Specification of the architecture

Distribution: Public

MedIEQ

Quality Labelling of Medical Web content using Multilingual Information Extraction

National Centre for Scientific Research "Demokritos"
Teknillinen Korkeakoulu – Helsinki University of Technology
Universidad Nacional de Educacion a Distancia
Collegi Oficial de Metges de Barcelona
Zentralstelle der deutschen Ärzteschaft zur Qualitätssicherung in der Medizin
Vysoka Skola Ekonomicka V Praze
I-Sieve Technologies Ltd

2005107 **D12**

February 2007

Project ref. no.	*2005107*
Project acronym	*MedIEQ*
Project full title	*Quality Labelling of Medical Web content using Multilingual Information Extraction*

Security (distribution level)	*Public*
Contractual date of delivery	*31 December 2006*
Actual date of delivery	*16 February 2007*
Deliverable number	*D12*
Deliverable name	*Specification of the architecture*
Type	*Report*
Status & version	*Final*
Number of pages	36
WP contributing to the deliverable	*WP8*
WP / Task responsible	*I-sieve*
Other contributors	*NCSR, UNED, UEP, TKK*
Author(s)	*K. Chandrinos (i-sieve), K. Stamatakis, P. Nasikas, V. Gatos, V. Karkaletsis (NCSR), M. Ruzicka, V. Svátek (UEP), E.A. Cabrera, V. Peinado (UNED), M. Póllä, T. Honkela (TKK)*
EC Project Officers	*Artur Furtado*
Keywords	*Web content collection, information extraction, linguistic resources management, label management, monitor a website, alert an expert, assisting quality assessment.*
Abstract (for dissemination)	*This document specifies the architecture for the integrated MedIEQ system (AQUA system). Several toolkits (content collection, information extraction, resources management, label management, monitor-update-alert) inter-operate aiming to assist the labelling expert to his work. The open architecture and the internationalization are among the main design principles of AQUA architecture.*

Table of contents

Executive summary

Based upon state-of-the-art technology in the areas of web crawling and spidering, multilingual information extraction, semantic resources and quality labelling, MedIEQ paves the way towards the automation of quality labelling process in health related web sites.

MedIEQ, aiming to facilitate the work of health quality labelling agencies, delivers tools that:

- collect unlabeled health-related web resources, in seven European languages, and label them according to a set of pre-specified labelling criteria,
- monitor already labelled health web resources alerting labelling experts in case the sites' content is updated against the labelling criteria,
- generate and maintain machine readable labels for health-related web resources, either manually or semi-automatically using the above tools.

MedIEQ continues and builds upon the work of previous projects in the areas of

- medical quality labelling (MedCIRCLE[1], MedCERTAIN[2], WRAPIN[3]),
- quality labelling standards (QUATRO[4]), and
- web content collection and extraction (Crossmarc[5], Rainbow[6]).

MedIEQ aims to tackle the main problem of current medical quality labelling mechanisms, that is, the need for a continuous review and control of medical web sites, a process which requires a huge amount of human effort. The resulting technology is expected to have a significant impact on medical quality labelling assisting the work of labelling experts, increasing the number of labelled medical sites across Europe and their effective monitoring, and thus improving the quality health knowledge disseminated through the Web.

This deliverable describes the architecture of the MedIEQ system (AQUA), which integrates the various tools developed in MedIEQ for content collection, information extraction, labels management, multilingual resources management, in order to assist the labelling expert in his/her work. The modularity and the internationalised approach are among the main design principles of AQUA architecture aiming to facilitate the use of more techniques and tools as well as the porting to new languages.

[1] http://www.medcircle.org/
[2] http://www.medcertain.org
[3] http://www.wrapin.org/
[4] http://www.quatro-project.org/
[5] http://www.iit.demokritos.gr/skel/crossmarc/
[6] http://rainbow.vse.cz/descr.html

1. Introduction

AQUA “Assisting Quality Labelling” is the name of the integrated MedIEQ system, a system aiming to assist the work of labelling experts. AQUA consists of several subsystems or toolkits having different roles in the generation and maintenance (including monitoring) of quality labels for health-related web resources.
AQUA is currently accessible from: http://www.medieq.org/aqua/aqua/welcome.seam

The first version of the integrated MedIEQ system, covering two of the project languages, English and Spanish, is scheduled at the end of June 2007.

This deliverable presents AQUA architecture. In section 2 we explain our choices concerning the AQUA design and implementation. In section 3, we decompose AQUA in its subsystems, describe the MedIEQ repository, discuss on the MedIEQ and the UMLS databases and explain the proposed integrated system architecture. Section 4 provides details on the MUA (Monitor-Update-Alert) toolkit and its components. In section 5 we describe the integrated system use case and provide the AQUA user manual by use case (details on the use cases can be found in deliverables D6, D8 and D10). In section 6 we discuss the remaining steps to the final version and present some early conclusions.
Finally, Appendix A gives the list of the MedIEQ tools by toolkit and work package, Appendix B contains the complete list of the MedIEQ metadata grouped by database table and Appendix C has the schema of the MedIEQ database.

2. Design specifications

In this section we explain:

1. How do we design AQUA for maximum modularity?
2. How internationalization is achieved?
3. Why do we decide to make AQUA a web-based system?
4. Why we opt for a centralized rather than distributed system architecture?

2.1 Making the AQUA system modular

AQUA was designed to provide maximum modularity and extensibility. Such a design enables us to test modules providing the same functionality (e.g. machine learning algorithms) with different implementations so that we can select the one that fits best to our data. Having a modular system lets us focus on the design and implementation of the system. We will not couple semantics or responsibilities from different modules and we will not mix control or presentation related code with application logic. It makes also easy to remove or change system components on demand. The AQUA system is deployed on the JBoss application server[7], which enables system modules to be added and removed from it without any need to stop the system execution to redeploy the application. This is accomplished by implementing some of the system modules as JBoss services, a standards based mechanism for creating service-based modular systems. This mechanism provides us with the infrastructure to manage, monitor and properly secure the modules that our system will use.

Java is used as the programming language for the system but we go far beyond a plain java stand-alone program. The AQUA system is running in the context of an application server (JBoss) and uses several services provided from it. JBoss itself and the provided services are standards based java APIs for designing enterprise systems, implemented by JBoss inc. (now a RedHat[8] division), so the system can be deployed in any other standards based java application server (IBM websphere[9], BEA[10], SUN glassfish[11], etc.). The services provided to AQUA from the application server relieve us from the need to take into account issues such as system modularity, scalability, load balancing, management and monitoring. Using these services and following the design patterns induced we can focus on our own application logic and domain models.

Our application uses class types, components and deployed services, listed in Table 1; we will provide more information on them in the following subsections.

Table 1: Components - Responsibilities

EJB3 Session beans	Application logic - Control
JBoss managed services	Application logic
EJB3 Entity beans[12]	Model
Java Server Faces[13]	Presentation
Seam[14]	Presentation - Control

[7] http://jboss.com/
[8] http://www.redhat.com/
[9] http://www-306.ibm.com/software/websphere/
[10] http://www.bea.com/framework.jsp?CNT=homepage_main.jsp&FP=/content
[11] https://glassfish.dev.java.net/
[12] http://labs.jboss.com/portal/jbossejb3
[13] http://java.sun.com/javaee/javaserverfaces/

EJB3 Session and Entity beans

Application control, logic and model mapping are implemented with EJB3 Session and Entity beans and JBoss JMX[15] managed services. We tried to follow some design patterns that would let us build a flexible componentized system that would be easy to extend and configure.

EJB3 Entity beans are Java classes that their properties map and model (datatypes, constraints) directly to database tables. For each database table we have created a different Java class. EJB3 Session beans are the application controllers. Each user request, whether it is a button click or a url request is mapped to method in an EJB3 Stateful or Stateless Session bean that handles it by executing some other Session bean method, sending or getting data from the database or dispatching the request to a Container hosted Service. For example, when the user selects a file to upload, the upload method is called from the UploadAction Stateless bean that uploads the data, creates an RDF[16] model from them, then dispatches the validation task to another EJB3 Session bean and if the validation succeeds calls another method from UploadAction to save the data to the server.

Services

Application Server (Container) deployed services give us maximum flexibility over controlling, changing and configuring application logic in a manner independent of the application control and presentation, on the one hand, while, on the other hand, let us enable and disable services and functionalities while the application is online. Once we have the services deployed and running on the Application Server we will call the methods we want from the EJB3 session beans.

The AQUA Services (every nested component is still an independent service used by its parent service) are:

- Crawler
- Spider
  - CCC (content classification component)
  - LSC (link-scoring component)
- TMG (trained module generator)
- LOFT (link-object formation tool & training)
- Information Extraction (IE) Engine
  - Statistical techniques IE Engine
  - Other IE Engines
- LAM (label management)
- MRM (multilingual resources management)
- MUA (monitor-update-alert)
  - MSC (monitoring scheduler)
  - DBU (database updater)
  - AME (alerting mechanism)

JSF

AQUA web user interfaces have been built with MyFaces implementation of the Java Server Faces technology and Java facelets[17]. We created a page template which contained the page structure, navigation menus, logos for header and footer. Then each new page was simply a

[14] http://jboss.com/products/seam
[15] http://www.jboss.org/developers/projects/jboss/jbossmx
[16] http://www.w3.org/TR/rdf-schema/
[17] https://facelets.dev.java.net/

facelet included in the template. Each new facelet created contains only the JSF and XHTML elements needed for a certain task. For example the facelet for the file uploading contains only a file selection control, an upload button and a placeholder for validation messages. This way we have taken advantage of a templating system to maximize flexibility for our system. Each facelet renders static and dynamic text messages and sends actions to EJB3 session beans (stateful and stateless) through jsf buttons and links.
Seam provides us with JSF components, Java annotations for configuring session beans and linking them with the JSF pages and finally a higher level application context that lets us manage and track user sessions and workspace easier.

2.2 Internationalization support

The application user and project partner scope spreads over five European countries (Czech Republic, Finland, Greece, Germany, Spain) and seven different languages (EN, ES, DE, CA, GR, FI, CZ). It has been a requirement for the AQUA system interface to be internationalised as a means to enhance usability. The AQUA system overall is also needed to be able to handle and process content (web resources) in several languages as well. The first requirement is handled by using the JSF and JBoss Seam mechanisms for internationalizing the interface and handling Dates and Time zones. The second requirement is handled by using UTF-8 strings for the internal processing of the content downloaded from the web and saved in the database.
We will explain the mechanism that supports the user interface Internationalization and will show how a new language can be added without any need to change the application code. First, we define the languages to support in the application using the ISO639 code[18] [5] for each one, in the configuration file faces-config.xml. We define English as the default language and all the other, that will be loaded on demand, as supported.

