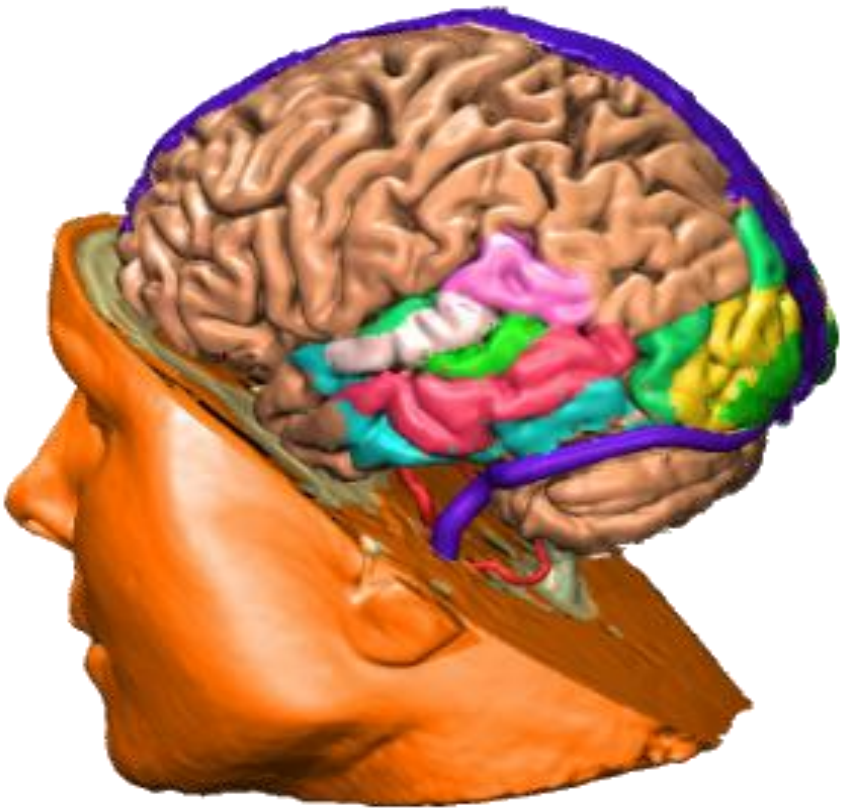
DESIGN AND DEVELOPMENT OF MERRYTS

A TOOL FOR VISUALIZATION OF MEDICAL TERMINOLOGY SYSTEMS



W.P.M.E. Hofland
Medical Informatics
AMC-UvA

Design and development of MERRYts:

A tool for visualization of Medical Terminology Systems

W.P.M.E. Hofland
5993032
AMC Amsterdam
Meibergdreef 9
1122AZ Amsterdam

January-May 2011

Supervisor: Dr. N. F. Keizer, Associate professor, Medical Informatics

Daily Supervisor: Dr. ir. R. Cornet, Assistant Professor, Medical Informatics

ABSTRACT

Structuring information in EHR[1] systems is essential to enable communication between various HISs and between the users of these systems. Data values captured in EHR systems can be structured and coded by using TSs.

When recording patient information, the medical professional needs to be supported in matching the concept he has in mind with a concept in the TS. There has been little evidence on the best way to provide this support.

A literature search resulted in some articles with basic advice for web based search-improvement, but no hard conclusions on the best way of presenting large collections of information could be drawn. Based on the literature requirements are defined for a tool that enables investigation of different ways of visualization of TS content. This paper describes the design and development of MERRYts, a tool to enable usability evaluation of several back- and frontends and API-functions to get evidence-based information on the best way to support TS-based recording of patient information.

MERRYts is a web based tool. It is built using server side scripting to ensure it can be accessed by various devices. MERRYts uses an existing SNOMED CT-API (SnAPI) and is built using Java, JavaScript and PHP to make it easy to create (3rd party) extensions.

We have implemented several frontends that can be used separately for use in a usability-study. There are various ways of retrieving knowledge from MERRYts, for example output on screen or direct communication with 3rd-party products such as EHRs.

At this moment one TS is implemented, SNOMED CT. This is done through the SnAPI.

New functions in the front- and backend, functions for (extra) data-retrieval, export-options and (usage-) statistics can be added in a relatively simple way because the API of MERRYts enables building an extension that can make use of all functions already built. This creates the possibility to build a custom environment for a particular case (e.g. to test a new GUI against the GUI used in the current EHR).

The number of functions is currently limited. There is however the possibility of adding extra API's to connect MERRYts with other TS or create new functions by building extensions. Also a good mobile-frontend and some EHR-specific bridges for direct data-input into the EHR should be created.

MERRYts is the start for investigating the usability of various GUI/backend combinations. More modules have to be built to get more ways of presentation, but this is a good first start for further research.

Keywords: Medical Informatics, Terminology System, Visualization, Research, Webbased Programming, Research on Visualization of Terminology Systems

[1] See the definitions section for used terms and abbreviations.

TABLE OF CONTENTS

Abstract....3
Introduction....5
Definitions....5
Goal of the Project....6
Methods....6
Literature research....6
Requirements analysis....6
Pilot implementation....7
Results....7
Literature research....7
Requirements analysis....8
Implementation....8
Discussion....9
Conclusion....10
ACKNOWLEDGEMENTS....10
References....11
Appendix A – MERRYts API....12
Package nl.mikproject.MERRYts....12
Class DragDrop....12
Class ResultTemplates....13
Class SearchByID....16
Class SearchByText....18
Class Servlet....20
Class Settings....21
Class SubSetHandling....21
Class RequestHandler....22
Package nl.mikproject.MERRYts.Support....24
Class ShortCuts....25
Appendix B – Ways of information-presentation in MERRYts....26
B.1 Plain text....26
B.2 Table....27
B.3 List of drag and drop labels....28
B.4 CliniClue-like skin....29
B.5 SNOB-like skin....31
B.6 The administration-section....32
B.7 List of images....33

INTRODUCTION

In the past decades the role of the computer in patient record keeping has increased and gradually replaced free-text paper record keeping by electronic free-text information storage.
Now that EHR usage is more common, one wants to profit from the possibilities EHRs bring such as reusing electronic data for research, management information and decision support. This requires better communication, interoperability, between various EHR-systems. To communicate it is important to preserve context and meaning of the data captured. In healthcare, TSs are used for electronic data capturing in a structured and standardized way. These TSs interrelate concepts and make use of codes to identify concepts and terms to describe these concepts. Because there are a lot of TSs and the number of terms can vary a lot (up to several hundred of thousand items) it is important to have a good way of presenting these items in a clear way to the users who have to capture and use the data.

There is a lot of research on usability of software and GUIs. Also, there is a lot of research on searching and search engines. However, research combining these two areas (research on GUI of search engines) is not often performed. This implies that there is not much research available on the best way of presenting large sets of information, like concepts, in such a way that a concept can be most efficient found. Research that is available is often specified on one specific situation(1,2).

DEFINITIONS

Terms and abbreviations as they are used in this thesis

API	Application Programming Interface	**OS**	Operating System
Backend	Part of application processing functions. Backend is not directly accessible by the GUI.	**Servlet**	Java-application running on webserver
EHR	Electronic Health Record	**Skin**	Custom created GUI available in, and only useable by, MERRYts
(G)UI	(Graphical) User Interface	**SNOMED CT**	Systematized Nomenclature of Medicine-Clinical Terms
HIS	Hospital Information System	**Terminology System**	A system which organizes the terms of a domain according to some structure, such as hierarchical indentations, codification or lookup tables.(3)
JRE	Java Runtime Environment	**TS**	Terminology System
MERRYts	Medical Electronic Recording-support for Research-based Yammering about Terminology Search		

GOAL OF THE PROJECT

The goal of this project is to build a customizable Terminology search tool, called MERRYts that can be used to present terminology information in a variety of ways. Using MERRYts would give the possibility to make a usability study of various user interfaces for browsing medical TSs. In the end, the results of that study can be used to create a generic user interface for terminology systems with a high degree of acceptance among physicians.

