Open Source Implementation of the NIST HealthCare Standards Landscape

Hai Tang, Roy Morgan, Thomas Rhodes, Clement Ridoret

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg MD 20899-8970

Abstract -- *The NIST HealthCare Standards Landscape (HCSL) is a project to develop and demonstrate a web-based repository of information (called "Landscape") on healthcare informatics standards and other related information. Users may navigate the HCSL web site, search and retrieve information about healthcare standards, about their development organization(s), about organizations adopting and using these standards, and other relevant healthcare standards' information. The HCSL was originally developed using an Oracle database on a Windows platform. To enable wider adoption of the HCSL application, it has also been implemented using open source software with PostgreSQL and MySQL databases on a Linux platform. This paper describes the software implementation and the lessons learned in porting to open source databases.*

The HCSL was built as a 3-tier web-based application providing user and content access controls. It may be adopted for other applications by modifying the software and database tables accordingly.

Keywords: open source software; health information technology; healthcare standards; standards repository; standards; web applications.

1 Introduction

The healthcare (HC) industry has many standards development organizations (SDOs) developing specifications and standards for healthcare informatics and information exchange covering a wide spectrum of HC activities. The many development efforts and large number of healthcare standards that exist or are in development, make it very difficult to monitor and track the overall healthcare standards landscape.

This in turn impedes harmonization efforts among SDOs and frustrates efforts by users and organizations to identify, understand, and adopt needed standards. By improving the availability and dissemination of healthcare standards information, many of these problems can be relieved and collaboration and implementation efforts can be improved among developers, implementers, and users of healthcare standards.

The National Institute of Standards and Technology (NIST) has developed a prototype web-based infrastructure and repository of healthcare informatics standards' information. This repository is referred to as the HealthCare Standards Landscape (HCSL) or simply as the "Landscape." It contains information and links to healthcare standards, standards development organizations, organizations that adopt and use these standards, and other relevant healthcare standards information.

NIST has been working with HC standards developers and stakeholders to define, develop, test, and demonstrate a prototype HCSL capability. This includes working with the ANSI Healthcare Information Technology Standards Panel (HITSP) [1], the Agency for Health Research and Quality (AHRQ)[2], the e-Gov Consolidated

Health Informatics (CHI) [3] program, the Health Information and Management Systems Society (HIMSS) [4] as well as monitoring and drawing from other SDOs and HC organizations, e.g. American Dental Association (ADA) [5], HL7 [6], the National Council for Prescription Drug Programs (NCPDP) [7], and the Institute of Electrical and Electronics Engineers (IEEE-1073) [8]. The HCSL prototype, now available on the Web uses actual data to demonstrate its overall capabilities. It has been made available for external partners to test and evaluate as well as to contribute additional HC standards' information into the HCSL repository. Subsequently, the open source HCSL software and data will be available for downloading and use by organizations that wish to install and operate their own version of the HCSL.

The HCSL repository was originally developed using an Oracle database and a Windows platform. However, to provide portability to other operating environments and database systems, an effort was undertaken to convert the HCSL software to open source. After some consideration, we proceeded to port the HCSL to an open source environment based on the PostgreSQL and MySQL databases and the Linux platform. The resulting open source software is also applicable to Oracle on Windows platform. Moreover, the open source HCSL retains all functionality of the original HCSL. The difference between the original HCSL and the open source HCSL lies in the Java servlet programs. There is no difference in architecture and HCSL functions, except that the open source has additional Jsp programs for porting from a database to another.

The open source implementation of HCSL adopts the "only once" concept from Java's "write only once and run everywhere", and the relational database concept of "storing data only once" principle. Software used in the original HCSL codes are being reorganized to make the code logically compact and to minimize programming changes when porting to different databases and to eliminate redundant code. Proprietary Oracle codes that retrieved documents in XML format are replaced with Java programs that are applicable to all three databases.