How to Internationalise

The mechanism that handles content Internationalization is provided from JSF and JBoss Seam. The languages configured at the faces-config.xml are linked with plain text files containing all the Strings presented to the users with a simple mechanism. As noted, every string presented to the user is contained in a text file that follows a certain naming convention as demanded by the Java Localization infrastructure and JSF / JBoss Seam. Each language should have its own text file filled with lines containing variable names and the Strings each one represents linked with the '=' character. These variable names will be used by our program and web pages internally and they are the same for each language. What changes is the value that must be the message translation in each language. So, for each language we have a file that follows the naming convention

filename_languageCode.properties

and placed in a directory that is included in the application classpath. For example, the file labels_el.properties will be holding all the messages in the Greek language and when the user selects the Greek language option from the drop down box the messages will be read from that file and be presented to the user. It is important to note that the messages can be reused in any part of the application and do not contain presentation or layout notation.
For a new language to be added, all that is needed is to have a translation of the terms found of any labels_languageCoge.properties file to the new language and a simple option to be added to the faces-config.xml file by the system administrator.

How to change the AQUA user interfaces language

On the user interface, we provide a drop down list control that presents the available languages for the user interface. The user can select a language, click the button 'Change', and the application will be reloaded with the user interface presented in the selected language.

[18] http://www.loc.gov/standards/iso639-2/php/code_list.php

2.3 Why web-based?

AQUA will have the following 2 types of users:

- the System administrator,
- the Labelling Expert user (a specialist from WMA[19] or AQUMED[20]).

From the Labelling Expert user point of view, they currently use Web interfaces in their daily work and can be considered familiar with web environments. On the other hand, System administrators will be able to administer the MedIEQ system from any PC, anywhere.
Several advantages and disadvantages of web interfaces when compared to desktop application interfaces can be found in developer's forums and online discussions. Just to mention the most important:

Advantages:

1. Accessibility: they are accessible from anywhere (any pc with Internet connection),
2. Simplicity: they are simple to learn and remember.
3. Navigation: offer superior navigation and menuing capabilities over desktop alternatives.
4. Portability: they have superior cross platform portability with low server-side proprietary lock-in.
5. Statistics: It is easier (ready-made solutions exist) to monitor your users and keep records of their actions.
6. Context help: Web interfaces make it easy to provide links to extra information about input fields, errors, posting options, updates, etc.

Disadvantages:

1. Security: Web interfaces have security and reliability vulnerabilities when using scripting and rich components.
2. Performance: they have runtime performance issues relative to the desktop.

The first disadvantage doesn't much concern us as far as we have a tendency to avoid an extensive use of rich components. The second is partially addressed by the centralized solution analyzed below.

Considering the above, we opt for web interfaces.

2.4 Why centralized?

The work flow between applications in MedIEQ involves, in many cases, transfer or re-use of content (html pages, pdf documents, etc.). To give a few such examples:

- In WCC (web content collection toolkit): The Spider forwards content to CCC (content classification component);
- In WCC: CFT (corpus formation tool) helps the user to form corpora to be used by TMG (trained module generator);
- Content collected by WCC is forwarded to IET (information extraction toolkit);
- Resources from MRM (multilingual resources management toolkit) are exploited by WCC or IET or LAM (label management toolkit);

The case of a distributed architecture, where toolkits or components run in different servers (e.g. Athens, Prague, Madrid, Helsinki), presents some disadvantages that a centralized solution doesn't have:

[19] http://wma.comb.es/home.php. WMA: Web Mèdica Acreditada (ES), a health quality labelling agency and a MedIEQ partner.
[20] http://www.aqumed.de/. AQUMED: Agency for Quality in Medicine (DE), a health quality labelling agency and a MedIEQ partner.

- the deployment of web services is necessary;
- resources in different machines should be available;
- run-time slowdowns seem probable (reasons: delay in file transfers between applications caused by slow connections, limited bandwidth, network overload, etc.);
- the system is paralyzed when a component is off (possible reasons: network failure, server down, etc.);
- the maintenance and the administration of a distributed system involves access to various servers, handling several user access rights and permissions and the implementation and maintenance of multiple versioning systems;
- logs and statistics are harder to manage in a distributed solution.

Taking into account the above, between centralized and distributed, we opted for centralized.

3. The architecture

3.1 Which are the AQUA subsystems?

The prototype MedIEQ labelling assisting system AQUA consists of 5 subsystems or toolkits:

1. the label management toolkit (LAM),
2. the web content collection toolkit (WCC),
3. the information extraction toolkit (IET),
4. the multilingual resources management toolkit (MRM),
5. the monitor-update-alert toolkit (MUA).

LAM manages (generates/validates/modifies/compares) quality labels based on the schema proposed by MedIEQ (*for further details see Deliverable D4.1 & D5*).

WCC identifies, classifies and collects on-line content relative to a number of machine readable quality criteria (proposed by the labelling agencies participating in the project) in seven languages: EN, ES, DE, CA, GR, FI, CZ (*for further details see Deliverables D6 & D7.1*).

IET analyses the web content collected by WCC and extracts attributes for MedIEQ compatible content labels (*for further details see Deliverable D8*).

MRM gives access to health-related multilingual resources from the UMLS System[21] (includes MeSH[22], ICD[23], etc.); input from such resources is needed in specific parts of both the WCC and IET toolkits (*for further details see Deliverable D10*).

MUA handles a few auxiliary but important jobs, like the configuration of monitoring tasks, the MedIEQ database entries updates, the alerts to labelling experts when important differences occur during monitoring existing quality labels (*for further details see section 4 in this document*).

Finally, all data necessary to the different subsystems as well as to the overall AQUA system are stored in:

- the MedIEQ repository,
- the MedIEQ database,
- the UMLS database.

(*for further details see following subsections 3.3, 3.4 & 3.5 in this document*)

3.2 Description of the integrated architecture

In the proposed architecture for our system (Figure 1), we distinguish three levels: the user interface level, the application level and the storage level.
The user interface level consists of the web interfaces for both the MedIEQ user types: the labelling expert and the system administrator.
The application level is where all software components are deployed. It actually supports all functionalities provided by the AQUA interfaces connecting, at the same time, the user with the data (user preferences, stored web documents and metadata on these documents, linguistic resources, RDF labels, etc.).

[21] http://www.nlm.nih.gov/research/umls/about_umls.html
[22] http://www.nlm.nih.gov/mesh/
[23] http://www.who.int/classifications/icd/en/

Finally, we have the storage level consisting of

- the MedIEQ repository,
- the MedIEQ database,
- the UMLS database.

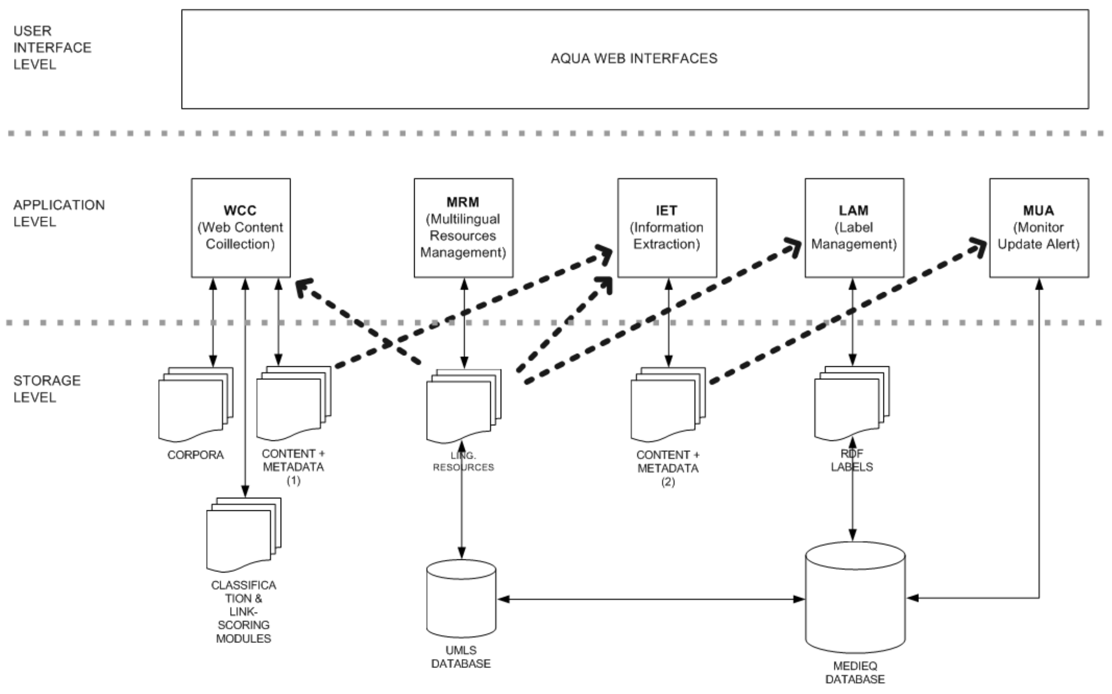


Figure 1: The AQUA system architecture

Figure 1 also shows all the possible data flows in AQUA (dashed arrows).

- From WCC to IET: pages and documents collected by WCC, once undergone a first level extraction by WCC (extraction of metadata 1), are then forwarded to IET for further processing (extraction of metadata 2).
- From IET to MUA: MUA (actually its component DBU) takes all metadata collected by both WCC and IET and updates the MedIEQ database.
- From MRM to WCC, IET, LAM: custom vocabularies (consisting of parts of the linguistic resources supported by MRM) generated by the MedIEQ users through MRM interface, can be accessed from other toolkits (e.g. WCC, IET and LAM), where the user may need them.

3.3 The MedIEQ repository

The MedIEQ repository, a storage area in the MedIEQ server, consists of:

- **The MedIEQ subversion system area**: hosts the MedIEQ versioning system; all binary and ASCII files needed in the project (software, documentation, the project deliverables and reports, etc.) are submitted here.
- **The MedIEQ runtime area**, storing:
    - o all configuration files needed by the different parts of AQUA,
    - o all files (e.g. web pages) and metadata needed by the different AQUA toolkits on runtime (temporal store), and
    - o those files, from temporal store, being actually exploited; meaning: all those documents from which useful information has been extracted (permanent store).

3.4 The MedIEQ database

The MedIEQ database schema was created in Postgresql v8.1.4 database and is part of the overall AQUA schema. We access the database from our EJB3 Session beans through EJB3 entity beans that map the tables to Java classes. The EJB3 persistence manager called from Session beans is used to save, update, delete and retrieve data to the database through the proper mapping entity beans. The schema of the MedIEQ database is provided in Appendix C. It actually maps the MedIEQ metadata as these are grouped and listed in Appendix B.

The metadata defined in MedIEQ are grouped in the following eight sets:

1. Metadata on a Web resource (doc or site) – General metadata
2. Metadata on File (or Document)
3. Metadata on Extracted Information
4. Metadata on User
5. Metadata on Label
6. Metadata on Corpus
7. Metadata on Alert Profiles
8. Metadata on Tasks

Appendix B gives details on all metadata included in the above eight sets.

Note that in MedIEQ, as a resource it is considered either

- a website (or a part of a website) consisting of several documents (fits to both WMA and AQUMED), or
- a single document (fits only to AQUMED), which may be one of the following:
  - an html page,
  - a Word document (.doc),
  - a PDF document (.pdf).