METHODS

This project is performed in three phases:

- Literature research
- Requirements analysis
- Pilot implementation

LITERATURE RESEARCH

Literature was searched using the Internet. Goal was to find publications on usability of (medical) search-engines and on usability of medical TSs. These searched where performed on Google, Google Scholar, Google Books, Pubmed.

Article selection was based on the following MeSH- and key terms:

- Medical Terminology GUI
- Medical Terminology Search
- Search Engine Usability
- Medical Terminology Usability
- Terminology Search Usability
- Terminology Search Engine
- Presentation of Medical Terminology

Another search was performed on the library catalog of the University of Amsterdam. Search-actions where restricted on books located at the medical library and library for beta sciences.

Recommendations of colleagues and suggestion from authors of retrieved papers were taken into account to further extend the set of retrieved papers.

REQUIREMENTS ANALYSIS

Because this is, as far as we know, the first research on the presentation of large sets of data, there are no relevant requirements of other studies found. Apart of common GUI requirements like to put often used functions in main screen. Knowing this, requirements are based on the fact the author wanted to have a research tool with a very high flexibility, regarding functionality, OS support and ‘look and feel’.

The main focus of the requirements analysis was on functionality of mobile devices. MERRYts should be accessible by most devices with an internet connection. Mobile devices lack support for various programming languages. If a web application works on a mobile device, it will almost always also run on personal computers. While there was a desire to build an 'open' tool, e.g. no restrictions in how data is presented to the user, there has been no requirements investigation among future users. The requirements have been based on the idea of creating an API. In this way, it is possible to customize the tool for future research with different front-ends, TS or even complete 3rd-party programs where MERRYts would only be used as connection to different TSs.

PILOT IMPLEMENTATION

Because the project result is the creation of a tool to be used in usability studies on TSs we focused on functions instead of implementation of a definitive product with all functions present in a GUI.

Because it should be easy for others to create extensions for the project and the project needs to be accessible by different devices.
It is desirable to make the tool web-based. When using Servlets, the user does not need to have a JRE, this makes the product available for some new devices that lack Java-support (like E-readers, some tablets and smart phones).

This resulted in the point of few MERRYts should make use of a few frameworks and libraries:

- The tool is built in 4 combined languages: PHP, HTML, JavaScript and Java.
- Libraries used are the standard libraries for these languages. For Java, the Servlets library is added.

RESULTS

LITERATURE RESEARCH

The literature research resulted in a total of 22 pieces of writing looking useful at first sight. After reading, most of them could be excluded. Main reason was the fact that information has become obsolete because of the evolution in computer technology. A second reason was information in the article did not match the subject, which was assumed based on the title.

A total of 4 useful articles(1,3-5) where included. One article(2) was added after contact with one of the authors.

One master thesis(6) and a guideline(7) where added based on recommendations of the daily supervisor.

One book(8) was included.

The literature made clear that user interfaces of search tools have changed a lot since the first usage of search tools on the computer. Most changes are based on the evolution of the computer.

This evolution is the reason why search tools like web based search engines can more and more focus on performance and graphical user interfaces.

The literature also pointed out that various TS had different UI's. There is, at the moment, no golden rule or best practice of presenting this information. After reading the literature, one main conclusion was drawn from all: "There has been no academic research of any importance on this subject."

A search on the Internet resulted in one API to access the knowledge stored in SNOMED CT(9) that was useful, i.e., free, actively developed, flexible. This API is the most important component of MERRYts at this moment. All requests on SNOMED CT TS are passed through this API.

REQUIREMENTS ANALYSIS

The requirements analysis resulted in a few requirements:

- There had to be an easy way to query SNOMED CT.
- To test the GUI, MERRYts needs to be a 'universal' tool. This implies that it should not depend on environmental variables like be OS, used software, resolution, etc. Also, it should be customizable to be able to create a frontend-backend combination according to the desired interface to test.
- The fact that the program needed to work independent of the environment resulted in the wish to build the tool online. To make sure that the tool works on various systems, also mobile devices, no user side java should be necessarily. As the used SnAPI is Java-based, this results in another requirement. To build the functions for this tool, there is a need for Java; the server needs to support this language.
- To create an easy way of adding functions / GUIs / skins / …, the used tool should be able to work with external extensions.

All mentioned requirements resulted in the choice of PHP and Java. These are commonly used languages so it should ease the creation of extensions.

IMPLEMENTATION

MERRYts has become a web tool with an open structure. This means that functions can be used in combination with external programs or extensions. You can use the browser to find the concepts you like, in the UI you like. It is relatively easy to add own GUIs and data-sources.

The only implemented TS at this moment is SNOMED CT. This is done through the SnAPI. The SnAPI is an API with a lot of functions. It is a Java-API and is free to use.

Two main functions are implemented:

- Searching SNOMED CT based on a string (possible search-methods are "whole word" or "Phrase search")
- Listing implemented subsets of concepts from SNOMED CT

Both functions can be used in five different implemented skins, each based on one of the following principles, returning information[2] in:

1. plain text
2. a table
3. a list of 'drag and drop' labels and boxes for (to be implemented) post-coordination
4. a CliniClue-like(10) skin
5. a SNOB-like(11) skin

MERRYts can communicate with other programs like EHRs using the API[3] of the tool itself. In theory every GUI / TS can be connected to MERRYts as long as it is able to communicate with Java-based software.

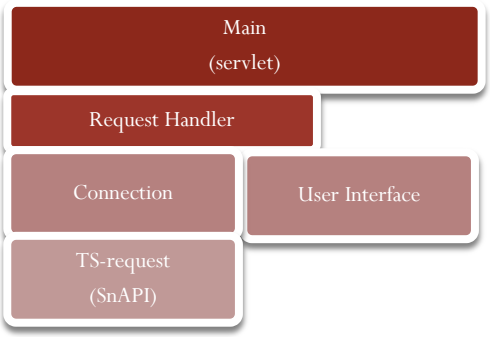


FIGURE 1 COMPONENT HIERARCHY IN MERRYTS

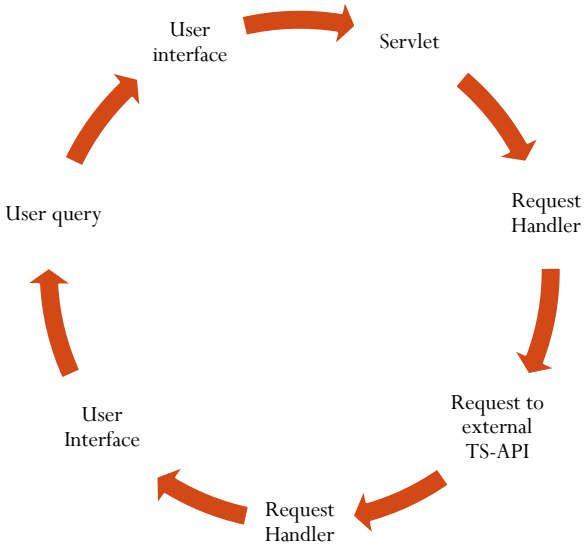


FIGURE 2 FLOW DURING A REQUEST

DISCUSSION

Although the computer is more and more used in Medical Health Record Keeping and TS are more and more used to capture data in a structured way, there has not yet been good research on usability of Graphical User Interfaces of TSs.

MERRYts is a product that makes it possible to research usability of search GUIs, with a focus on medical Terminology Systems. MERRYts is highly configurable. It can be completely customized to fit in the GUI research project, in terms of search-possibilities, look and feel, and searchable TS.

[2] Numbers refer to subsections in appendix B. These subsections contain screenshots and additional information.
[3] See Appendix A for API-definition

The decision was made not to ask users about desired functions or good / bad current products. Reason for this decision was the idea that no info would give an 'open view' to the problem. This tool should also be able to test new GUIs or GUIs that would otherwise not go into production because, for example, the perception is that some GUIs would not work, based on own experience of the programmer.

