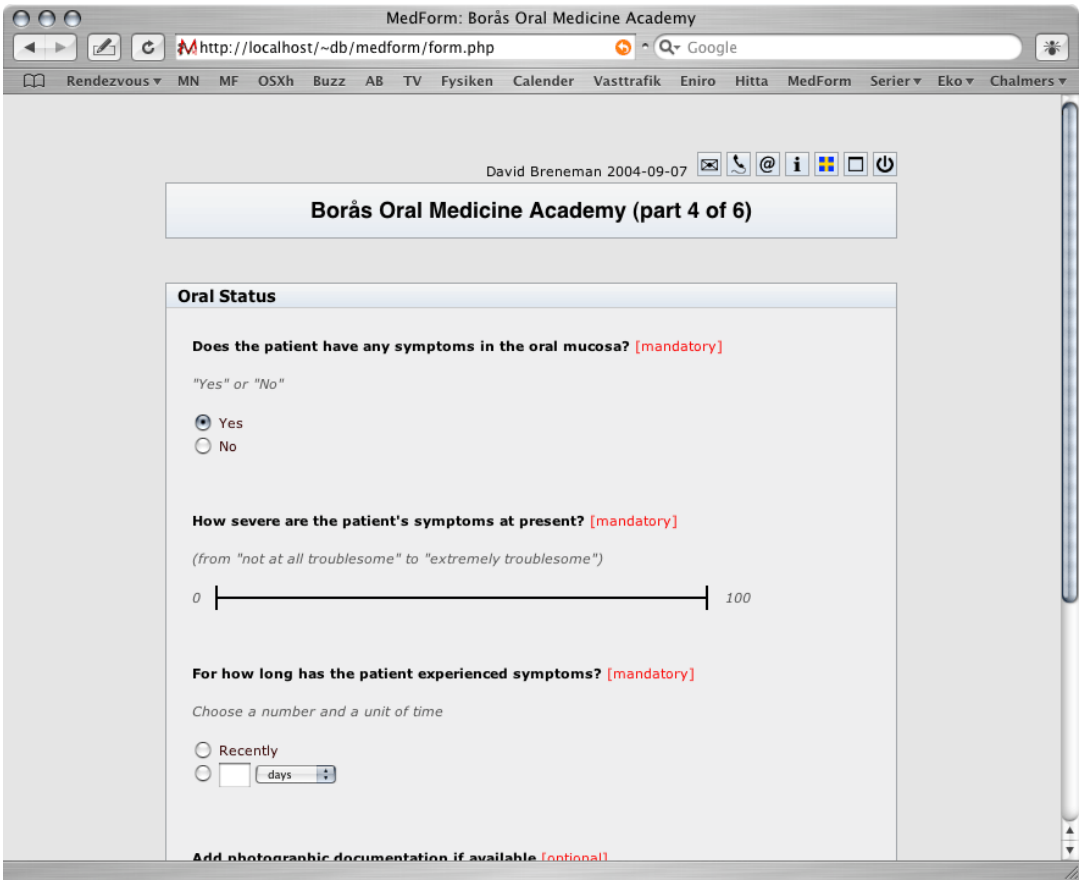


Figure 28. After dependency rules are activated.

Once the entire form has been filled, the last page displays a summary of all the questions and their answers. Information concerning if answering a question is mandatory or not is also given. If the user has failed to answer a mandatory question a warning is given. The user can choose to go back and make changes or submit the answers. Once the form has been submitted a corresponding tree file is created, and is directly available to the form creator.