3.5 The UMLS database

The UMLS Metathesaurus is a very large, multi-purpose, and multi-lingual vocabulary database that contains information about biomedical and health-related concepts, their various names, and the relationships among them. It is built from the electronic versions of many different thesauri, classifications, code sets, and lists of controlled terms used in patient care, health services billing, public health statistics, indexing and cataloguing biomedical literature, and/or basic, clinical, and health services research.

In the Metathesaurus, all the source vocabularies are available in a single, fully-specified database format. It is organized by concept. Its purpose is to link alternative names and views of the same concept together and to identify useful relationships between different concepts. It is necessary to define which Metathesaurus concepts and terms will be available depending on the use case. Other Metathesaurus concepts and terms will then provide synonyms and related terms which can help to lead users to the vocabularies selected for a particular use case.

The UMLS Database can be accessed through the Internet via UMLSKS (UMLS Knowledge Source Server), but there are some problems accessing remotely to the UMLSKS mainly with firewalls. It is preferable to create a local copy of the database using the custom tools provided with the UMLS data. These tools offer database customization so there's not a fixed schema associated to the UMLS database. The definitive database schema for the MRM repository will be decided to improve the access to the selected resources of UMLS database and custom vocabularies.

3.6 A document’s lifecycle

To track a document’s lifecycle, starting from when the document is identified by the spidering mechanism until when the MedIEQ database has been updated with all relevant data being extracted from within this document, the schema of the architecture in Figure 1 should again be our guide.

1. WCC

The WCC mechanism may be initialized either by an AQUA user or by a monitoring task set-up by an AQUA user but scheduled to run in specific time intervals, e.g. once a month. WCC identifies and stores the document and some metadata on it (content + metadata 1 in Fig. 1). The Spider, with the help of the link-scoring component decides which links to follow and which to ignore. When following a URL, the Spider, with the use of the content classification component, decides whether the accessed document fits to one or more MedIEQ classification categories. Fit documents need further processing (information to be extracted) and are locally stored (along with some initial metadata extracted by the Spider).

2. IET

IET extracts further information from the local copy of the document and additional metadata slots are filled.

3. MUA

All extracted information, by both WCC and IET, has to be stored in the database; for this, Database Updater (DBU) is responsible. In case there is some conflict between the new and possibly existing data (derived from the same resource in a previous run) the AQUA alerting mechanism (AME) is activated and an alerting e-mail message is sent to the MedIEQ user who had initialized WCC.

Independently from the above described steps, LAM could be asked to generate a RDF quality label, by aggregating all the information corresponding to a MedIEQ resource. In fact, the aggregation is needed only when reviewing an entire or a part of a web site (see definition of a MedIEQ resource in 3.4), where information from several documents contribute to the label. In contrary, when reviewing a single document (AQUMED case), no aggregation is necessary (all data can be found into a single ID database entry).

4. The MUA (Monitor-Update-Alert) toolkit

4.1 The tools of MUA

Below we can see the tools of the MUA toolkit.

```
MUA (Monitor – Update – Alert)
 |
 |---MSC (monitoring scheduler)
 |
 |---DBU (database updater)
 |
 |---AME (alerting mechanism)
```

MUA complements the content collection and the extraction toolkits, WCC and IET respectively. With the help of MSC, a user can create and configure monitoring tasks (see 4.2). The DBU is responsible for the maintenance of the MedIEQ database, providing both automatic and manual updating mechanisms (see 4.3). Last, AME provides a configurable mechanism alerting the user in case the monitored resources' content is updated against the quality criteria (see 4.4).

4.2 MSC (Monitoring Scheduler)

Description

MSC is a mechanism allowing the AQUA user to create and configure monitoring tasks. A monitoring task is a cron job. It can be either a searching-for-unlabelled-health-resources or a reviewing-health-resources task. To configure a task, MSC needs the information listed in Appendix B, paragraph 8. The AQUA user interface of MSC is shown in section 5, fig. 3.

Specifications

a. What is a MedIEQ task: a task is a cron job that runs in specified time intervals, which may be week | month | year.
b. A task's owner: a task is always bound to a specific user, its creator; he only has the right to re-configure, temporarily cease or completely remove the task.
c. Active/inactive task: a task may be set inactive, which means that although we may be between its start and end dates, the task is not running.
d. A task's base directory: all configuration, initialization, output files of a specific task are stored in its base directory, a directory in the local disk automatically assigned by the system.

Actions of the MSC

(Group 1 - Actions provided to the labelling expert user through the expert interface)
MA1. Create a new task (a search-t which means a searching-for-unlabelled-health-resources or a review-t which means a reviewing-health-resources task), either from scratch or by loading the configuration options from another, existing task.
MA2. Edit existing tasks: rename, reschedule, remove, etc.
MA3. When for search-t, import terms from MRM,
MA4. When for review-t, import URLs from Crawler's results.
(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
There is no such action.

4.3 DBU (Database Updater)

Description
DBU is the AQUA mechanism responsible for the maintenance of the MedIEQ database and can work in two ways: automatic and manual. On the one hand, DBU automatically inserts new entries taking the output of both WCC and IET. Alternatively, an AQUA user may manually insert new data in the MedIEQ database, update contained information and also remove unwanted entries. Manual updates are restricted to information included in MedIEQ RDF labels.
Additionally, in automatic updates, DBU compares the data to-insert with existing ones (corresponding to the same resource) and marks possible differences. Then, when these differences are important enough[24], DBU triggers the reaction of AME.

Specifications

a. Automatic database updates: the biggest part of the MedIEQ database is maintained and updated without any human intervention (data coming from the AQUA tools and components are automatically handled by DBU).
b. Manual database updates: DBU provides web forms through which a labelling expert can insert new data and update or remove existing database entries. However, given that manual database updates are allowed only to information included in RDF labels, these forms can be found under "Label management" in AQUA web interfaces (see Fig. 5).
c. Differences between new and existing data: once a difference identified, DBU keeps pointers to the conflicting entries into ConflictingData table (see Appendix B). When those differences are important (an issue to be resolved), DBU calls AME in action.

Actions of the DBU
(Group 1 - Actions provided to the labelling expert user through the expert interface)
DA1. Update information contained in the MedIEQ database.
DA2. Remove unwanted entries.
(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
DA3. Define differences importance threshold.

4.4 AME (Alerting Mechanism)

Description
AME is the AQUA alerting mechanism. When AQUA reviews/monitors a health resource, information relevant to the quality criteria is extracted from its contents. In case of conflicts between the recently extracted and possibly existing data (extracted from the same resource in previous runs) the AQUA alerting mechanism (AME) is activated and an alerting e-mail message is sent to the MedIEQ user who had initialized the review/monitoring process.

Specifications

a. Alert profiles: AME allows various alert profiles per user (stored in Table AlertProfiles of the MedIEQ database). A user can modify and/or remove only the profiles created by him.

[24] This is an issue still under discussion between the MedIEQ partners. An approach we examine is the definition of an importance threshold (which eventually needs to be "estimated" through real use and evaluation). The idea is to calculate the ratio of conflicting to non-conflicting cases (among all conflicting cases from a given resource) and determine their overall importance by comparing this number to the importance threshold. This threshold should be possibly defined by the system administrator only.

b. Configuring an alert profile: an alert profile contains the alerting preferences specified by its owner, which include the e-mail address to receive alerts, the alerting frequency and the exact parts of information to be included in an alert message (the resource URL, the criteria where differences occur, the URLs where conflicting information was identified, etc.). A preview of the interface for the creation / configuration of alert profiles is given in Fig. 6.

Actions of the AME

(Group 1 - Actions provided to the labelling expert user through the expert interface)
AA1. Create a new alerts profile.
AA2. Edit/update an existing alerts profile.
(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
There is no such action.

5. User manual

5.1 The integrated system use case: review / monitor a health resource

An actor is a user; it's anyone who can use the AQUA system and/or its different toolkits and components. An actor can be either a user or a program (S/W). Furthermore, we distinguish two interaction levels:

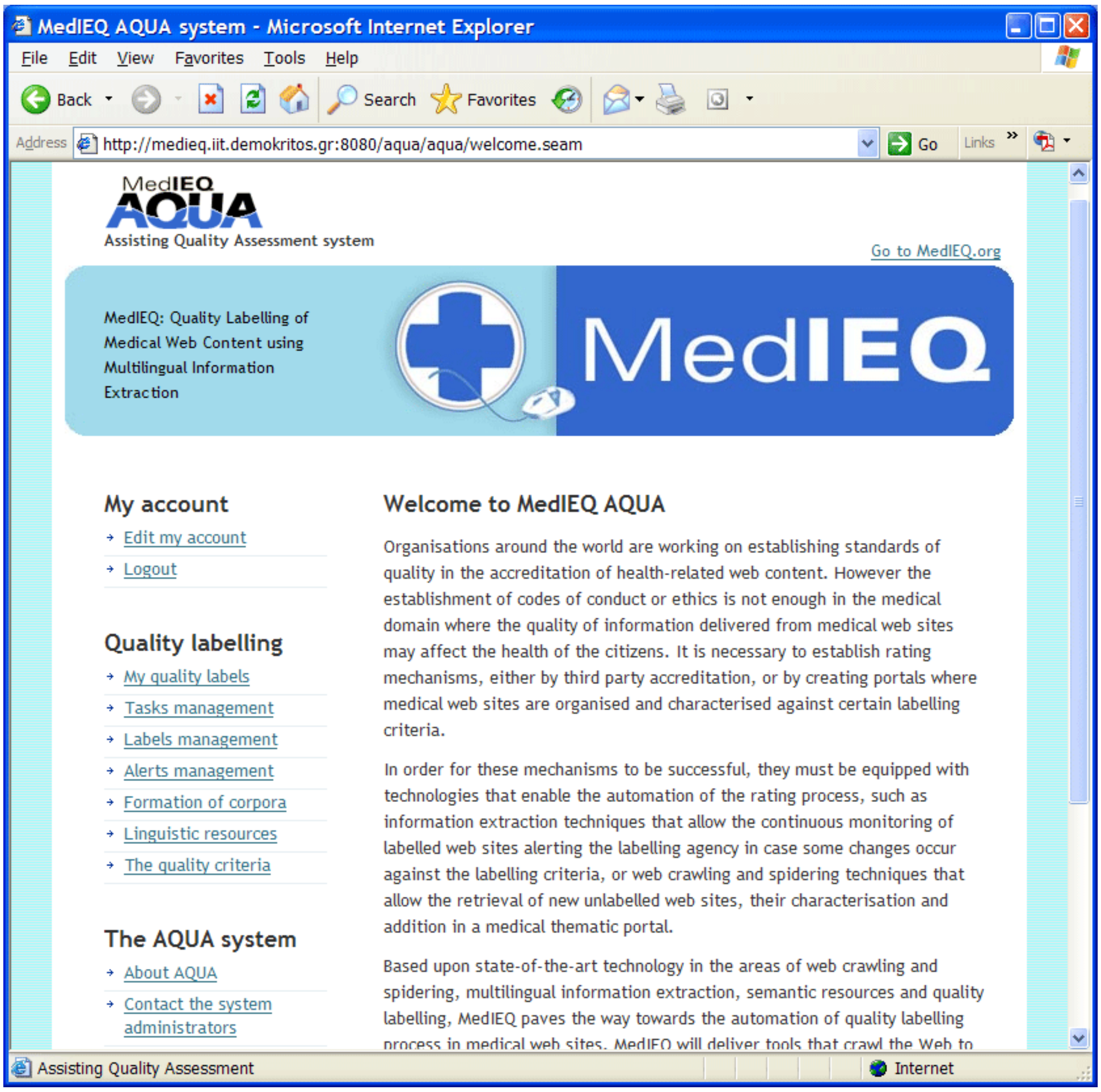
- High-level Interactions [-H-]: where actors directly interact with each other. E.g. toolkits interact with other toolkits, the database and the cron jobs.
- Low-level Interactions [-L-]: within this level we group the interactions between components inside a toolkit.