MERRYts is built and tested by the author. It has not been tested by others yet. So although it seems to work, it is possible that functions are tested in a complete different way than end-users would do. It is assumed that the open structure gives the opportunity in this case to write an extension that changes the behavior in the desired way.

The biggest advantage of MERRYts is that it enables usability research to support decision on the way TSs should be implemented in an EHR / HIS. Now there is a tool to compare TS GUIs on performance (best registration-results) or usability (user friendly interface).

The programmatic structure of MERRYts enables the possibility of extending the program by 3rd-party extensions. The possibility of extending MERRYts is mainly limited by the skills of the extension-programmer. A second limitation could be the specifications of the used server or network-overhead during usage. But this is only limiting if an extension is high on resource usage.

A next step would be to test the current version of MERRYts on usefulness. That means that we have to test whether it is possible to get significant results on GUI testing by using this tool. After that, some extra TS should be implemented and a pilot-test of GUI comparison using MERRYts should be started.

CONCLUSION

MERRYts is a web tool that is able to present and search TS in various ways. The tool can retrieve input and present the content of TSs in various ways, by using the API.
The tool is a basis to be used for research on usability of various TS-Search-GUI.

ACKNOWLEDGEMENTS

I am very grateful for the interest and contribution to the study of the supervisors, students, teachers, colleagues and authors that I've had contact with during the study. Special thanks goes to all those people out of sight for providing an optimal work environment.

REFERENCES

(1) Buzzi M, Andronico P, Leporini B. Accessibility and usability of search engines: a preliminary study. UI4ALL 2004 28-29/06/2004;1.

(2) Leporini B, Andronico P, Buzzi M, Castillo C. Evaluating a modified Google user interface via screen reader. UAIS 2008;7(3):155-175.

(3) Pisanelli DM, Gangemi A, Steve G. WWW-available Conceptual Integration of Medical Terminologies: the ONIONS Experience. AMIA 1997;Fall Symposium.

(4) Baeza-Yates R, Ribeiro-Neto B. Modern information retrieval. Beijing, China: Beijing China Machine Press; 2004.

(5) Ide NC,MS., Loane RF,PhD., Demner-Fushman D,MD. PhD. Essie: A Concept-based Search Engine for Structured Biomedical Text. JAMIA 2007;14(3):253-263.

(6) Marlijne Dorrepaal. Usability evaluation of an Interface Terminology on SNOMED CT for the Intensive Care. Amsterdam: University of Amsterdam, Medical Informatics; 2009.

(7) Microsoft. Design Guidance: Terminology matching, Version 1.0.0.0. 2007 5 July 2007.

(8) Haux R, Winter A, Ammenwerth E, Birgl B. Strategic Information Management in Hospitals. ; 2003. p. 151-176.

(9) Dataline Software. SnAPI™, your applications window into Snomed CT. 2011; Available at: http://snomed.dataline.co.uk/snapi.htm. Accessed februari, 2011.

(10) The Clinical Information Consultancy Ltd. CliniClue® Xplore. 2010 23 August 2010.

(11) eggbird. SNOB. 2011;3.20.04.

APPENDIX A – MERRYTS API

Package nl.mikproject.MERRYts

DragDrop
ResultTemplates
RequestHandler
SearchByID
SearchByText
Servlet
Settings
SubSetHandling

nl.mikproject.MERRYts

Class DragDrop

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.DragDrop
            public class DragDrop
                  extends java.lang.Object
```

Author:
WPME Hofland

DragDrop

```
public DragDrop()
```

Method for building a drag/drop template

showSite

```
public java.lang.String showSite(java.util.ArrayList
listOfDragItems, int locationOfTitle, int labelposition, int
locationOfID, boolean multipleDropboxes)
```

Method to get a html-string to present a drag-drop site in a browser

Parameters:
`listOfDragItems` - Items that should be in labels to drag/drop
`locationOfTitle` - The index of the title to show on the label
`labelposition` - The index of the label of the boxes
`locationOfID` - Location of the ID in the listOfDragItems
`multipleDropboxes` - True if multiple boxes should be shown. False if one is shown (for example when presenting subsets)

Returns:
HTML-string to present in browser

nl.mikproject.MERRYts

Class ResultTemplates

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.ResultTemplates
            public class ResultTemplates
                  extends java.lang.Object
```

Author:
WPME Hofland

ResultTemplates

```
public ResultTemplates()
```

buildCliniClueLikeResponse

```
public java.lang.String
buildCliniClueLikeResponse(java.util.ArrayList infoImported,
boolean isTextSearch)
```

Build a skin presenting the information in a CliniClue-like environment

Parameters:
`infoImported` - ArrayList containing subsets or categories. See the SearchByText Class for information about the order of strings
`isTextSearch` - True if infoImported contains labels and categories. False if it contains subsets

Returns:
HTML-string Presenting infoImported in a CliniClue-like environment

buildColorTableResponse

```
public java.lang.String
buildColorTableResponse(java.util.ArrayList infoImported,
boolean isTextSearch)
```

Prepare a template containing a Table

Parameters:
`infoImported` - An arraylist containing information fetched from a TS. See the SearchByText Class for information about the order of strings
`isTextSearch` - True if infoImported is an array of retrieved terms and various categories. False if it is a list of subsets.

Returns:
Html-string containing send information. Formatted in a table highlighted in various colors.

buildDefaultColorTableResponse

```
public java.lang.String
buildDefaultColorTableResponse(java.util.ArrayList infoImported)
```

Create a table response using colors

Parameters:

`infoImported` - an arraylist of terminological information. See the SearchByText Class for information about the order of strings

Returns:

HTML-string containing send information. Formatted in a table highlighted in various colors.

buildDefaultTableResponse

```
public java.lang.String
buildDefaultTableResponse(java.util.ArrayList infoImported,
boolean useColor)
```

Build a table response with or without colors. Used to create tables for subsets

Parameters:

`infoImported` - An arraylist containing subsets.
`useColor` - True if the table needs to use an alternate color to present the ID.

Returns:

HTML-string presenting a table with information set by infoImported

buildDragDropSite

```
public java.lang.String buildDragDropSite(java.util.ArrayList
listOfDragItems, int type, java.lang.String searchText)
```

Builds a template presenting data in a drag and drop environment used to create functions like post-coordination

Parameters:

`listOfDragItems` - Arraylist of items. Positions needs to be formatted in this way: (**index = content(type=1)/content(type=2)**) 0 = label/conceptID, 1 = title of target box/-, 2 = conceptID/title of target box, 3 = -/label
`type` - type of presentation of target-boxes. 1 = use 1 target-box. 2 = use multiple target boxes. Default is 1.
searchText - Text used to fetch items found in listOfDragItems

Returns:

HTML-string Presenting listOfDragItems in a DragDrop environment

buildPlainResponse

```
public java.lang.String buildPlainResponse(java.util.ArrayList
information, boolean isTextSearch)
```

Returns a string of all information set by information

Parameters:
`information` - Arraylist presenting data.
`isTextSearch` - True if information contains concepts and categories, False if information contains subsets

Returns:
A string presenting al data. Information on one item is separated by a tab \\t. Items are separated by a return \\n

buildSnobLikeResponse

```
public java.lang.String
buildSnobLikeResponse(java.util.ArrayList infoImported, boolean
isTextSearch)
```

Create a template presenting information in a SNOB-like environment

Parameters:
`infoImported` - An arraylist with data to present. See the SearchByText Class for information about the order of strings
`isTextSearch` - True if infoImported contains concepts and categories. False if it contains subsets

Returns:
HTML-string presenting the SNOB-like environment

buildTextColorTableResponse

```
public java.lang.String
buildTextColorTableResponse(java.util.ArrayList infoImported)
```

Build a table response with colors. Used to create tables with IDs, concepts, categories and weight

Parameters:
`infoImported` - An arraylist containing subsets.