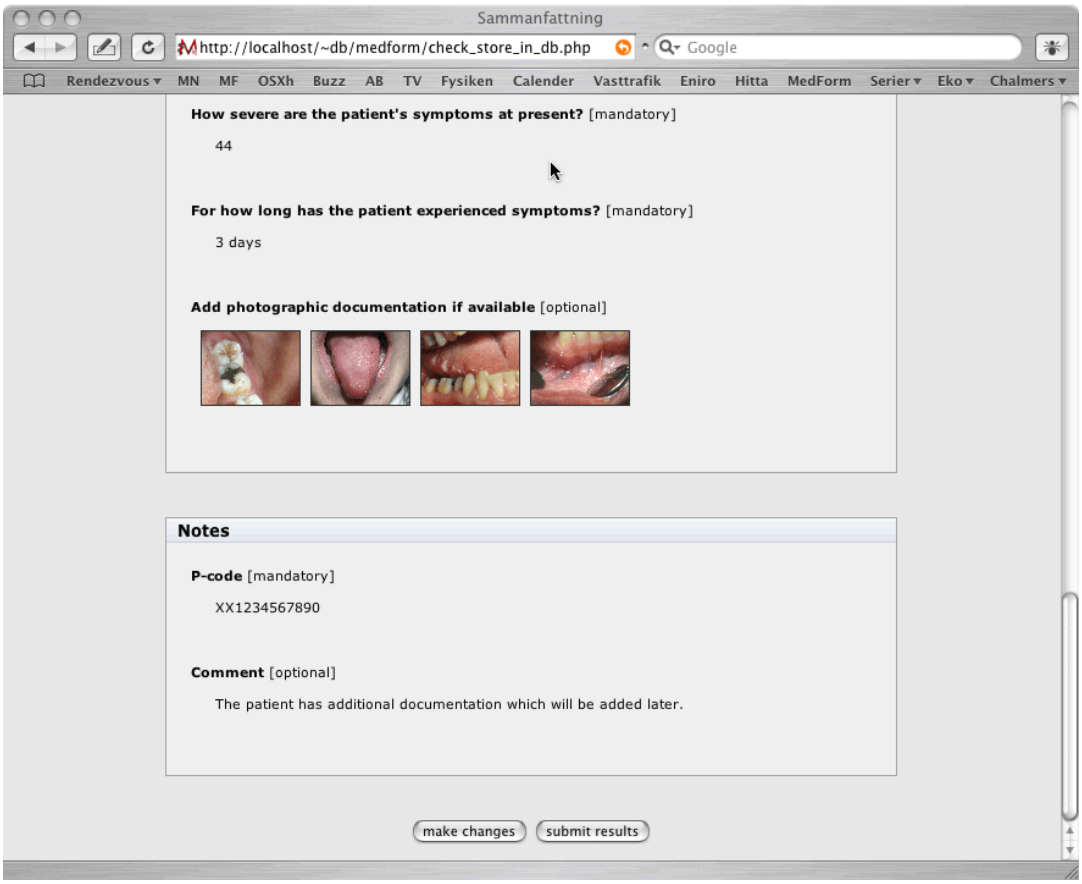


Figure 29. Form summary.

11. User Testing

As stated earlier, the number of actual medical professionals available for user testing during the development stage of the project has been very limited. Because of this, a number of other individuals have supplied comments and requests after every small iteration of the implementation phase.

After every major revision, a more formal user test was conducted via e-mail, giving the user a specific task to accomplish.

All in all, 26 such tests were conducted at different times during the development, allowing individuals with different operating systems, browsers and degrees of computer savviness to enter data into a test questionnaire containing most of the possible permutations of question types and types of sets of values. The users were only given brief instructions, and were told to try the different options, such as logging out before submission, switching view modes and so forth. Once the data had been entered, the user was presented with a number of questions concerning the experience.

Some of the questions asked were:

1. Do you believe that everything worked as intended?
2. Were any error messages presented at any time?
3. Was anything in graphical user interface difficult to understand or use?
4. Did anything odd or unexpected happen during the session?
5. Which operating systems and browsers have been used for testing?
6. Any other messages or opinions?

Only a few users at a time were allowed to test the system, leaving time to address identified issues with specific operating systems and browsers between batches of tests.

Most of the users had no problems understanding and using the system in general, but many useful descriptions of problems and valuable opinions concerning the interface and functionality of the system resulted from the user tests.

A few of the uncovered issues were:

1. Non-responsiveness on Windows/Explorer systems.
2. Line breaks in Notes questions are not properly displayed in the overview presentation.
3. Image uploads allowed the "slots" to be clicked in the "wrong" order.
4. Questions of type "question" do not save entered data after reloading the page on Windows systems.
5. Values in Single and Multi questions are not presented in alphabetical order.
6. Addition of values to Single and Multi questions should automatically select the new value.
7. Some MacOS X-based systems do not properly display non-ASCII characters in some situations.
8. There are too many steps involved in uploading an image.
9. The same value can be added to the term values several times.
10. The same value can be chosen from the LUAA-style Multi list several times.
11. The terminology used is somewhat computer-oriented, not human-oriented.
12. JavaScript errors using the OmniWeb browser.
13. The overview should link directly back to each question, instead of forcing the user to go back to the beginning of the questionnaire.
14. The system does not work with Netscape 4-style browsers.
15. Adding a term value reloads the page, forcing the user to scroll down to the previous position on the page.
16. Entered values are not saved when the auto-logout function is triggered.

The actual list was much longer, of course. However, this is a rather representative sample of the types of problems that had to be eliminated.

Other members of the MedView project have also provided useful feedback concerning the system between major revisions of it. When the opinions and requests from different users were not compatible, usually suggestions from the user base most likely to use the system in the future were

honored.

The final product will be tested in a clinical environment in the near future. Depending on the results from this test, some changes may be made. However MedForm is considered to be stable and usable in its current state.

12. Results and Conclusions

The major goal, to create an application that can save valuable time and money while conducting clinical trials, has been accomplished. It has been proven easy to use and adds significant improvements over equivalent paper-based forms. One such improvement is the possibility to adapt a presented form to the current user's preference concerning variables such as language or detail. Another is the simplification that the dynamic aspect made possible by the dependency rules. By only presenting information that is actually relevant to the user's current situation, the level of cognitive stress is reduced.