Actors
[-H-] Labelling expert, WCC, IET, MUA
[-L-] Spider, CCC, LSC, extraction engines, DBU, AME
Interactions
[-H-] The labelling expert wants to review or monitor a web resource against specific quality criteria. In case of an immediate review, the system is activated, by the labelling expert, via the AQUA UI (see Fig. 3). In case of a scheduled monitoring task, the system is activated by MSC. An activation of the AQUA system signifies that various components from different toolkits will run one after the other. First, WCC is called.
[-L-] WCC calls the Spider to collect, by storing locally, on-line health-related content. The user specifies a number of parameters like the web resources to be spidered, the preferred languages of the required content, the accepted content type(s), the labelling criteria etc.
[-L-] The Spider runs: a) internal (same host) links are collected and scored (LSC is called) and the most promising followed, b) every visited page's content is classified (CCC is called) and fit (to specified criteria) pages are locally stored.
[-H-] IET is called: once finished, WCC dispatches a relevant message to IET.
[-L-] IET runs: IET passes the collected content successively from the extraction engines it disposes. Extracted information is forwarded to MUA.
[-H-] MUA is activated: it calls DBU and, possibly, AME.
[-L-] DBU updates the MedIEQ database.
[-L-] In case of changes against the quality criteria, AME alerts the labelling expert.

5.2 What a labelling expert can do from the AQUA interface?

Figure 1 shows the home page of the AQUA web interface for the labelling expert.

The navigation menu on the left is divided in three submenus ("My account", "Quality labelling", "The AQUA system"), each one giving access to different information and functionalities. We are going here to focus on "Quality labelling", as its links point straight to the heart of the AQUA system itself.


MedIEQ AQUA system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://medieq.iit.demokritos.gr:8080/aqua/aqua/welcome.seam Go Links
MedIEQ AQUA
Assisting Quality Assessment system
Go to MedIEQ.org
MedIEQ: Quality Labelling of Medical Web Content using Multilingual Information Extraction
MedIEQ
Assisting Quality Assessment
Internet

My account
- Edit my account
- Logout

Quality labelling
- My quality labels
- Tasks management
- Labels management
- Alerts management
- Formation of corpora
- Linguistic resources
- The quality criteria

The AQUA system
- About AQUA
- Contact the system administrators

Welcome to MedIEQ AQUA

Organisations around the world are working on establishing standards of quality in the accreditation of health-related web content. However the establishment of codes of conduct or ethics is not enough in the medical domain where the quality of information delivered from medical web sites may affect the health of the citizens. It is necessary to establish rating mechanisms, either by third party accreditation, or by creating portals where medical web sites are organised and characterised against certain labelling criteria.

In order for these mechanisms to be successful, they must be equipped with technologies that enable the automation of the rating process, such as information extraction techniques that allow the continuous monitoring of labelled web sites alerting the labelling agency in case some changes occur against the labelling criteria, or web crawling and spidering techniques that allow the retrieval of new unlabelled web sites, their characterisation and addition in a medical thematic portal.

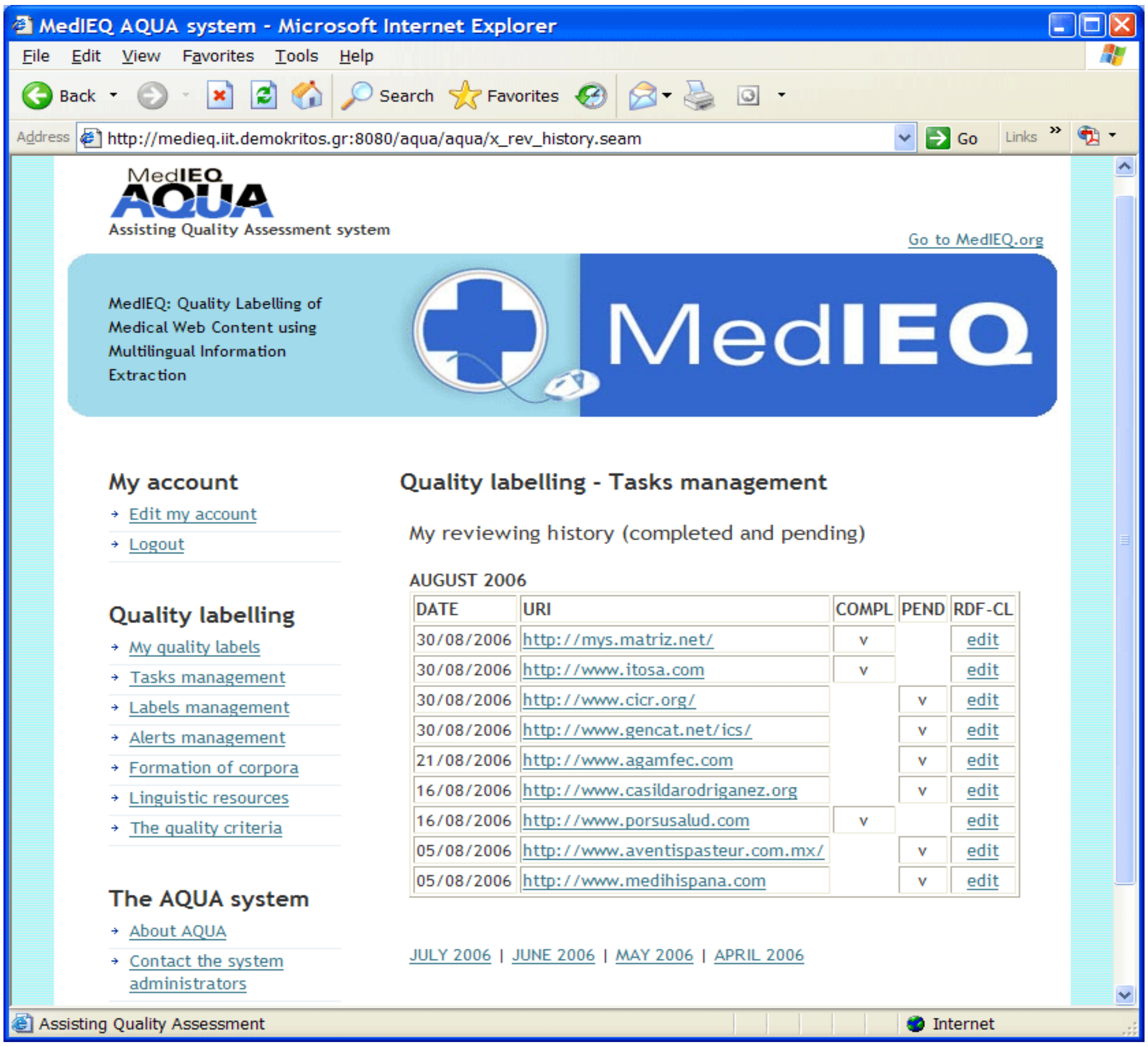
Based upon state-of-the-art technology in the areas of web crawling and spidering, multilingual information extraction, semantic resources and quality labelling, MedIEQ paves the way towards the automation of quality labelling process in medical web sites. MedIEQ will deliver tools that crawl the Web to

Figure 1: Home page of AQUA

Quality labelling > My quality labels

Here, the labelling expert sees a table listing the most recent URIs he/she has reviewed (also, links to previous listings are given at the bottom of the page, below this table, see fig.2). ‘COMPL’ indicates a completed quality label (where all label’s attributes have a value) while ‘PEND’ indicates incomplete (pending) labels.
A pending label for a URI usually means that, when the AQUA system ran for this URI, it didn’t find all values for all the label’s attributes[25] (however, it may also mean that the labelling expert while manually filled some of the attributes, left the rest unfilled).

[25] In any case, no one could expect a labelling assisting system to do all the work alone, without any human intervention. AQUA is supposed to identify and propose values for only a subset of the quality criteria a label consists of (those being machine processable) assisting thus the manual work of labelling experts.


MedIEQ AQUA system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://medieq.iit.demokritos.gr:8080/aqua/aqua/x_rev_history.seam
Go Links
MedIEQ AQUA
Assisting Quality Assessment system
Go to MedIEQ.org
MedIEQ: Quality Labelling of Medical Web Content using Multilingual Information Extraction
MedIEQ
My account
Edit my account
Logout
Quality labelling
My quality labels
Tasks management
Labels management
Alerts management
Formation of corpora
Linguistic resources
The quality criteria
The AQUA system
About AQUA
Contact the system administrators
Assisting Quality Assessment
Internet

Quality labelling - Tasks management

My reviewing history (completed and pending)

AUGUST 2006

DATE	URI	COMPL	PEND	RDF-CL
30/08/2006	http://mys.matriz.net/	v		edit
30/08/2006	http://www.itosa.com	v		edit
30/08/2006	http://www.cicr.org/		v	edit
30/08/2006	http://www.gencat.net/ics/		v	edit
21/08/2006	http://www.agamfec.com		v	edit
16/08/2006	http://www.casildarodriganez.org		v	edit
16/08/2006	http://www.porsusalud.com	v		edit
05/08/2006	http://www.aventispasteur.com.mx/		v	edit
05/08/2006	http://www.medihispana.com		v	edit