Returns:
HTML-string presenting a table with information set by infoImported

buildTextColorTableResponse

```
public java.lang.String
buildTextColorTableResponse(java.util.ArrayList infoImported,
boolean useColor)
```

Build a table response with colors. Used to create tables with IDs, concepts, categories and weight

Parameters:
`infoImported` - An arraylist containing subsets.
`useColor` - True if the table needs to be presented using colors for various types of information

Returns:
HTML-string presenting a table with information set by infoImported

setIDLocation

```
public void setIDLocation(int locationOfIDInArray)
```

Sets the location of the ID in an Array. If this option is set, the table column with the ID will turn green

Parameters:

`locationOfIDInArray` - Location of ID in array. -1 if no ID available. Default: -1.

setResponseType

```
protected void setResponseType(int desiredResponse)
```

Store type of desired Response

Parameters:

`desiredResponse` - value representing the skin to be stored

---

nl.mikproject.MERRYts

Class SearchByID

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.SearchByID
            public class SearchByID
                  extends java.lang.Object
```

Author:

WPME Hofland

SearchByID

```
public SearchByID()
```

Class containing functions to use on searching by ID.

getConceptByID

```
public java.lang.String getConceptByID(java.lang.Long ID)
```

Get name of concept based on ID

Parameters:

`ID` - ID of concept to display

Returns:

Name of concept

getConceptDescription

```
public java.lang.String getConceptDescription(java.lang.Long
ID, java.lang.String separator)
```

Get description of a concept

Parameters:

`ID` - ID of the concept to search for

`separator` - separator to use for separating various part (like \\n or "<"br">")

Returns:
String containing various parts of description separated by separator

getNeighbourIDsOfConcept

```
public java.util.ArrayList
getNeighbourIDsOfConcept(java.lang.Long ID)
```

Get neighbors ID's of a concept

Parameters:
`ID` - ID of concept you want neighbour ID's to be displayed

Returns:
Arraylist of neighbourIDs

getNeighbourIDsOfConcept

```
public java.lang.String getNeighbourIDsOfConcept(java.lang.Long
ID, java.lang.String separator)
```

Get IDS of Neighbour concepts separated by separator

Parameters:
`ID` - ID you want to see neighbours of
`separator` - string to separate neighbours with

Returns:
String of neighbours separated by separator

getNeighboursOfConcept

```
public java.lang.String getNeighboursOfConcept(java.lang.Long
ID, java.lang.String separator)
```

Get neighbours of concept

Parameters:
`ID` - ID to get neighbours of
`separator` - string to separate neighbours

Returns:
ID's of the neighbours given by ID, separated by separator

getNeighboursOfConceptByID

```
public java.lang.String
getNeighboursOfConceptByID(java.lang.Long ID, java.lang.String
separator)
```

Get names of the neighbours of concept given by ID

Parameters:
`ID` - ID of concept you want to show neighbours of
`separator` - String to separate various neighbours

Returns:
String containing neighbour names, separated by separator

---

nl.mikproject.MERRYts

Class SearchByText

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.SearchByText
            All Implemented Interfaces:
                  java.io.Serializable
            public class SearchByText
                  extends java.lang.Object
                  implements java.io.Serializable
```

Author:
WPME Hofland

SearchByText

```
public SearchByText()
```

Class containing methods to communicate with the external SnAPI

getInfo

```
public java.util.ArrayList getInfo()
```

Returned information arraylist contains:

0 ConceptID
1 Fully Specified Name
2 Fully specified Name minus Type
3 Type of concept
4 Weight (likelihood this is the item searched for)
5 CTV3ID

Returns:
An arraylist containing requested information

getListOfDescriptions

```
public static
org.datacontract.schemas._2004._07.snapiwcfservice.ArrayOfDescr
iption[] getListOfDescriptions()
```

Get a list of descriptions of concepts

Returns:
A list of Descriptions of concepts

getSearchText

`public java.lang.String` **`getSearchText`**`()`

Get the current stored search text

Returns:

Current stored search text

setMatchtype

`public void` **`setMatchtype`**`(int type)`

Set the match-type to use

Parameters:

`type` – 'match type'. 0 = Exact Word, 1 = Phrase Search

setMaxResults

`public void` **`setMaxResults`**`(int maxResults)`

Set the maximum results to show

Parameters:

`maxResults` - maximum number of results to show

setResponsType

`public void` **`setResponsType`**`(int response)`

Set the type of response (template) you want to get

Parameters:

`response` - type of response.

setSearchText

`public void` **`setSearchText`**`(java.lang.String searchText)`

Set the string to use for searching

Parameters:

`searchText` - string to search for

nl.mikproject.MERRYts

Class Servlet

```
java.lang.Object
      |
      +--HttpServlet
            |
            +--nl.mikproject.MERRYts.Servlet
                  public class Servlet
                        extends HttpServlet
```

Author:
WPME Hofland

Servlet

```
public Servlet()
```

doGet

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws null, java.io.IOException
Handles the HTTP GET method.
```

Parameters:
`request` - servlet request
`response` - servlet response

Throws:
`null` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

doPost

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws null, java.io.IOException
Handles the HTTP POST method.
```

Parameters:
`request` - servlet request
`response` - servlet response

Throws:
`null` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

getServletInfo

```
public java.lang.String getServletInfo()
```

Returns a short description of the servlet.

Returns:
A string containing servlet description

processRequest

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws null, java.io.IOException
```

Default method called when site is requested

Parameters:

`request` - http-requests
`response` - http-response

Throws:

`null` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

---

nl.mikproject.MERRYts

Class Settings

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.Settings
            public class Settings
                  extends java.lang.Object
```

Author:

WPME Hofland

Settings

```
protected Settings()
```

Class used to get global settings

getGuid

```
protected static java.lang.String getGuid()
```

ID for the SnAPI

Returns:

SnAPI GUID

---

nl.mikproject.MERRYts

Class SubSetHandling

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.SubSetHandling
            public class SubSetHandling
                  extends java.lang.Object
```

Author:
WPME Hofland

SubSetHandling

```
public SubSetHandling()
```

Class presenting functions used for working with subsets

getInfo

```
public java.util.ArrayList getInfo()
```

Get an arraylist of subsets available in the SnAPI

Returns:
Arraylist of subsets available in the SnAPI

---

nl.mikproject.MERRYts

Class RequestHandler

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.RequestHandler
            public class RequestHandler
                  extends java.lang.Object
```

Author:
WPME Hofland

RequestHandler

```
public RequestHandler()
```

Basic Class for calling the right template, starting the right request function, etc.

buildResponse

```
public void buildResponse(java.util.ArrayList information)
```

Method to fit found information into a template

Parameters:
`information` - List with data to present in a template

getConceptByText

```
public java.lang.String getConceptByText()
```

Get concepts based on text set in the setSearchKey method

Returns:
HTML-formatted response

getConceptByText

```
public java.lang.String getConceptByText(java.lang.String text)
```

Get a formatted response based on searched text

Parameters:
`text` - text to search for

Returns:
HTML-formatted response

getConceptByText

```
public java.lang.String getConceptByText(java.lang.String text,
int desiredResponse)
```

Get concept based on searched text, formatted as desired

Parameters:
`text` - text to search for
`desiredResponse` - response desired

Returns:
HTML-string presenting concepts searched by text

getConceptByText

```
public java.lang.String getConceptByText(java.lang.String text,
int desiredResponse, int searchType)
```

Get a HTML-formatted string containing concepts.