That said, there are still issues to overcome.

Several of the individuals participating in the user testing became frustrated when the application did not behave as expected. Sometimes the behavior was that which was intended, sometimes not. In any case, the system must be subjected to severe user testing before it can be put to use in an industrial environment.

Even if the system did work perfectly, and in strict accordance with every user's expectations, there are still individuals who do feel more comfortable using pen and paper. Hopefully, this attitude is an occurrence that will become less and less common with time, but at this point in time it cannot be ignored. This means that a wide-spread utilization, including allowing patients to fill out their own forms, is still a future goal. However, there is no reason not to have a group of computer-tolerant medical personnel use MedForm in their daily work.

On the more technical side of things, there are a few things to keep in mind.

The complexity of the system, and the source files that the forms are defined with, are related to the generality. It is extremely difficult to allow any possible type of questionnaire without an unreasonably complex description language. Comments during testing included statements such as:

- Why can the list of values not be horizontal instead of vertical?
- How can I create a list of alternatives, and an "other" text space at the end?
- What if I do not want my list of values to be automatically alphabetized?
- Can I force a page break without creating a new section?

Each individual request is easy to fulfill, of course, but to handle all of them simultaneously, as well as anticipating future requests, is an extremely intricate task.

The same holds for the dependency rules. It is simple to allow the basic case where a certain chosen value allows certain other questions to be presented. It is just as simple to add additional rules, for example, presenting a question only if a previous question has been answered. However, there is a considerable amount of conceivable rules that a form creator might want to use. Implementing all of them would increase the complexity of the rules system dramatically.

All in all, the project must be considered a success. The developed software integrates well with existing MedView applications and has high potential for becoming a valuable tool for clinical professionals. With more research and work, it could become even more useful and effective, even outside the current area of use.

13. Future Work

The available editor is definitely not the ultimate way to create the XML source files, since it does not update dynamically and lacks much of the required functionality, most notably creation of dependency rules. A question to deal with in the future would be to decide if a web-based interface is the way to go. There are other alternatives, such as updating FormEditor to handle the extended XML format, which would be extremely simple to do.

User interface portability is relatively easily achieved as it is, since the entire form is defined by the XML and the termValue file, both of which can be parsed for other kinds of presentation. However, use of XIML, or a similar standard for user interface definition, would make it even simpler to port forms to other platforms, such as handheld devices, mobile phones, touch screens or even for printing on paper.

As far as the current web-interface is concerned, there is room for quite a bit of improvement. There are many inherent problems with HTML and JavaScript, such as the lack of form validation and problems keeping the client-side and server-side scripts synchronized, and transferring variables between the two, based on user input. It is impossible to guarantee that the application will function as intended, since JavaScript can behave differently on two different builds of the same version of Internet Explorer, or depending on user settings. XForms would be a great step toward a partial solution, but before it can even be considered it must be included in the standard install of all the major browsers.

The form description format used by MedForm imposes certain limitations for what can be done with a form. For example, it is not possible to have a node depth greater than one, it is not possible to have subcategories nested inside categories. The same type of limitations are imposed by the termValue standard. For example, it does not allow questions of the matrix-like questions, which would produce values as pairs.

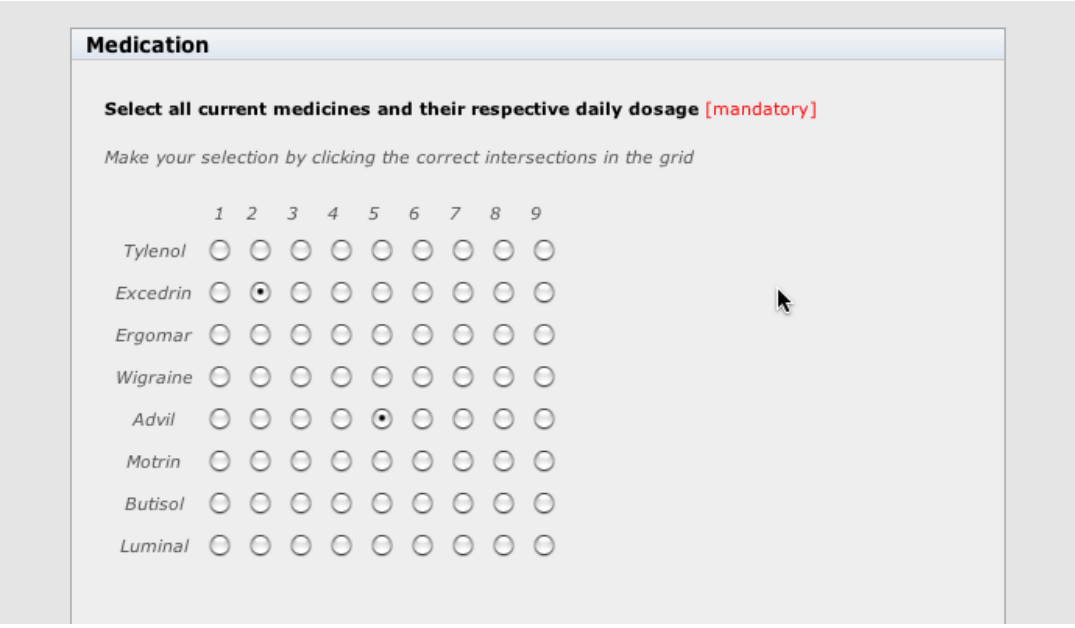


Figure 30. A matrix question type.