JULY 2006 | JUNE 2006 | MAY 2006 | APRIL 2006

Figure 2: My quality labels

<u>Quality labelling > Tasks management</u>

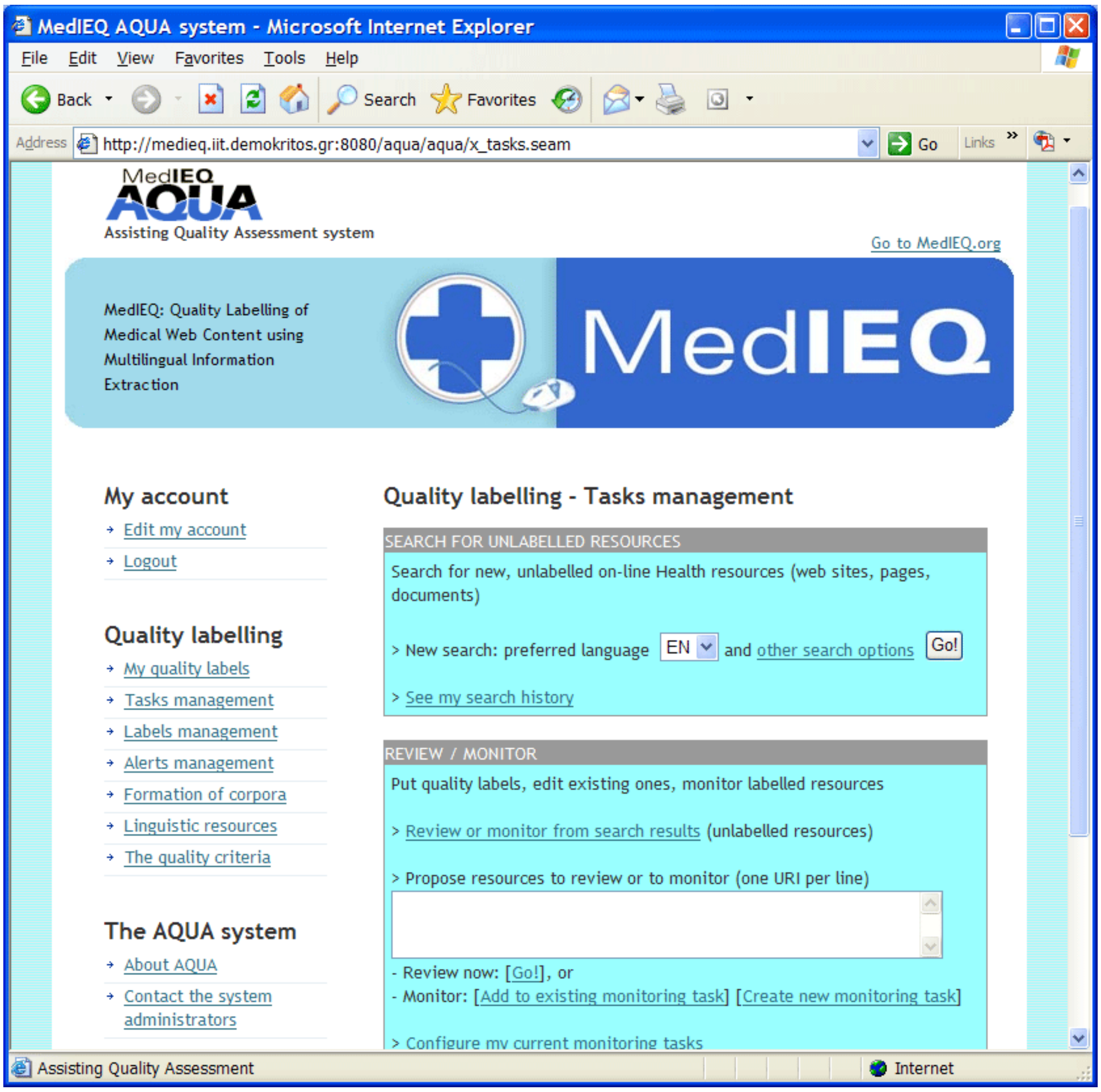
From this page (see fig.3) the labelling expert manages two major task-groups:

A) Search for new unlabelled web resources with health-related content, what is also called "Focused crawling", and
B) Review those unlabeled resources and/or monitor known, labelled ones.

In (A) the user can configure the MedIEQ search mechanism (the Crawler) by specifying the preferred language of the identified resources. Additionally, there is a second screen providing a number of other controls on how/where to search:

a) Has the Crawler to query known search engines? And if yes:
   - Which (from available) search engines?
   - After how many results the queries should be interrupted?
   - With which sets of keywords to query the search engines?
   - (*and many other parameters*)
b) Has the Crawler to use Web directories? And if yes:
   - Which Web directories to explore? and
   - How to explore them, by parsing down the entire directory tree or merely collect the containing URIs?
c) Is there a list of URIs to include in search results (white list)? or

d) Is there a list of hosts to ignore (black list)?


MedIEQ AQUA system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://medieq.iit.demokritos.gr:8080/aqua/aqua/x_tasks.seam
Go Links
MedIEQ AQUA
Assisting Quality Assessment system
Go to MedIEQ.org
MedIEQ
Assisting Quality Assessment
Internet

MedIEQ: Quality Labelling of Medical Web Content using Multilingual Information Extraction

My account
- Edit my account
- Logout

Quality labelling
- My quality labels
- Tasks management
- Labels management
- Alerts management
- Formation of corpora
- Linguistic resources
- The quality criteria

The AQUA system
- About AQUA
- Contact the system administrators

Quality labelling - Tasks management

SEARCH FOR UNLABELLED RESOURCES

Search for new, unlabelled on-line Health resources (web sites, pages, documents)

> New search: preferred language EN and other search options Go!

> See my search history

REVIEW / MONITOR

Put quality labels, edit existing ones, monitor labelled resources

> Review or monitor from search results (unlabelled resources)

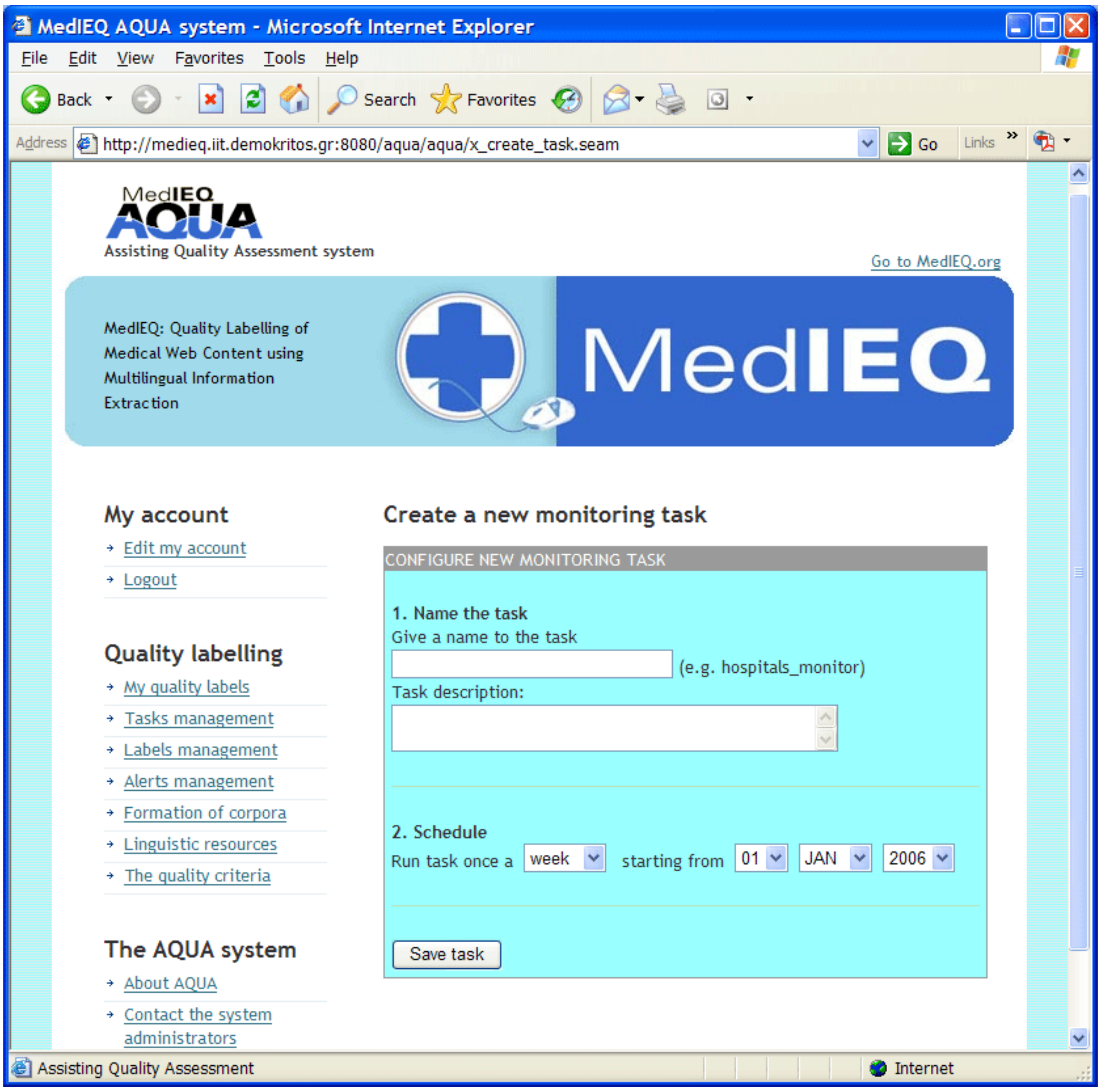
> Propose resources to review or to monitor (one URI per line)

- Review now: [Go!], or
- Monitor: [Add to existing monitoring task] [Create new monitoring task]

> Configure my current monitoring tasks

Figure 3: Tasks management

In (B) ("Review / monitor", fig. 3) the user can either access the results of the Crawler and start a reviewing or a monitoring task, or propose his/her own URIs to review or to monitor. In both cases, any URIs (if not set to be immediately reviewed) can be either added in an existing monitoring task or the user can create and configure a new monitoring task dedicated to these URIs (see fig. 4).


MedIEQ AQUA system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://medieq.iit.demokritos.gr:8080/aqua/aqua/x_create_task.seam
Go Links
MedIEQ AQUA
Assisting Quality Assessment system
Go to MedIEQ.org
MedIEQ: Quality Labelling of Medical Web Content using Multilingual Information Extraction
MedIEQ
My account
Edit my account
Logout
Quality labelling
My quality labels
Tasks management
Labels management
Alerts management
Formation of corpora
Linguistic resources
The quality criteria
The AQUA system
About AQUA
Contact the system administrators
Assisting Quality Assessment
Internet

Create a new monitoring task

CONFIGURE NEW MONITORING TASK

1. Name the task
Give a name to the task
(e.g. hospitals_monitor)
Task description:

2. Schedule
Run task once a week starting from 01 JAN 2006

Save task

Figure 4: Create a new monitoring task

Quality labelling > Labels management

The form that shows up (see fig.5) is split to four parts the first three involve user input for the actual label data and the last options for saving label. The Label Metadata sections accept the following type of input for each field:

Organization Name	any string
Mailbox address	valid email address of the form auser@adomain.com
Homepage , Authority for	a string representing a valid url e.g. http://mydomain.com
Issued date , Modification date	date format DD/MM/YYYY