The HCSL was built as a 3-tier web-based application providing user and content access controls as well as the capability for establishing and managing groups and subgroups of users and contents. It may be adopted for other applications by modifying the software and database tables accordingly. This paper describes the software implementation experience and what has been changed to implement the HCSL with open source software and databases. It also describes the steps to implement an open source HCSL and the contents of HCSL from a programmer and system perspective. This paper presents the main ideas and concepts; it does not provide details of all the changes.

2 Types of Information in HCSL

The HCSL uses a relational database as a repository to contain information about health informatics standards and related information. It does not include the actual standards themselves. Currently, the HCSL repository contains three main types of information objects described below:

- Document: e.g., specifications, standards, regulations, guidelines, implementation guides, and code lists.
- Organization: e.g., development organizations, consortia, government groups, user groups, and conformance testing providers.
- Portfolio: e.g., collections of Documents developed, maintained, or used by an Organization. To illustrate, the group of standards consisting of HL7 Messaging, DICOM, and LOINC are part of the Portfolios for the Connecting-for-Health (C-f-H) and the Consolidated Health Informatics (CHI) organizations.

These three types of information objects (Document, Organization, Portfolio) form the basis for the current 'information model' of the HCSL.

In addition, the HCSL contains the relationship between pairs of these information objects. Some examples are: an Organization develops or uses a Document (perhaps a standard or code list); a Document is published by an Organization; a Document references another Document.

3 HCSL Architecture

HCSL is designed as a web-based system for publishing and finding information on health informatics standards and organizations. In addition to being an information repository, the HCSL provides administrative controls to add and manage contents, users and groups.

The HCSL was originally developed using Tomcat Servlet [9] container with connection to an Oracle database. The open source HCSL retains all functionality of the original HCSL and expands to open source databases. Tomcat, PostgreSQL, and MySQL are chosen because they are open source and freely available under GPL or compatible license and they have large user communities.

The HCSL is a 3-tier client/server system. Except for the database software that resides in the database server, all software components reside in the web server. Figure 1 depicts the open source HCSL architecture.