Another useful enhancement might be to allow subgroups of termValues, such as cities in a chosen country. It would also be useful to be able to have typed termValues, since the type of freely entered input cannot be restricted as it is. All of these suggestions would require changes to be made in other MedView applications for them to be useful, or even possible, to implement.

Developing the dependency rules to use a more advanced relational grammar, possibly not all that different from the one used by Active Forms, would allow MedForm to handle even more dynamic

forms. However, the current implementation is more than adequate to meet most current needs.

Some major improvements can be made to the code base to allow a more efficient and structured work flow to be attained during future development. The most urgent one would be to create a model representation of an entire form as a JavaScript Object that can be saved server-side between pages or sessions. This would greatly minimize the complexity of the code since much of the interface and document structure generation must be done concurrently on the server and on the client.

Many enhancements could be made to the user interface. One of the most obvious would be the addition of many more question types, and of course different representations of them. There may also be situations when it could be appropriate to award the content creator or the user more control over the presentation type of different question types based in his or her preferences.

References

[1] Youssef Ali, Göran Falkman, Lars Hallnäs, Mats Jontell, Ulf Mattsson, Nader Nazari, and Olof Torgersson. *An Overview of MedView*.

[2] Youssef Ali, Göran Falkman, Lars Hallnäs, Mats Jontell, Nader Nazari, and Olof Torgersson. *MedView: Design and Adoption of an Interactive System for Oral Medicine*. Medical Infobahn for Europe: Proceedings of MIE2000 and GMDS2000. IOS Press. 2000.

[3] Göran Falkman and Olof Torgersson. *Knowledge Modeling and Management in Clinical Information Systems: A Case Study*. Knowledge Engineering and Knowledge Management: Ontologies and Semantics of the Web. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002). 2473. 96-101. Springer-Verlag. 2002.

[4] Göran Falkman and Olof Torgersson. *MedView: A Declarative Approach, to Evidence-Based Medicine*. Health Data in the Information Society. Proceedings of MEI2002, Studies in Health Technology and Informatics. 90. 577-581. IOS Press. 2002.

[5] Göran Falkman and Olof Torgersson. *Using Text Generation to Access Clinical Data in a Variety of Contexts*. Health Data in the Information Society. Proceedings of MEI2002, Studies in Health Technology and Informatics. 90. 460-465. IOS Press. 2002.

[6] Olof Torgersson. *Trädfiler i MedView*. http://www.cs.chalmers.se/proj/medview/developer/docs/trees.pdf September 13, 2004. .

[7] David Breneman, Cecilia Fu, Lisa Lagerstrand, Camilla Lindström, Anna Lindvall, Olsson, Yan, and Westlin. Copyright © 2003 Department of Computing Science, Chalmers University of Technology. *Intelligent Interfaces for Web-Based Clinical Trials*. http://medview.breneman.se/ September 13, 2004. .

[8] *XML: Principles, Tools, and Techniques*. O'Reilly & Associates, Inc.. 1085-2301. Dan Connolly. “A Guide to XML”. Norman Walsh. Copyright © 1997 ArborText, Inc.. 97-108.

[9] Harry R Tennant, Kenneth M Ross, Richard M Saenz, Craig W Thompson, and James R Miller. *Menu-Based Natural Language Understanding*. Proceedings of the 21st Annual Meeting of the Association for Computational Logistics. 151-158. Association for Computational Linguistics. Morristown, NJ, USA. 1983.

[10] Martin R Frank and Pedro Szekely. *Adaptive Forms: An Interaction Technique for Entering Structured Data*. University of Southern California - Information Sciences Institute. 4676 Admiralty Way, Marina del Rey, CA 90292 , USA. 1998.

[11] *IEEE Transactions on Software Engineering*. 23. 6. 0098-5589. June, 1997. “The Amulet environment: new models for effective user interface software development”. B A Myers, R G McDaniel, R C Miller, A S Ferrency, A Faulring, B D Kyle, A Mickish, A Klimovitski, and P Doane. Copyright © 1997 Sch. of Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA. 347-365.

[12] A Girgensohn, B Zimmermann, A Lee, B Burns, and M E Atwood. *Dynamic forms: An enhanced interaction abstraction based on forms*. Proceedings of Interact'95, Fifth IFIP Conference on Human-Computer Interaction. 362-367. Chapman & Hall. London, England. 1995.