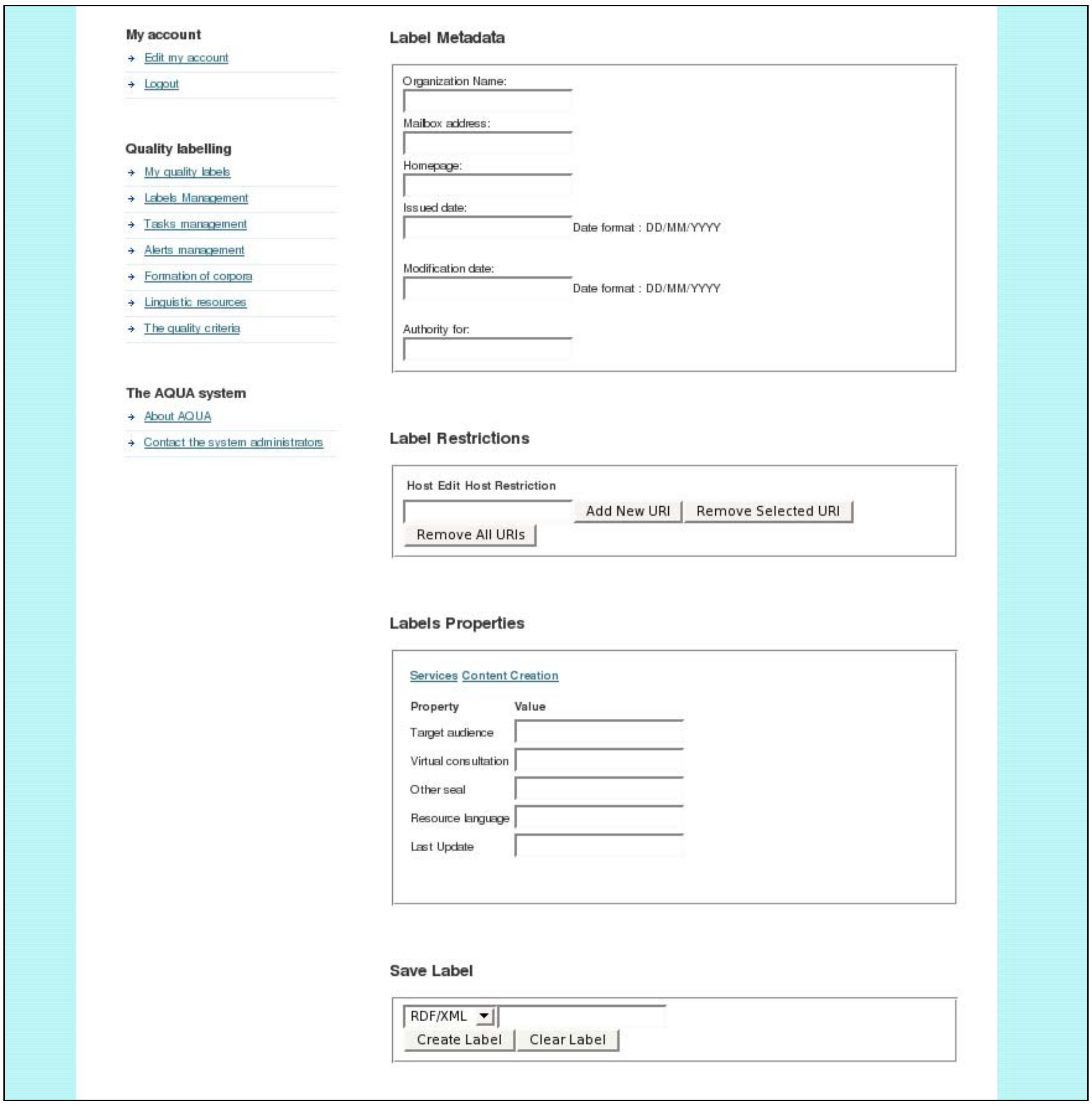


Figure 5: Generate a label

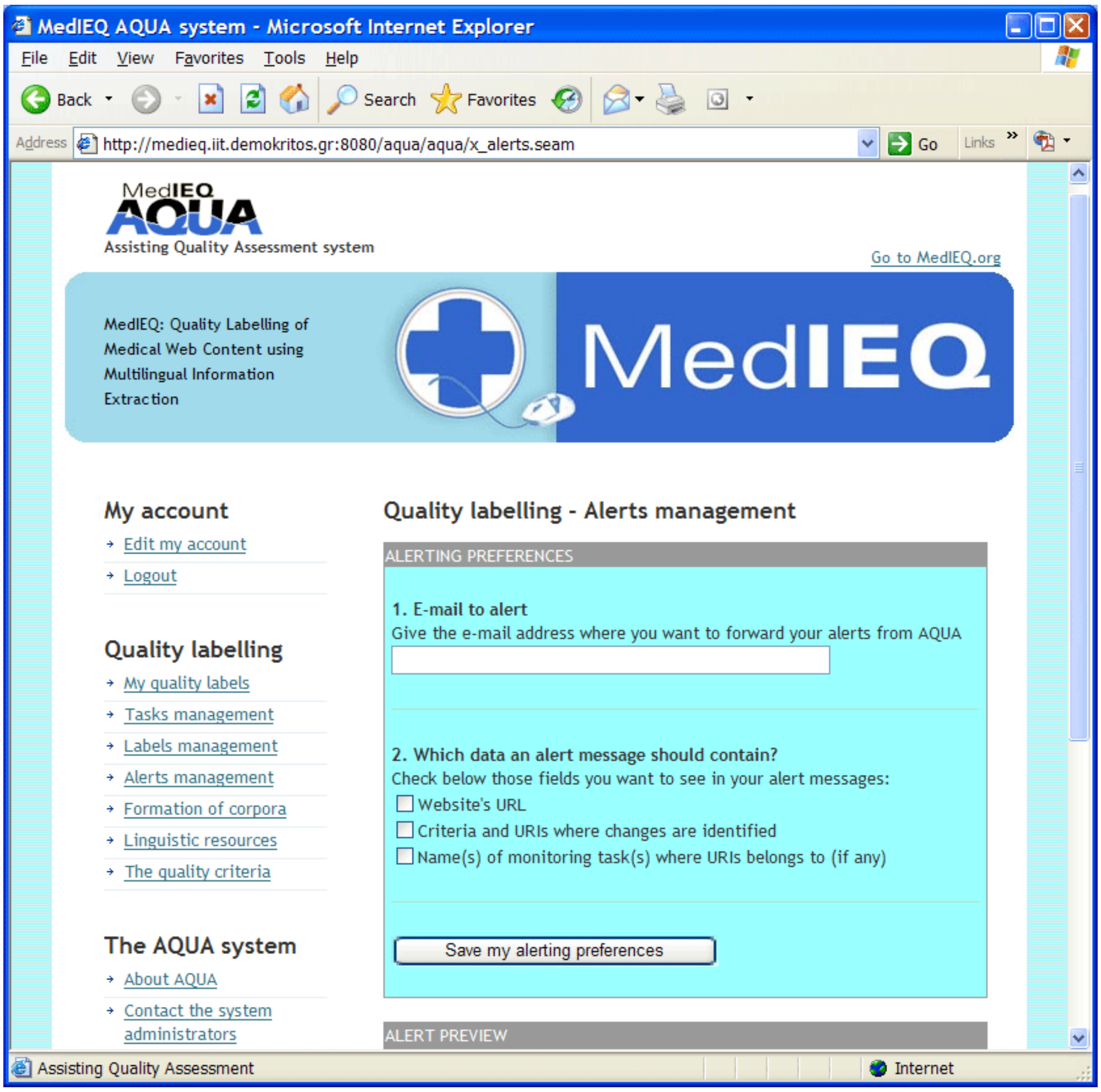
The Label Restrictions section holds a list of URLs that this label is restricted for. Inserting URLs to the input box and by pressing “Add new URI” button a list over the input box is populated. This list has an edit option to let the user change a URL. Once edit is clicked the URL is editable, the user can refill it, deselect the edit box and click the “Update Hosts” button to update the list.

Label Properties section contains a form that changes its content dynamically depending on the user selection. There is a tabbed pane with two links currently. Each link represents a different group of properties, clicking on each link displays the proper form.

The fourth part contains a select combo box that lets the user specify the file format he/she wishes to save the label; next to the combo box there is a text field for specifying the file name. Clicking “Clear label” clears all the fields in the form and “Create label” creates an RDF label, saves a copy to the server and lets the user download his/her own copy locally.

Quality labelling > Alerts management

Here, the user can configure his/her preferences concerning the alerts from AQUA (see fig.6). First, he/she has to fill the e-mail field (note that a user can give a different e-mail address for all communication / administration issues – through “My account” - and a second one for the AQUA alerts – through this form). Then, the user specifies the required structure of the alerting messages and saves its preferences. It is possible now to see a sample alert message in the text area at the bottom of the page by clicking on the “preview” button (if not happy, changes preferences, previews again and so on).


MedIEQ AQUA system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://medieq.iit.demokritos.gr:8080/aqua/aqua/x_alerts.seam Go Links
MedIEQ AQUA
Assisting Quality Assessment system
Go to MedIEQ.org
MedIEQ: Quality Labelling of Medical Web Content using Multilingual Information Extraction
MedIEQ
My account
Edit my account
Logout
Quality labelling
My quality labels
Tasks management
Labels management
Alerts management
Formation of corpora
Linguistic resources
The quality criteria
The AQUA system
About AQUA
Contact the system administrators
Assisting Quality Assessment
Internet

Quality labelling - Alerts management

ALERTING PREFERENCES

1. E-mail to alert
Give the e-mail address where you want to forward your alerts from AQUA

2. Which data an alert message should contain?
Check below those fields you want to see in your alert messages:
☐ Website's URL
☐ Criteria and URIs where changes are identified
☐ Name(s) of monitoring task(s) where URIs belongs to (if any)

Save my alerting preferences

ALERT PREVIEW

Figure 6: Alerts management

Quality labelling > Formation of corpora

This page will provide the UI for the MedIEQ corpus formation tool or CFT1 (the page is currently under construction as the early version of CFT1 has an AWT (Abstract Window Toolkit) interface; however, it is planned to be available soon).
With the aid of CFT1, the user forms collections of web pages / documents, stored in different directories, corresponding to the different categories specified. Such a collection is called a corpus. The corpora are necessary to train and test ML algorithms and, therefore, produce content classification and link scoring models (needed by the Web content collection toolkit).

Quality labelling > Linguistic resources

This page will give access to the management of the available (for the purposes of MedIEQ) multilingual resources (such as MeSH, ICD, etc.) This page also is currently under construction.

Quality labelling > The quality criteria

Finally, here we can see the first set of quality criteria proposed by the labelling authorities participating in MedIEQ.

5.3 What a system administrator can do from the AQUA interface?

Although the AQUA interface for the system administrator is still under development, what he will be able to do is well defined. The following table enumerates the actions for the system administrator as these are defined in Deliverables D6, D8 and D10; therefore, for more details on them, refer to these documents.

WP	**Toolkit**	**Actions**
WP5	WCC	Generate a CCM
		Generate a LSM
WP6	IET	Derive a new Extraction model (or edit existing)
		Add extraction knowledge to an Extraction model
		Add human expert extraction knowledge
		Add extraction knowledge from training data
		Define, run and evaluate an extraction task
WP7	MRM	Manipulate the available formats
		Manipulate the available encodings
		Manipulate the available languages
WP8	MUA	Define differences importance threshold

6. Adapt AQUA to a new language: estimating the cost

The adaptation to a new language is a two-step process:
Step 1: Adapt the AQUA web interfaces to the new language;
Step 2: Adapt the AQUA tools from all the participating toolkits to properly handle data in the new language.