Parameters:
`text` - text to search
`desiredResponse` - type of response (template)
`searchType` - Type of search. 0 = whole word, 1 = phrase search

Returns:
HTML-formatted string containing concepts fetched by searching on text by searchType. Formatted as set in desiredResponse.

getListOfSubsets

```
public java.lang.String getListOfSubsets()
```

Get a list of subsets implemented in the SnAPI

Returns:
String containing a list of available subsets

getListOfSubsets

```
public java.lang.String getListOfSubsets(int ResponseType)
```

Return a list of subsets based presented in various ways

Parameters:
`ResponseType` - type of presentation

Returns:
HTML-string containing list of subsets available in the SnAPI

setIDLocation

```
public void setIDLocation(int locationOfIDInArray)
```

Sets the location of the ID in an Array. If this option is set, the table column with the ID will turn green

Parameters:

`locationOfIDInArray` - Location of ID in array. -1 if no ID available. Default: -1.

setNumberOfResults

```
public void setNumberOfResults(int results)
```

Set the maximum number of results to be fetched

Parameters:

`results` - maximum number to be fetched

setResponseType

```
public void setResponseType(int desiredResponse)
```

Set the type of response (template) you want to get

Parameters:

`desiredResponse` - type of response you want. 0 = Plain text. 1 = Colored Table. 2 = Snob like. 3 = DragDrop. 4 = CliniClue like. Default = 1. On out of bound, default = 0.

setSearchKey

```
public void setSearchKey(java.lang.String text)
```

Set the term to search for

Parameters:

`text` - text to search for

setSearchMode

```
public void setSearchMode(int type)
```

Set the search mode for text based search. See SearchByText.setMatchType(int type) for available options.

Parameters:

`type` - type of search used

Package nl.mikproject.MERRYts.Support

ShortCuts

nl.mikproject.MERRYts.Support

Class ShortCuts

```
java.lang.Object
      |
      +--nl.mikproject.MERRYts.Support.ShortCuts
            public class ShortCuts
                  extends java.lang.Object
```

Author:
WPME Hofland

ShortCuts

```
public ShortCuts()
```

Class presenting some common used tasks

findBiggestInt

```
public static int findBiggestInt(int a, int b)
```

Get the biggest integer out of two integers

Parameters:
`a` - Integer A
`b` - Integer B

Returns:
Biggest integer of A and B. If A and B are equal, B is returned

APPENDIX B – WAYS OF INFORMATION-PRESENTATION IN MERRYTS

This appendix shows the various ways of output available in MERRYts. When a screenshot of a search-output is demonstrated, the term Diabetes is used. A limitation of returning 100 terms is set. The search-method is set on “Phrase Search”, meaning a term matches when the searched string is available as a (part) of the term.

B.1 PLAIN TEXT

The plain text modus is used as a demonstration modus. Output in this mode is a presentation of data that as it is returned when you use the API to retrieve information with an extension.

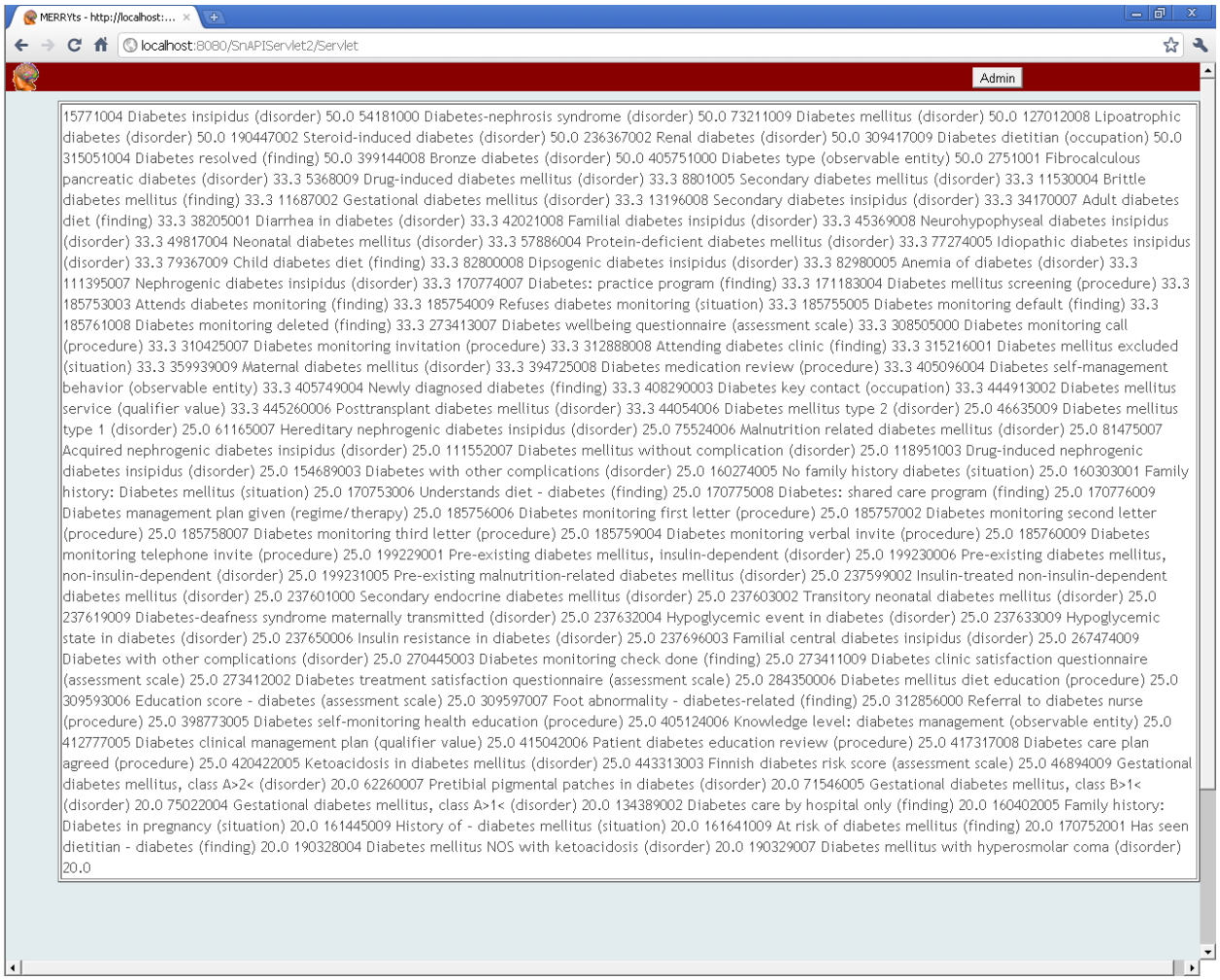
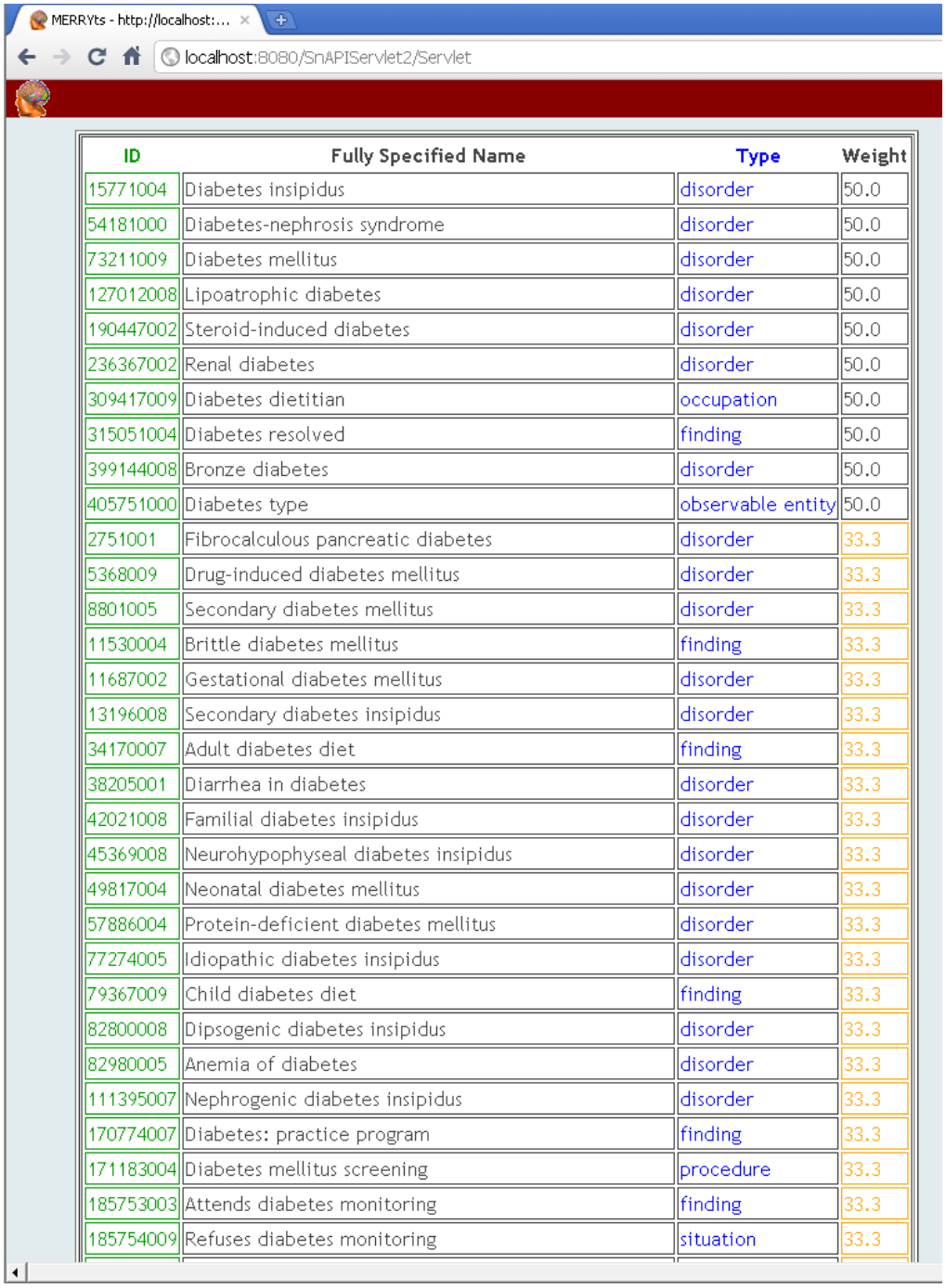