[13] Andreas Girgensohn and Alison Lee. *Seamless Integration of Interactive Forms into the Web*. Proceedings for the Sixth International World Wide Web Conference. 619-629. 1997.

[14] Dianne K Hackborn and Cherri M Pancake. *Interactive HTML*. Department of Computer Science, Oregon State University. 1997.

[15] Paul T Thistlewaite and Steve Ball. *Active Forms*. Fifth International World Wide Web Conference. Cooperative Research Centre for Advanced Computational Systems of the Australian National University. 1996.

[16] Angel Puerta and Jacob Eisenstein. *XIML: A Common Representation for Interaction Data*. Proceedings A CM 1UI '01. 214-215. 2001.

[17] Micah Dubinko. *XForms Essentials*. 0596003692. O'Reilly Media, Inc. 2003.

[18] *PHP: Hypertext Preprocessor*. http://www.php.net/ . September 13, 2004.

[19] *MySQL Database*. http://www.mysql.org/ . September 13, 2004.

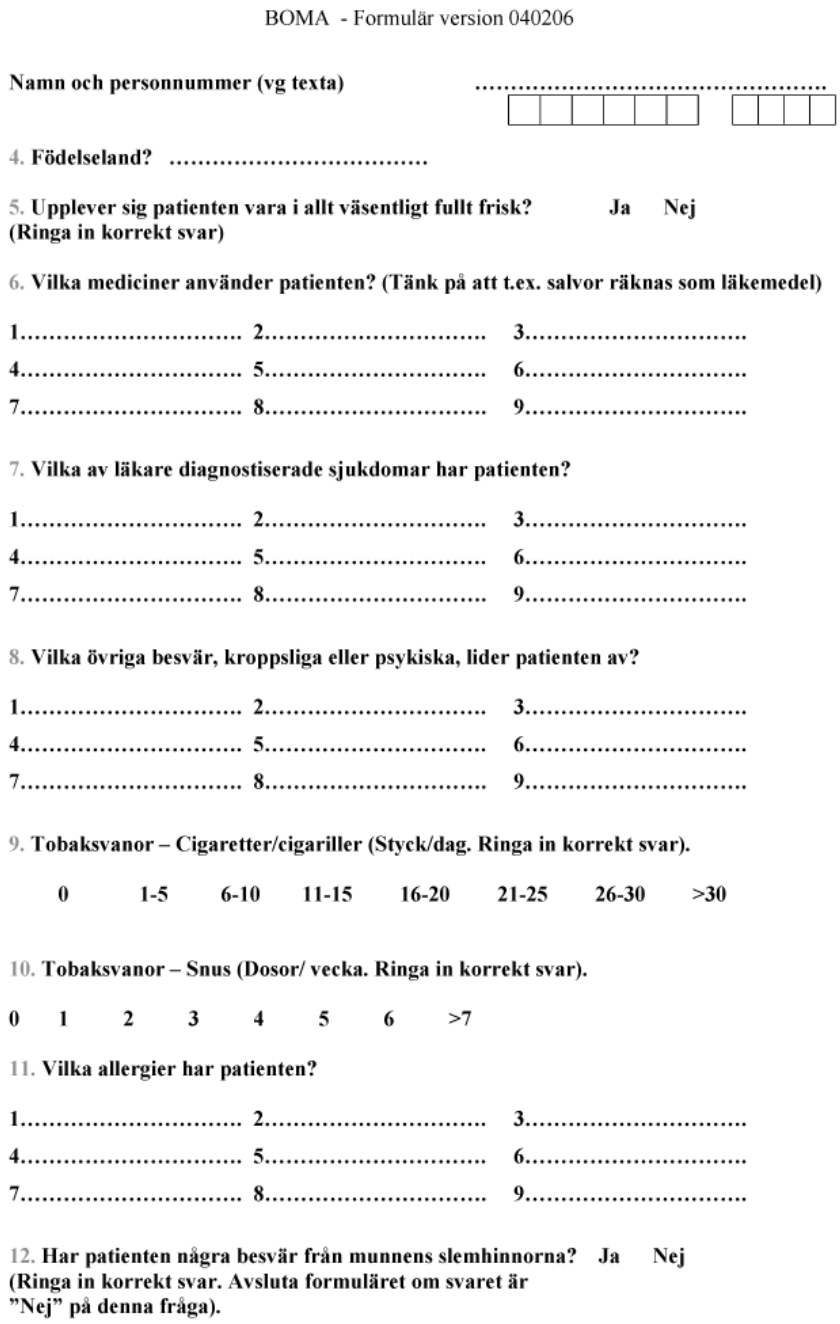
A. The MedForm Document Type Declaration