We are here going to estimate how expensive such an adaptation may be in terms of human work.

Regarding the 1st step, as described in 2.2, the mechanism that handles the web interface Internationalization is provided from JSF and JBoss Seam. Plain text files contain all the Strings presented to the users. Each language has its own text file filled with lines containing variable names and their value Strings (messages), linked with the '=' character. Variable names are the same for each language; what changes is the value that is the message translation in each language and does not contain any presentation or layout notation. Variable names are used by our system internally to easily alter interfaces views across the supported languages. Therefore, when support of a new language is required, all that is needed is to have the translation of all variables values into the new language. As noted in the "1st Interim report: Technical Implementation Report", Appendix IV.1, to translate all language files in a new language, about 7 person days are estimated as necessary.

As for the 2nd step, AQUA's open architecture requires that several components from all the AQUA subsystems will run as independent services deployed in the application server where the system runs. A proper abstraction layer has been created to enable the system to accept, integrate and use such new services as alternative implementations of existing ones, e.g. any web crawler respecting MedIEQ Crawler specifications could easily replace the proper AQUA Crawler. Such modularity needs led us to specify tools having the minimum possible dependence from speaking languages. We also opted for UTF-8: it is used in every internal process where text is needed, in every downloaded document, in all information extracted and data saved in the MedIEQ database.

Therefore, to estimate the work needed in tools' customization to a new language, all we need is to aggregate (again from the "1st Interim report: Technical Implementation Report") the efforts reported for each toolkit.

LAM:	2-3 PDs
Crawler:	3-4 PDs
CFT:	1 PD
Spider:	1 PD
LSC:	0 PDs (language independent)
CCC:	0 PDs (language independent)
TMG:	0 PDs (language independent)
IET:	5 PDs
Ex:	30 PDs
MRM:	0 PDs (language independent)
MUA:	0 PDs (part of AQUA user interfaces language customization)
TOTAL:	44 PDs

The above calculation does not include the effort required to form (using the CFT and the LOFT tools) the necessary training/testing corpora in the new language, which is estimated at about 10 PDs.

Concluding, the migration of the AQUA interfaces and all the AQUA components in a new language would require:

AQUA user interfaces:	7 PDs
AQUA tools:	44 PDs
Formation of corpora:	10 PDs
TOTAL:	61 PDs

It has to be mentioned that the above estimations are made upon the 1st version of the tools and the user interfaces. However, foreseen tool improvements may further shorten that time.

7. Concluding remarks

The development and implementation of the overall MedIEQ system took into account specific design requirements such as:

1. Modularity: modules of the original system should be easily replaceable by other modules (respecting certain specifications), possibly developed outside MedIEQ, who would do similar work with the original modules.
2. Internationalization: both user interfaces and software of the MedIEQ system should ensure support for multiple languages.
3. Accessibility: the MedIEQ users (mainly labelling experts from labelling agencies) should access MedIEQ system easily, at anytime and from anywhere; what else if not a web based system would better support that?
4. Performance: requiring both run-time performance and effectiveness in the maintenance and control of the MedIEQ system, we opted for a centralized solution.

Therefore, the resulting architecture is open and the foreseen integrated system (its 1st version is due for M18 or June 2007) will be modular (a design requirement we repeatedly support in all our documents already since the MedIEQ project proposal).

Additionally, customization of the MedIEQ system to new languages, as this was calculated for its forthcoming 1st version, is not too expensive in terms of human work: 61 PDs (person days) are estimated to be enough to totally adapt all interfaces and software tools into a new language. This estimation will be evaluated in practice when customising the MedIEQ systems to the rest languages of the project (English and Spanish are to be supported in the 1st version).

The next steps in the development of the MedIEQ system are the following:

- Implement the user interfaces for the system administrator;
- Properly join all tools functionalities with relevant user interface controls;
- Resolve possible communication malfunctions;
- Tackle performance issues and suggest usability improvements taking into account results from scheduled future evaluation sessions and feedback from labelling experts having tested the MedIEQ system in practice.

APPENDIX A: AQUA toolkits & tools

In the table below, we see all AQUA toolkits and tools per Work Package (WP). The following legend explains all toolkits acronyms.

WP	TOOLKIT	TOOL ABBREV.	TOOL DESCRIPTION
WP7	**MRM**	**RM**	Resources Manager
		Generic parser	Generic parser
		Specific parsers	Specific parsers
		Browser	Browser
		Converter	Converter
WP5	**WCC**	**Crawler**	Crawler
		Spider	Spider
		CFT1 and CFT2[26]	Corpus Formation Tools
		LOFT[27]	Link Objects Formation tool
		TMG	Train Module Generator
		CCC	Content Classification Component
		LSC	Link Scoring Component
WP6	**IET**	**DMM**	Data Model Manager
		TM	Task Manager
		AT	Annotration tool
		Preprocessor	Preprocessor
		NER (IEEngine A)	Named Entity Recognition
		EXO (IEEngine B)	Extraction Ontologies
		ML (IEEngine C)	Machine Learning
		STA (IEEngine D)	Statistical methods
		Integrator	Integrator
		Visualiser	Visualiser
WP8	**MUA**	**MSC**	Monitoring scheduler
		DBU	Database updater
		AME	Alerting mechanism
WP4	**LAM**	**RDF-G**	RDF generator
		RDF-C	RDF comparator

Toolkits legend
MRM: Multilingual Resources Management
WCC: Web Content Collection
IET: Information Extraction Toolkit
MUA: Monitor - Update - Alert
LAM: Label Management

[26] There are two corpus formation tools:
- CFT1 developed by NCSR (available through AQUA) and
- CFT2 or Scrapbook, a plug-in for Mozilla (available at http://amb.vis.ne.jp/mozilla/scrapbook/)

[27] LOFT forms the link objects necessary for training / testing the link scoring models (needed for the link scoring mechanism employed in spidering).

APPENDIX B: MedIEQ metadata

1. Metadata on Web resources (doc or site) – General metadata
MedIEQ database > Table: Webs

WebID (PK)	An id automatically assigned by the system
WebType	(< WebTypeID from WebTypes) [doc | site]
WebCategory	(< WebCategoryID from WebCategories) [Government Organization | Healthcare service provider | Media and publishers | Pharmaceutical company / retailer | Universities / research institutions | Scientific or professional organizations | Patient organizations / self-support groups | Private individual]
WebBaseDir	The path to a local directory containing all files stored from the specific web, as well as other files with metadata information and/or configuration files
WebURI	The URI of the Web; may be the URI of an entire (e.g. http://www.foo.org/)or a part of a web site (e.g. http://www.foo.org/health) or URI of a webpage or document (e.g. http://www.foo.org/health/abc.html)
WebTitle	The title of the Web
DateOfCreation	The date the Web was first introduced in AQUA

2. Metadata on Files (or Document)
MedIEQ database > Table: Files

FileID (PK)	An id automatically assigned by the system
FilePurpose	(< PurposeTypeID from PurposeTypes) [ie | cc | ls]
FileType	(< FileTypeID from FileTypes) [text/html | doc | pdf]
FileContentType	(< ContentTypeID from ContentTypes) [about | advert | article | contact | health | policy | vc]
FileLanguage	(< LanguageID from Languages) [en | es | gr | de | cz | ca | fi]
FileParent	(< FileID from Files – self-relationship)
FileName	The filename given by the system when storing a file
FileLocation	The location of the file in the local storage
FileEncoding	The original encoding of the file
FileContentHash	A hash (e.g. SHA 1) is calculated corresponding to the original status of the content of a file
WebURI	File’s web address (e.g. http://abc.com/10/1.html)
BaseURI	File’s base address (e.g. http://abc.com/10/)
HostURI	File’s host (e.g. http://abc.com/)

3. Metadata on Extracted Information
MedIEQ database > Table: ExtractedInformation

XID (PK)	An id automatically assigned by the system
XFile	(< FileID from Files)
XCategory	(< XCategoryID from XCategories) [PurposeOfWeb | PurposeOfOwner | TargetAudience | ProvidedInfoLimitationDisclaimer | OwnerOrganization | OwnerOrganizationType | WebResponsible | Webmaster | MeSHTerms | MeSHTopics | VCServicePresent | VCResponsible | VCLimitationDisclaimer | AdvertisingPresent | AdvertisingSeparated |

	AdvertisingPolicy | AdvertisingStatement | OtherSealsPresent | OtherSealsList | BibliographyPresent | PublicationDate | LastModifiationDate | Author | ProvidedInfoResponsibilityDisclaimer | FundingSourcesClear | FundingSourcesLimitationDisclaimer | SponsorOrganization | SponsorOrganizationType | CPSPolicyPresent | CPSPolicyOnPersonalData | AccessibilityLevel]
XType	(< XTypeID from XTypes) [freetext | personName | organizationName | organizationType | address | region | city | country | phone | fax | email | Boolean | date]
XStatus	(< StatusTypeID from StatusTypes) [pending | completed]
XDate	The date of the information extraction
XData	The extracted information
XStart	The starting offset of the extracted information
XEnd	The ending offset of the extracted information

4. Metadata on Users
MedIEQ database > Table: Users

UserID (PK)	An id automatically assigned by the system
UserType	(< UserTypeID from UserTypes) [admin | expert | aqua]
OrganizationName	(<OrganizationNameID from OrganizationNames) [aqumed | wma | other]
UserName	The username specified by the user
UserPass	The password specified by the user
UserFirstName	User’s real first name
UserLastName	User’s real last name
UserContactEmail	User’s contact e-mail address
UserBaseDir	The path to a local directory (automatically created by the system) containing all information (other than the information of an entry of this table, e.g. preferences files, configuration files etc.) corresponding to the specific user

5. Metadata on Labels
MedIEQ database > Table: Labels

LabelID (PK)	An id automatically assigned by the system
LabelCreator	(< UserID from Users)
LabelStatus	(< StatusTypeID from StatusTypes)
LabelInitiationDate	The date the user initiated the label
LabelCompletionDate	The date the user completed the label
LabelHash	The hash of the label

6. Metadata on Corpora
MedIEQ database > Table: Corpora

CorpusID (PK)	An id automatically assigned by the system
CorpusCreator	(< UserID from Users)
CorpusPurposeType	(< PurposeTypeID from PurposeTypes) [cc | ls]
CorpusType	(<ContentTypeID from ContentTypes) [about | advert | article | contact | health | policy | vc]
CorpusLanguage	(<LanguageID from Languages) [en | es | gr | de | cz | ca | fi]

CorpusName	A name given by the user
CorpusDescription	A description given by the user
CorpusCategories	The categories of the corpus (e.g. pos|neg)
CorpusBase	The path to the local dir containing everything related to the specific corpus
CorpusSize	The size of the corpus (number of sample files)
CorpusDistribution	The distribution in the different categories (e.g. pos=350, neg=450)
CorpusCreationDate	The date the corpus was formed
CorpusLastModificationDate	The date the corpus was last modified

7. Metadata on AlertProfiles

MedIEQ database > Table: AlertProfiles

AlertProfileID (PK)	An id automatically assigned by the system
AlertProfileCreator	(< UserID from Users)
AlertPeriod	(<PeriodTypeID from PeriodTypes) [week | month | year]
AlertEmail	User's alert e-mail address
AlertStartDate	The date the alert profile becomes activate
AlertMessageParameters	The parameters an alert message should contain (*will probably become a separate table*)

8. Metadata on Tasks

MedIEQ database > Table: Tasks

TaskID (PK)	An id automatically assigned by the system
TaskOwner	The user who created the task; UserID from Users table
TaskPeriod	The time interval between consecutive runs (PeriodTypeID from PeriodTypes table, value range: week | month | year)
TaskName	The name the user gave to the task
TaskDescription	The description of the task
TaskBaseDir	A directory in the local disk (automatically created by the system) containing all those files (configuration, task outputs, etc.) needed for the task
TaskType	The type of the task: either a searching-for-unlabelled-health-resources or a reviewing-health-resources task (TaskTypeID from TaskTypes table, value range: search-t | review-t)
TaskCreationDate	The date when the task was created
TaskStartDate	The date the task is scheduled to initialize
TaskEndDate	The date the task is scheduled to cease
TaskIsActive	Whether the task is active or not

The following are auxiliary tables, containing the restricted values for several fields of the above 8 tables.