FIGURE 3 PLAIN TEXT OUTPUT

B.2 TABLE

The table presentation shows the result of a search, presenting the ID, name, category and weight (likelihood of the current concept being the searched concept). Likelihood can be influenced by configuration of the researcher or (if allowed by the researcher) the end-user.



ID	Fully Specified Name	Type	Weight
15771004	Diabetes insipidus	disorder	50.0
54181000	Diabetes-nephrosis syndrome	disorder	50.0
73211009	Diabetes mellitus	disorder	50.0
127012008	Lipoatrophic diabetes	disorder	50.0
190447002	Steroid-induced diabetes	disorder	50.0
236367002	Renal diabetes	disorder	50.0
309417009	Diabetes dietitian	occupation	50.0
315051004	Diabetes resolved	finding	50.0
399144008	Bronze diabetes	disorder	50.0
405751000	Diabetes type	observable entity	50.0
2751001	Fibrocalculous pancreatic diabetes	disorder	33.3
5368009	Drug-induced diabetes mellitus	disorder	33.3
8801005	Secondary diabetes mellitus	disorder	33.3
11530004	Brittle diabetes mellitus	finding	33.3
11687002	Gestational diabetes mellitus	disorder	33.3
13196008	Secondary diabetes insipidus	disorder	33.3
34170007	Adult diabetes diet	finding	33.3
38205001	Diarrhea in diabetes	disorder	33.3
42021008	Familial diabetes insipidus	disorder	33.3
45369008	Neurohypophyseal diabetes insipidus	disorder	33.3
49817004	Neonatal diabetes mellitus	disorder	33.3
57886004	Protein-deficient diabetes mellitus	disorder	33.3
77274005	Idiopathic diabetes insipidus	disorder	33.3
79367009	Child diabetes diet	finding	33.3
82800008	Dipsogenic diabetes insipidus	disorder	33.3
82980005	Anemia of diabetes	disorder	33.3
111395007	Nephrogenic diabetes insipidus	disorder	33.3
170774007	Diabetes: practice program	finding	33.3
171183004	Diabetes mellitus screening	procedure	33.3
185753003	Attends diabetes monitoring	finding	33.3
185754009	Refuses diabetes monitoring	situation	33.3

FIGURE 4 PRESENTED AS A TABLE

B.3 LIST OF DRAG AND DROP LABELS

This output can be used to post-coordinate SNOMED-concepts. Labels can be dragged to boxes. A check is performed to see if a label is dragged into the right box. In this way a first validation test can be performed. E.g. an infection have a causative agent bacteria x. An infection can have a location lung. An infection can't have a location bacteria x.

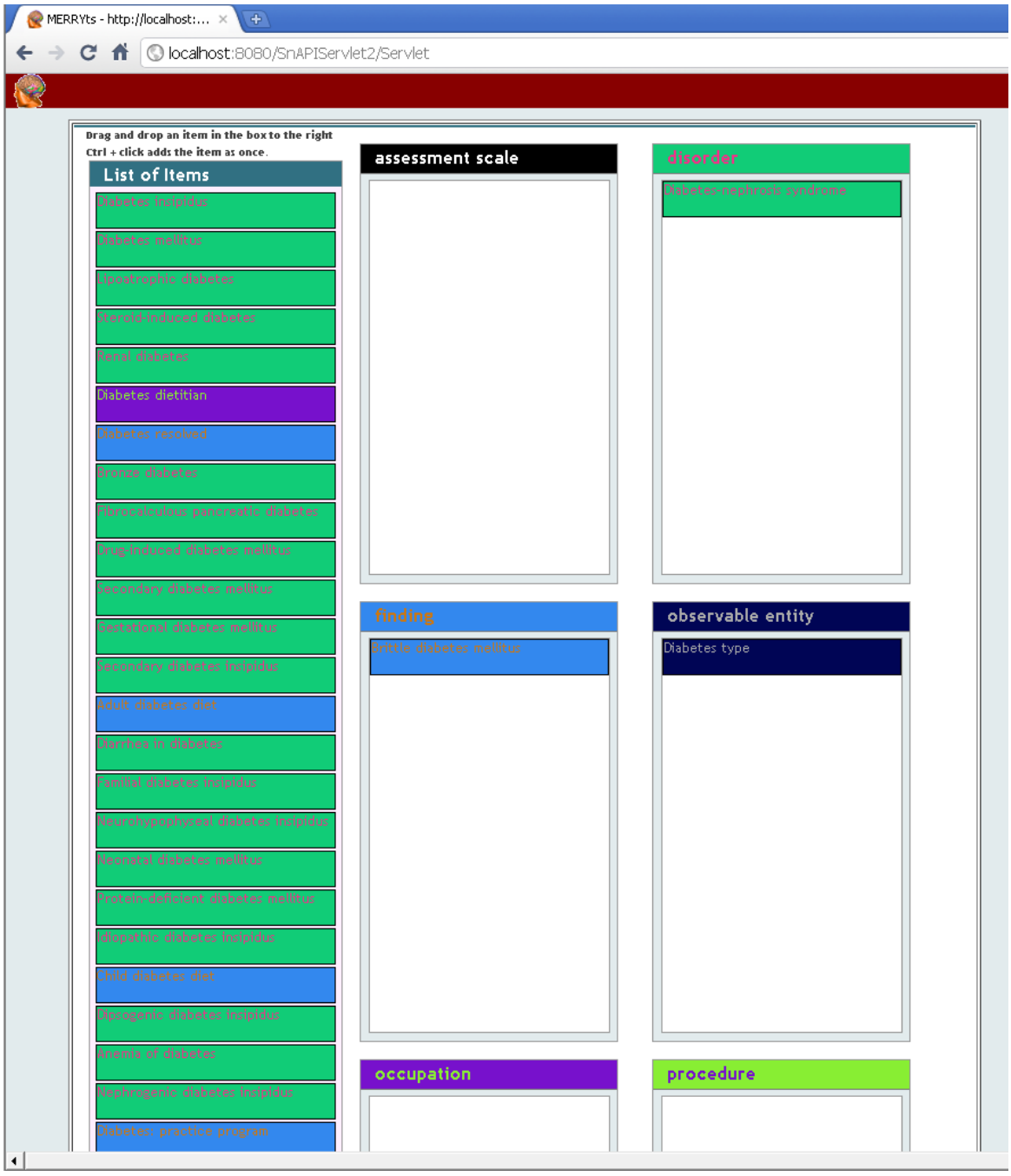


FIGURE 5 DRAG AND DROP LABELS

B.4 CLINICLUE-LIKE SKIN

CliniClue is a “freeware browser for SNOMED Clinical Terms®”. The current implementation in MERRYts is limited to the main screen. Only the search-function is implemented. The screen shows additional information on a concept like relations, hierarchy and current concept-status.