The following DTD can be used to validate all forms generated by MedForm, or to validate forms produced by an external source for use with MedForm.

```
<!ELEMENT EXAMINATION (FORMINFO, CATEGORY*)>
<!ELEMENT FORMINFO (AUTHOR, DATE, CONTACT, NOTICE, TITLE)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT CONTACT (ADDRESS, TELEPHONE, FAX, EMAIL)>
<!ELEMENT ADDRESS (STREET, POSTCODE, CITY, COUNTRY)>
<!ELEMENT STREET (#PCDATA)>
<!ELEMENT POSTCODE (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT COUNTRY (#PCDATA)>
<!ELEMENT TELEPHONE (#PCDATA)>
<!ELEMENT FAX (#PCDATA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ELEMENT NOTICE (#PCDATA)>
<!ELEMENT CATEGORY (NODE, HEADER, (INFO | INPUT)*)>
<!ELEMENT NODE (#PCDATA)>
<!ELEMENT HEADER (#PCDATA)>
<!ELEMENT INFO (SUBHEADER|TEXT|IMAGE)>
<!ELEMENT SUBHEADER (#PCDATA)>
<!ELEMENT TEXT (#PCDATA)>
<!ELEMENT IMAGE (#PCDATA)>
<!ELEMENT INPUT (TERM, DESCRIPTION, COMMENT, DEPENDENCY)>
<!ATTLIST INPUT freetext (true|false) "false">
<!ATTLIST INPUT translatable (true|false) "false">
<!ATTLIST INPUT required (true|false) "true">
<!ATTLIST INPUT visible (true|false) "true">
<!ATTLIST INPUT type (identification | text | note | single
  | multi | question | interval | vas | image) "text">
<!ELEMENT TERM (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT COMMENT (#PCDATA)>
<!ELEMENT DEPENDENCY (DEPRULE)*>
<!ELEMENT DEPRULE (DEPVAL, DEPTERM)>
<!ELEMENT DEPVAL (#PCDATA)>
<!ELEMENT DEPTERM (#PCDATA)>
```

B. Sample Form

This an actual paper form currently in use that has been used as a model for all the examples in this report.



BOMA - Formulär version 040206

Namn och personnummer (vg texta) ……

4. **Födelseland?** ……

5. **Upplever sig patienten vara i allt väsentligt fullt frisk? Ja Nej**
(Ringa in korrekt svar)

6. **Vilka mediciner använder patienten? (Tänk på att t.ex. salvor räknas som läkemedel)**

1…… **2**…… **3**……
4…… **5**…… **6**……
7…… **8**…… **9**……

7. **Vilka av läkare diagnostiserade sjukdomar har patienten?**

1…… **2**…… **3**……
4…… **5**…… **6**……
7…… **8**…… **9**……

8. **Vilka övriga besvär, kroppsliga eller psykiska, lider patienten av?**

1…… **2**…… **3**……
4…… **5**…… **6**……
7…… **8**…… **9**……

9. **Tobaksvanor – Cigaretter/cigariller (Styck/dag. Ringa in korrekt svar).**

0 1-5 6-10 11-15 16-20 21-25 26-30 >30

10. **Tobaksvanor – Snus (Dosor/ vecka. Ringa in korrekt svar).**

0 1 2 3 4 5 6 >7

11. **Vilka allergier har patienten?**

1…… **2**…… **3**……
4…… **5**…… **6**……
7…… **8**…… **9**……

12. **Har patienten några besvär från munnens slemhinnorna? Ja Nej**
(Ringa in korrekt svar. Avsluta formuläret om svaret är "Nej" på denna fråga).

Figure B.1. Borås Oral Medicine Academy, page 1.

BOMA - Formulär version 040206

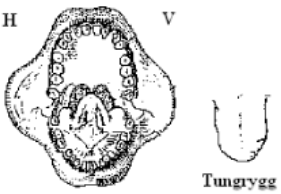
13. **Hur graderar patienten svårigheten av sina besvär från munnens slemhinnor? Sätt ett kryss på linjen.**

Inga besvär — **Värsta tänkbara besvär**

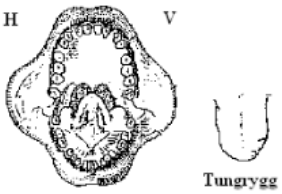
14. **Hur länge har patienten haft sina besvär (Ange dagar, veckor, månader eller år)**

.....

15. **I vilken lokalisation har patienten sina besvär?**



16. **I vilken lokalisation har patienten sina förändringar?**



17. **Vilken diagnos vill du ställa på patientens besvär/förändringar?**

.....

18. **Patientkod**

Ex HN00019510 (sista siffra 0 = kvinna, 1 = man)

Markera med 1 i den grå rutan vid tilläggsdiagnos. Fyll sedan i denna sida ännu en gång.

Kommentar

.....

.....

Figure B.2. Borås Oral Medicine Academy, page 2.