WebTypes

doc
site

WebCategories

Government Organization
Healthcare service provider

Media and publishers
Pharmaceutical company / retailer
Universities / research institutions
Scientific or professional organizations
Patient organizations / self-support groups
Private individual

AudienceTypes

prof
adult
child

SealTypes

hon
wma
pwmc
urac
eHealth trust-e
afgis
aqumed
omni
who

AccessibilityLevels

A
AA
AAA

FileTypes

text/html
doc
pdf

ContentTypes

about
advert
article
contact
health
policy
vc

PurposeTypes

ie
cc
ls

StatusTypes

pending
completed

LanguageTypes

en

es
gr
de
cz
ca
fi

UserTypes

expert
admin
aqua

OrganizationNames

aqumed
wma
other

OrganizationTypes

la
other

PeriodTypes

week
month
year

TaskTypes

Search-t
Review-t

XCategories

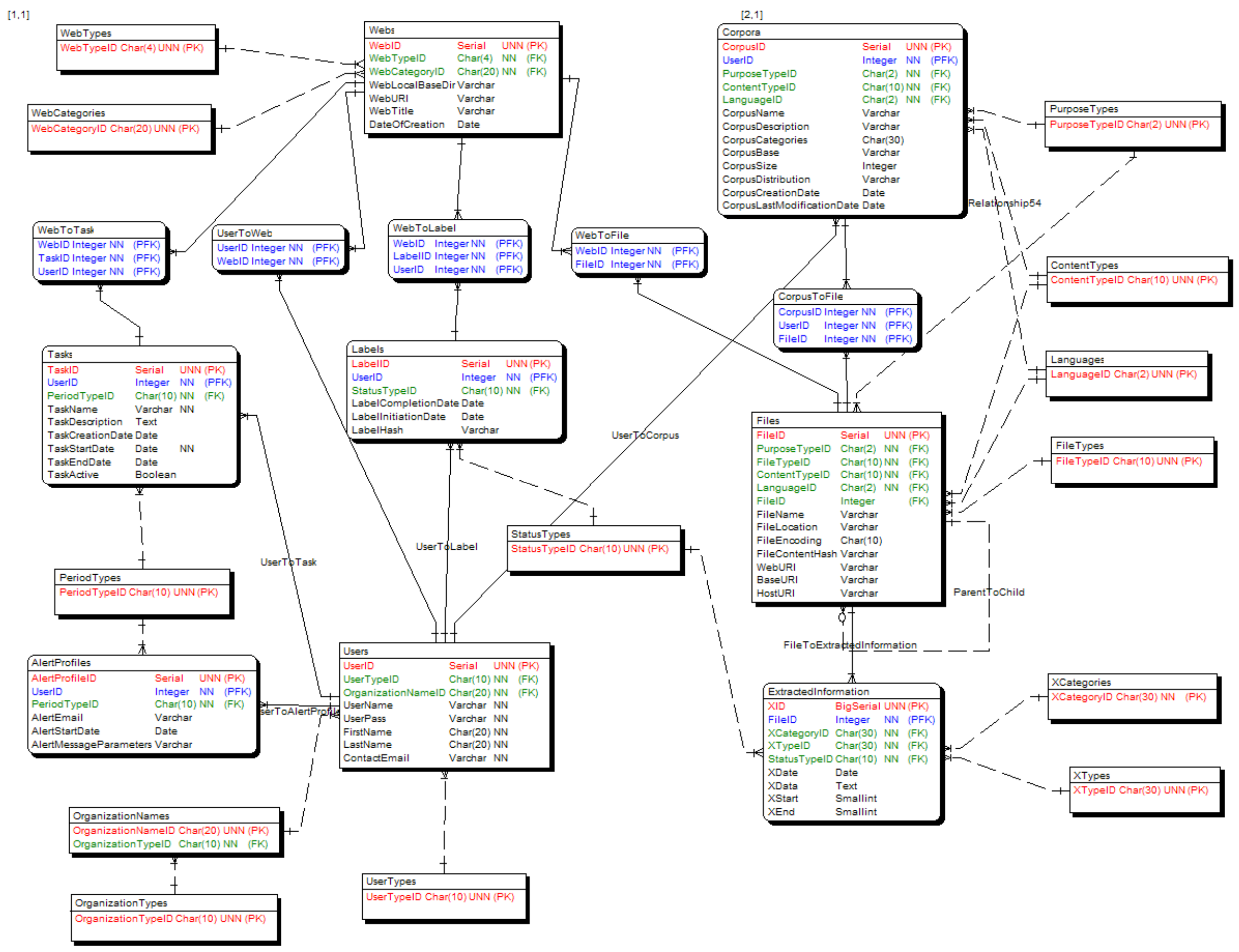
PurposeOfWeb
PurposeOfOwner
TargetAudience
ProvidedInfoLimitationDisclaimer
OwnerOrganization
OwnerOrganizationType
WebResponsible
Webmaster
MeSHTerms
MeSHTopics
VCServicePresent
VCResponsible
VCLimitationDisclaimer
AdvertisingPresent
AdvertisingSeparated
AdvertisingPolicy
AdvertisingStatement
OtherSealsPresent
OtherSealsList
BibliographyPresent
PublicationDate
LastModifiationDate

Author
ProvidedInfoResponsibilityDisclaimer
FundingSourcesClear
FundingSourcesLimitationDisclaimer
SponsorOrganization
SponsorOrganizationType
CPSPolicyPresent
CPSPolicyOnPersonalData
AccessibilityLevel

XTypes

organizationName
organizationType
personName
address
region
city
country
phone
fax
email
freetext
boolean
date

APPENDIX C: The schema of the MedIEQ database


[1,1]
WebTypes
WebTypeID Char(4) UNN (PK)
WebCategories
WebCategoryID Char(20) UNN (PK)
Webs
WebID Serial UNN (PK)
WebTypeID Char(4) NN (FK)
WebCategoryID Char(20) NN (FK)
WebLocalBaseDir Varchar
WebURI Varchar
WebTitle Varchar
DateOfCreation Date
[2,1]
Corpora
CorpusID Serial UNN (PK)
UserID Integer NN (PFK)
PurposeTypeID Char(2) NN (FK)
ContentTypeID Char(10) NN (FK)
LanguageID Char(2) NN (FK)
CorpusName Varchar
CorpusDescription Varchar
CorpusCategories Char(30)
CorpusBase Varchar
CorpusSize Integer
CorpusDistribution Varchar
CorpusCreationDate Date
CorpusLastModificationDate Date
PurposeTypes
PurposeTypeID Char(2) UNN (PK)
Relationship54
WebToTask
WebID Integer NN (PFK)
TaskID Integer NN (PFK)
UserID Integer NN (PFK)
UserToWeb
UserID Integer NN (PFK)
WebID Integer NN (PFK)
WebToLabel
WebID Integer NN (PFK)
LabelID Integer NN (PFK)
UserID Integer NN (PFK)
WebToFile
WebID Integer NN (PFK)
FileID Integer NN (PFK)
ContentTypes
ContentTypeID Char(10) UNN (PK)
CorpusToFile
CorpusID Integer NN (PFK)
UserID Integer NN (PFK)
FileID Integer NN (PFK)
Languages
LanguageID Char(2) UNN (PK)
Tasks
TaskID Serial UNN (PK)
UserID Integer NN (PFK)
PeriodTypeID Char(10) NN (FK)
TaskName Varchar NN
TaskDescription Text
TaskCreationDate Date
TaskStartDate Date NN
TaskEndDate Date
TaskActive Boolean
Labels
LabelID Serial UNN (PK)
UserID Integer NN (PFK)
StatusTypeID Char(10) NN (FK)
LabelCompletionDate Date
LabelInitiationDate Date
LabelHash Varchar
UserToCorpus
Files
FileID Serial UNN (PK)
PurposeTypeID Char(2) NN (FK)
FileTypeID Char(10) NN (FK)
ContentTypeID Char(10) NN (FK)
LanguageID Char(2) NN (FK)
FileID Integer (FK)
FileName Varchar
FileLocation Varchar
FileEncoding Char(10)
FileContentHash Varchar
WebURI Varchar
BaseURI Varchar
HostURI Varchar
FileTypes
FileTypeID Char(10) UNN (PK)
StatusTypes
StatusTypeID Char(10) UNN (PK)
UserToLabel
UserToTask
PeriodTypes
PeriodTypeID Char(10) UNN (PK)
ParentToChild
FileToExtractedInformation
AlertProfiles
AlertProfileID Serial UNN (PK)
UserID Integer NN (PFK)
PeriodTypeID Char(10) NN (FK)
AlertEmail Varchar
AlertStartDate Date
AlertMessageParameters Varchar
serToAlertProfile
Users
UserID Serial UNN (PK)
UserTypeID Char(10) NN (FK)
OrganizationNameID Char(20) NN (FK)
UserName Varchar NN
UserPass Varchar NN
FirstName Char(20) NN
LastName Char(20) NN
ContactEmail Varchar NN
XCategories
XCategoryID Char(30) NN (PK)
ExtractedInformation
XID BigSerial UNN (PK)
FileID Integer NN (PFK)
XCategoryID Char(30) NN (FK)
XTypeID Char(30) NN (FK)
StatusTypeID Char(10) NN (FK)
XDate Date
XData Text
XStart Smallint
XEnd Smallint
XTypes
XTypeID Char(30) UNN (PK)
OrganizationNames
OrganizationNameID Char(20) UNN (PK)
OrganizationTypeID Char(10) NN (FK)
UserTypes
UserTypeID Char(10) UNN (PK)
OrganizationTypes
OrganizationTypeID Char(10) UNN (PK)