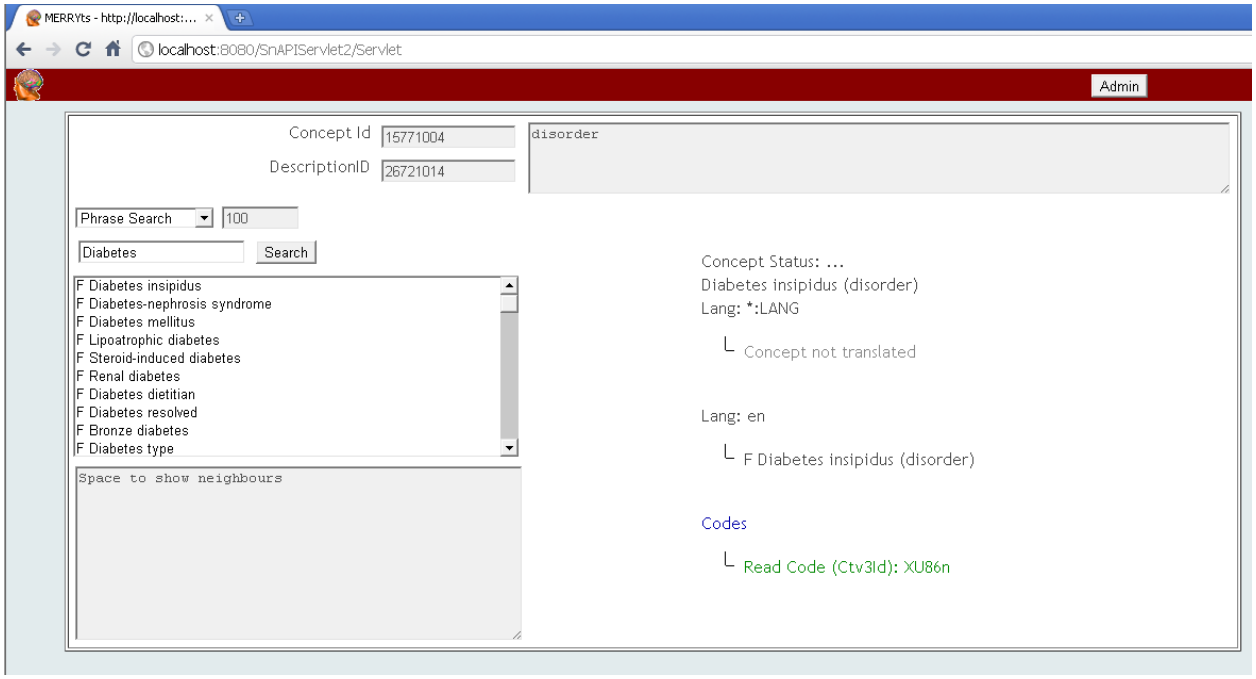


FIGURE 6 SKIN BASED ON CLINICLUE

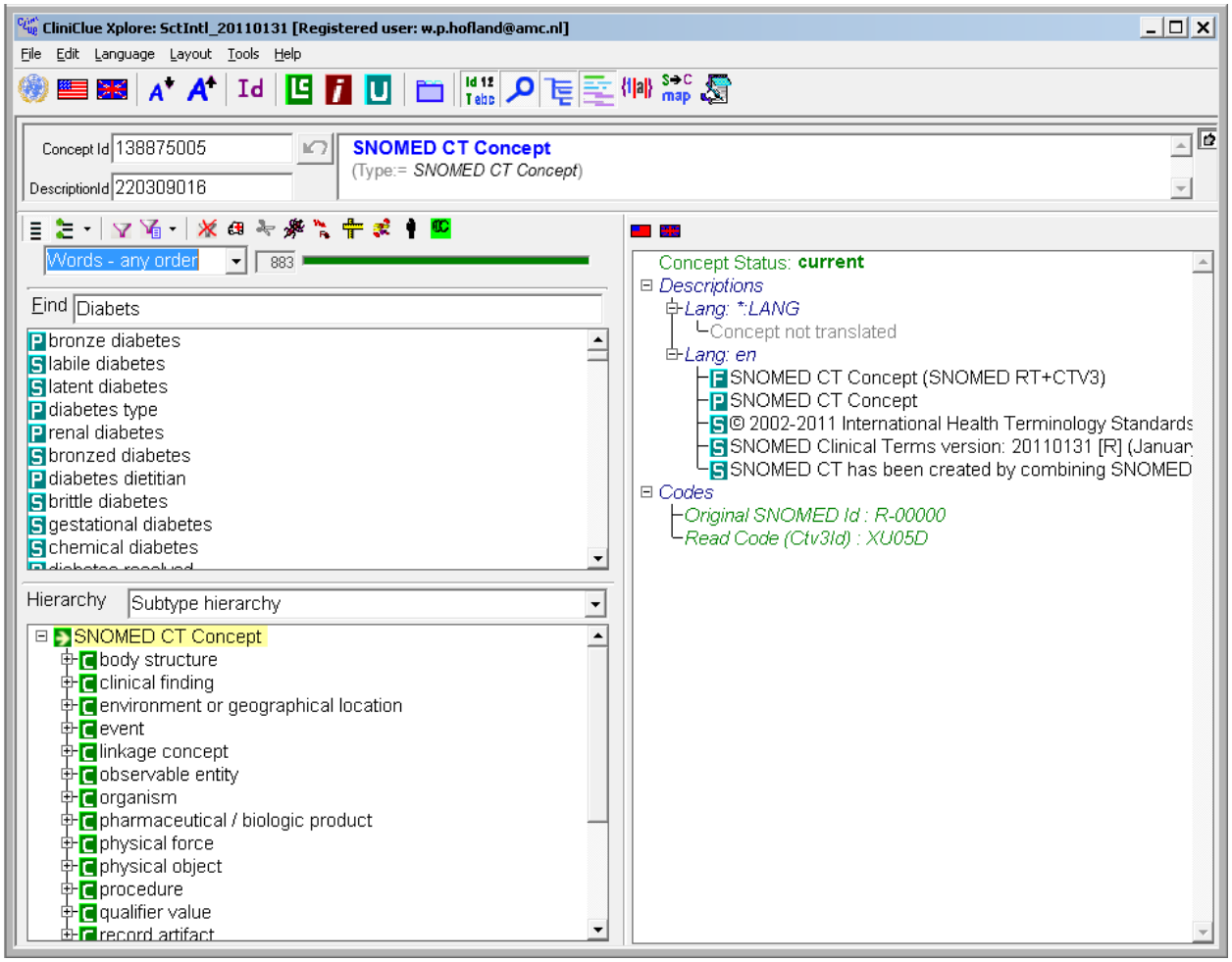


FIGURE 7 AN ORIGINAL CLINICLUE SCREEN

B.5 SNOB-LIKE SKIN

SNOB is a SNOMED Browser. It shows concepts in a tree grouped by categories. The fact that different main-categories are shown can be explained by the fact that a concept in SNOMED CT can be in different categories at the same time. This implementation in MERRYts does not look at the importance of a category. The first found category is shown as the main-category.