C. Sample MedForm Examination (XML)

This form as been used as the source in most of the examples througout the report.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE EXAMINATION SYSTEM "examination.dtd">
 <EXAMINATION>
  <FORMINFO>
   <AUTHOR>David Breneman</AUTHOR>
   <DATE>2004-09-07</DATE>
   <CONTACT>
    <ADDRESS>
     <STREET>Rännvägen 4</STREET>
     <POSTCODE>421 44</POSTCODE>
     <CITY>Göteborg</CITY>
     <COUNTRY>Sweden</COUNTRY>
    </ADDRESS>
    <TELEPHONE>0707-235829</TELEPHONE>
    <FAX>0707-235829</FAX>
    <EMAIL>david@breneman.se</EMAIL>
   </CONTACT>
   <NOTICE>Warning! Transmitted information is not
cryptographically secure.</NOTICE>
   <TITLE>Borås Oral Medicine Academy</TITLE>
  </FORMINFO>
  <CATEGORY>
   <NODE>Personal</NODE>
   <HEADER>Personal Information</HEADER>
   <INPUT type="text" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Name</TERM>
    <DESCRIPTION>Name</DESCRIPTION>
    <COMMENT>Enter your name</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="text" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>P-number</TERM>
    <DESCRIPTION>ID Number</DESCRIPTION>
    <COMMENT>(XXXXXX-XXXX)</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="single" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Born</TERM>
    <DESCRIPTION>Country of Birth</DESCRIPTION>
    <COMMENT>Choose one ...</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
  </CATEGORY>
  <CATEGORY>
   <NODE>Health</NODE>
   <HEADER>Health Status</HEADER>
   <INPUT type="single" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Health</TERM>
    <DESCRIPTION>Does the patient feel completely
healthy?</DESCRIPTION>
    <COMMENT>"Yes" or "No"</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="true" visible="true"
translatable="false" required="true">
```

```
    <TERM>Medication</TERM>
    <DESCRIPTION>Does the patient currently use any
medication?</DESCRIPTION>
    <COMMENT>Choose as many as necessary</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="true" visible="true"
translatable="false" required="true">
    <TERM>Diagnosis</TERM>
    <DESCRIPTION>Does the patient currently suffer from
any disease or illness?</DESCRIPTION>
    <COMMENT>Choose as many as necessary</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="true" visible="true"
translatable="false" required="true">
    <TERM>Disorder</TERM>
    <DESCRIPTION>Does the patient currently suffer from any other
physical or psychological disorders?</DESCRIPTION>
    <COMMENT>Choose as many as necessary</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
  </CATEGORY>
  <CATEGORY>
   <NODE>Tobacco</NODE>
   <HEADER>Tobacco Habits</HEADER>
   <INPUT type="single" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Smoke</TERM>
    <DESCRIPTION>How many cigarettes per day does the patient
smoke?</DESCRIPTION>
    <COMMENT>Choose the correct interval</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="single" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Snuff</TERM>
    <DESCRIPTION>How many packs of snuff does the patient use per
day?</DESCRIPTION>
    <COMMENT>Choose the correct number</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
  </CATEGORY>
  <CATEGORY>
   <NODE>Oral</NODE>
   <HEADER>Oral Status</HEADER>
   <INPUT type="single" freetext="false" visible="true"
translatable="false" required="true">
    <TERM>Mucous</TERM>
    <DESCRIPTION>Does the patient have any symptoms in the
oral mucosa?</DESCRIPTION>
    <COMMENT>"Yes" or "No"</COMMENT>
    <DEPENDENCY>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Status</DEPTERM>
     </DEPRULE>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Time</DEPTERM>
     </DEPRULE>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Symp-Site</DEPTERM>
```

```
     </DEPRULE>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Var-Site</DEPTERM>
     </DEPRULE>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Diag</DEPTERM>
     </DEPRULE>
     <DEPRULE>
      <DEPVAL>Yes</DEPVAL>
      <DEPTERM>Mucous-Image</DEPTERM>
     </DEPRULE>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="vas" freetext="false" visible="false"
translatable="false" required="true">
    <TERM>Mucous-Status</TERM>
    <DESCRIPTION>How severe are the patient's symptoms at
present?</DESCRIPTION>
    <COMMENT>(from "not at all troublesome" to "extremely
troublesome")</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="question" freetext="false" visible="false"
translatable="false" required="true">
    <TERM>Mucous-Time</TERM>
    <DESCRIPTION>For how long has the patient experienced
symptoms?</DESCRIPTION>
    <COMMENT>Choose a number and a unit of time</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="false" visible="false"
translatable="false" required="true">
    <TERM>Mucous-Symp-Site</TERM>
    <DESCRIPTION>In what localizations has patient experienced
symptoms?</DESCRIPTION>
    <COMMENT>Select the areas where symptoms have
occurred</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="false" visible="false"
translatable="false" required="true">
    <TERM>Mucous-Var-Site</TERM>
    <DESCRIPTION>Select the areas where variations have
occurred</DESCRIPTION>
    <COMMENT>Choose a n of time</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="multi" freetext="false" visible="false"
translatable="false" required="false">
    <TERM>Mucous-Diag</TERM>
    <DESCRIPTION>What oral diagnosis has the patient
recieved?</DESCRIPTION>
    <COMMENT>Descibe the diagnoses</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="image" freetext="false" visible="false"
translatable="false" required="false">
    <TERM>Mucous-Image</TERM>
    <DESCRIPTION>Add photographic documentation if
available</DESCRIPTION>
    <COMMENT>Click to upload</COMMENT>
    <DEPENDENCY>
```