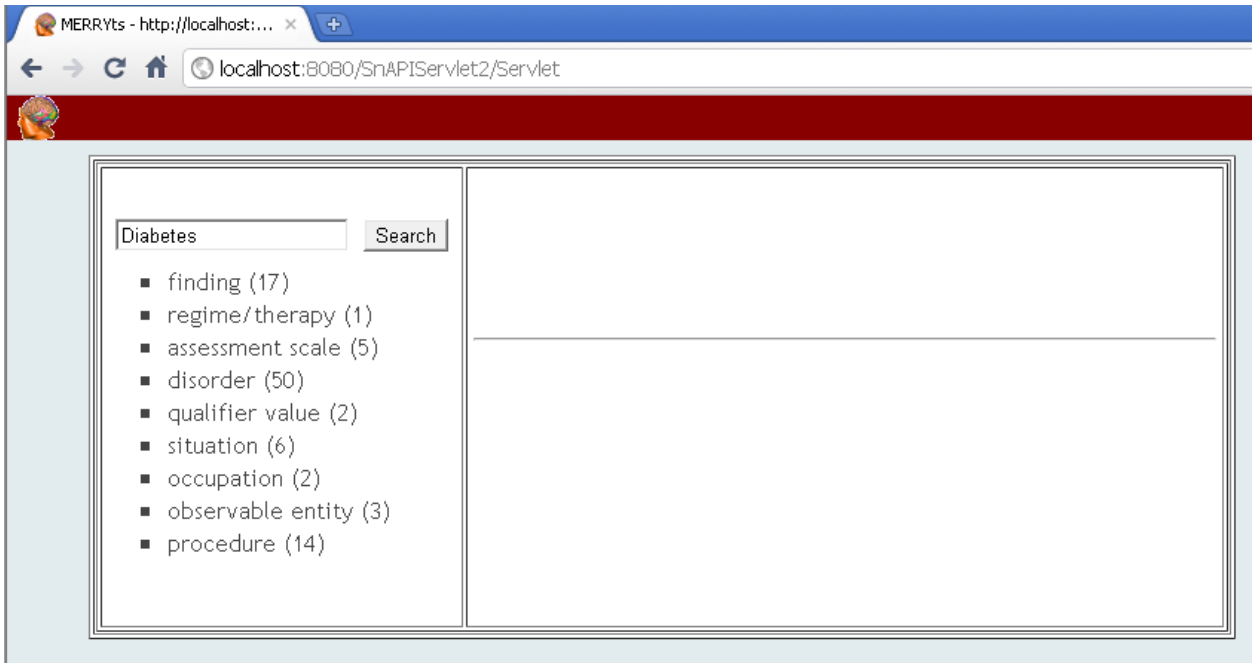


FIGURE 8 SKIN BASED ON SNOB

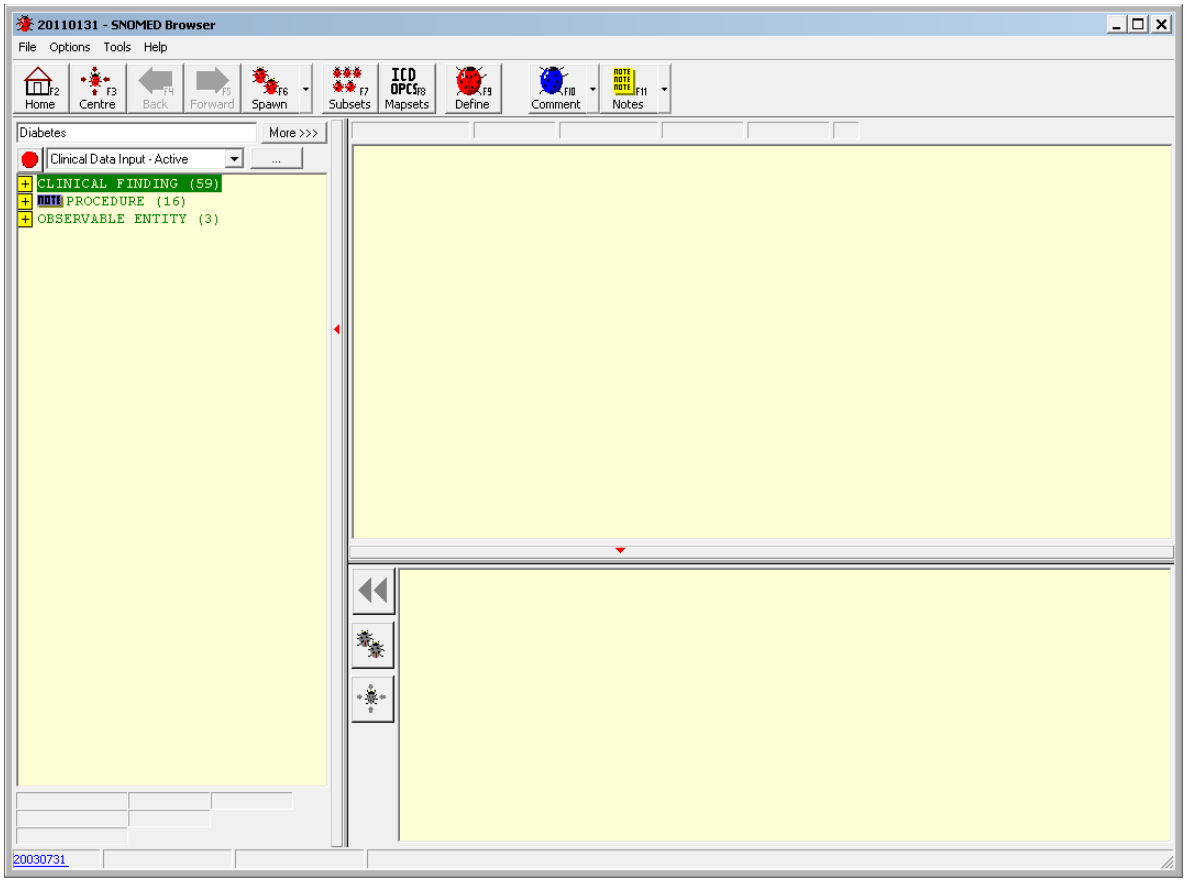


FIGURE 9 AN ORIGINAL SNOB SCREEN

B.6 THE ADMINISTRATION-SECTION

The administration-section can be shown by clicking on a button at the top of the screen. It is shown on the left side of MERRYts and is used to set the current interface. This part of MERRYts is typically only shown to the researcher that wants to use the tool in a usability evaluation of TS GUIs; the end-user user of the TS itself should not be able to get into this section.

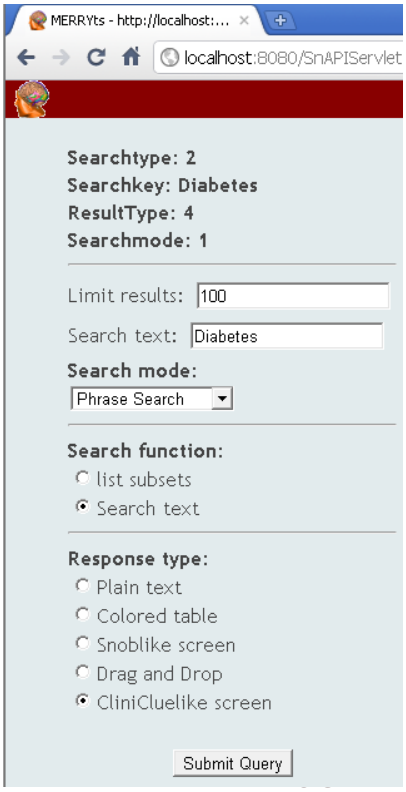


FIGURE 10 THE ADMINISTRATION SECTION

B.7 LIST OF IMAGES

Figure 1 Component Hierarchy in MERRYts 9
Figure 2 Flow during a request 9
Figure 3 Plain Text output 26
Figure 4 Presented as a table 27
Figure 5 Drag and drop labels 28
Figure 6 Skin based on CliniClue 29
Figure 7 An original CliniClue screen 30
Figure 8 Skin based on SNOB 31
Figure 9 An original SNOB screen 32
Figure 10 The administration section 32