```
    </DEPENDENCY>
   </INPUT>
  </CATEGORY>
  <CATEGORY>
   <NODE>Notes</NODE>
   <HEADER>Notes</HEADER>
   <INPUT type="identification" freetext="false"
visible="true" translatable="false" required="true">
    <TERM>P-Code</TERM>
    <DESCRIPTION>P-code</DESCRIPTION>
    <COMMENT>Enter the patients P-code (XXXXXXXXXX)</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
   <INPUT type="note" freetext="false" visible="true"
translatable="false" required="false">
    <TERM>Comment</TERM>
    <DESCRIPTION>Comment</DESCRIPTION>
    <COMMENT>Add a comment if necessary</COMMENT>
    <DEPENDENCY>
    </DEPENDENCY>
   </INPUT>
  </CATEGORY>
 </EXAMINATION>
```

D. Sample MedForm Examination (termValues)

The following termValues definition has been used for most of the examples in the report.

```
$Name

$P-number

$Born
Sweden
Finland
Denmark
Norway
Iceland
Africa
Asia
South America
X ...

$Health
Yes
No

$Medication
Abbopen
Abboticin
Aberala
Absenor
Accupro
Accupro Comp
Acetylcystein
Achromycin
Aciloc
Acinil
Acipen Solutab
Actisite
Activell
Actrapid
Adalat
Adalat Oros
Adrenalin Medihaler
X ...

$Diagnosis
Agranulocytos
AIDS
Acne
Aldosteronism
Alcoholism
Aneurysm
Angina pectoris
Anorexia nervosa
Aplastisk anemi
Atros
Basaliom
Bulimia nervosa
Cancer Thyreoidea
Celiaci
X ...

$Allergy
Al
Banana
Citrus
Dust
Epoxy
Feathers
Fish
```

```
Formaldehyde
Gold
Kiwi
Latex
X ...

$Disorder
Depression
Muscle aches
Panic
Tinnitus
Tiredness
Viral Infection
Whiplash
X ...

$Smoke
0
1-5
6-10
11-15
16-20
21-25
26-30
>30

$Snuff
0
1
2
3
4
5
6
>7

$Mucous
Yes
No

$Mucous-Status

$Mucous-Time
Recently
? days
? weeks
? months
? years

$Mucous-Symp-Site
Bottom of Tongue
Top of Tongue
Lips (inside)
Lips (outside)
Entire Mouth
X ...

$Mucous-Var-Site
Bottom of Tongue
Top of Tongue
Lips (inside)
Lips (outside)
Entire Mouth
X ...

$Mucous-Diag
Burning mouth syndrome
Decubitus
Hairy Leukoplakia
```

```
Hairy Tongue
Herpangina
Herpes Labialis
Herpes Zoster
Lichen Planus
Nevus
Ulcer
X ...

$Mucous-Image

$P-Code

$Comment
```

E. Sample MedForm Output (Tree file)

The following tree file has been generated as a result of the input demonstrated in most of the examples in the report.

```
user_files/fs2/treefiles/oralmedicine.mvd/Forest.forest/XX1234567890_040910192052.tree


040910192052
NKonkret_identifikation
LXX1234567890_040910192052.tree##
NDatum
L2004-09-10 19:20:52##
NPersonal
NName
LJohn Doe##
NP-number
L601219-1234##
NBorn
LSweden##
NHealth
NHealth
LNo##
NMedication
LAbboticin#
LAbsenor#
LActivell##
NDiagnosis
LAneurysm#
LCeliaci##
NDisorder
LMuscle aches#
LStiff Joints#
LTiredness##
NTobacco
NSmoke
L0##
NSnuff
L1##
NOral
NMucous
LYes##
NMucous-Status
L44##
NMucous-Time
L3 days##
NMucous-Symp-Site
LTop of Tongue#
LBottom of Tongue#
LLips (inside)##
NMucous-Var-Site
LBottom of Tongue#
LTop of Tongue##
NMucous-Diag
LDecubitus#
LHairy Leukoplakia#
LHerpes Labialis##
NMucous-Image
Loralmedicine.mvd/Pictures/Pictures/040910182541_G03753.jpg#
Loralmedicine.mvd/Pictures/Pictures/040910182559_M04392.jpg#
Loralmedicine.mvd/Pictures/Pictures/040910182729_M04091.jpg#
Loralmedicine.mvd/Pictures/Pictures/040910182742_M04621.jpg##
NNotes
NP-Code
LXX1234567890##
NComment
LThe patient has additional documentation which will be added later.##
```