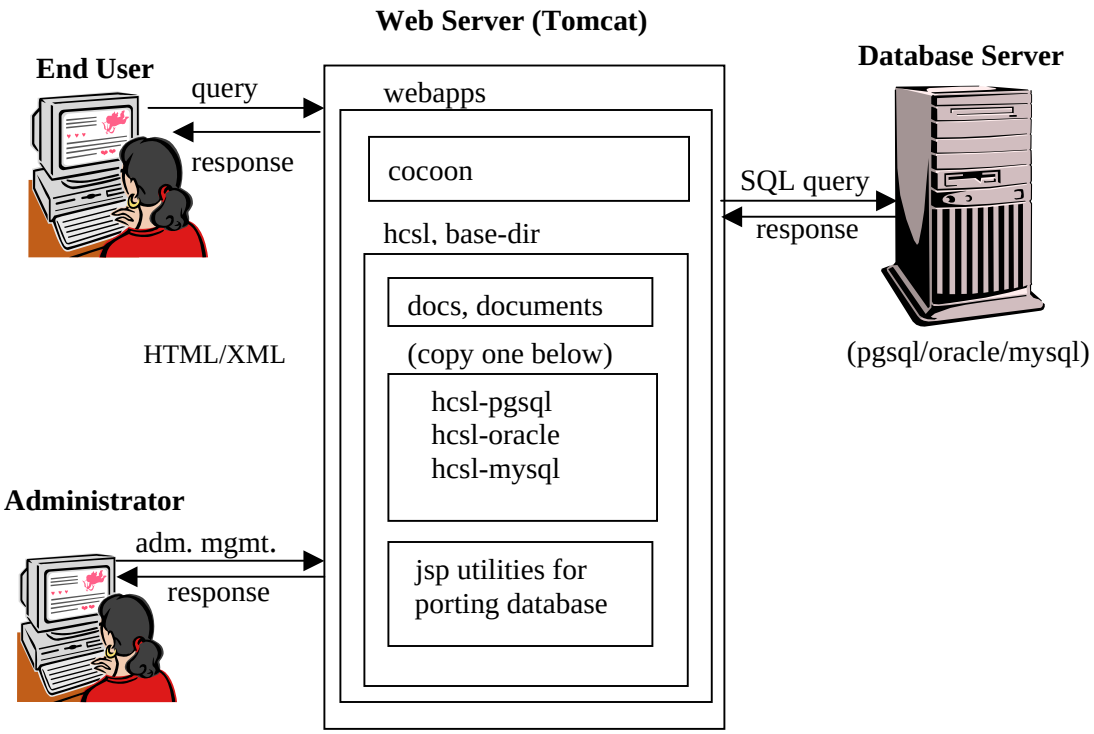

Figure 1. Open Source HCSL Architecture

Access control in the HCSL is applied to update or delete individual records in the database tables. Although database servers also provide such access control, the access control code is not portable from one database to another. Thus, the open source HCSL access control is implemented in the web sever and is independent of individual databases.

The Tomcat web server and the database server may reside on separate computers; both servers must be installed first. The HCSL software package is then placed within the Tomcat webapps directory. Before starting HCSL on a web browser, both servers must be running on respective servers.

4 Open Source Implementation

The open source implementation of HCSL adopts the "only once" concept from Java's "write only once and run everywhere" and relational database's "storing data only once" principle. Original HCSL codes are reorganized to make the code logically compact and to easily port to different databases and redundant codes are eliminated. Proprietary Oracle code to retrieve documents in XML format are replaced with Java programs to be applicable to PostgreSQL and MySQL, as well as to Oracle.

We have configured the open source HCSL to handle any of the three databases – Oracle, PostgreSQL, or MySQL and allow implementers to choose their database. To enable a choice of which database to use, the connection between web server and database is kept in a file called "DBConnection.java". Thus, an implementer can select a specific database by commenting and un-commenting lines of code within the "DBConnection.java" file. As the HCSL codes for different databases are stored in separate directories in Tomcat webapps (See Figure. 1), these directories are defined as "hcsl_dir" in "DBConnection.java" and read with "getHcslDir" method. Specific driver imports (ojdbc14.jar [10], postgresql-8.0-312.jdbc3.jar [11], and mysql-connector-java-3.1.12-bin.jar [12]) are kept in respective WEB-INF/lib directories. Thus, besides the specific database driver that needs to be stored in the respective WEB-INF/lib directory, DBConnection.java is the only place to be changed for the database to be used.

An individual implementation of open source HCSL needs only to select and use a single database whether it is a PostgreSQL, an Oracle, or a MySQL and to copy the respective database directory (hcsl-pgsql, hcsl-oracle, or hcsl-mysql) into “hcsl” as the base-directory in Tomcat “webapps”. The database driver needed for use with the chosen database is already placed in subdirectory WEB-INF/lib and the “DBConnection.java” program has been changed, compiled and placed in subdirectory WEB-INF/classes.

In order to support all databases with uniform code, Oracle proprietary getResponseXML method, OracleXMLDocGen, and OracleXMLQuery, which are used in GetXMLBean.java to get an XML document, are replaced with a modified getXML method in GetXMLBean.java. The same code works with PostgreSQL and MySQL, as well as Oracle.

Similarly, data input to the Oracle database from an XML document, was handled by InsertXMLBean that uses Oracle proprietary methods is modified to provide a uniform interface to all databases. The modified InsertXMLBean reads in only valid elements for a given table from an XML document and inserts them as record(s) into the table in the database; non-value elements are set to empty string.

Occasionally, “Null Pointer” exceptions would occur whenever an attempt was made to retrieve data items that were not defined in the database. This was handled using a “try { ...} catch {NullPointerException npe}{ ...}” code pair, as shown below:

```
try {
    xmlStr += rs.getObject(i).toString();
} catch (NullPointerException npe) {
    xmlStr += "";
}
```

We had to pay particular attention to variable names and the scope of variables to avoid problems during execution. As is well known, variables inside functions should always be defined, so that they stay local and do not default or interfere with variable in the calling function, having the same name. Even though Java is a strong typed language that will throw an undefined symbol error message, it defaults to the same variable if it happens to be defined in calling function. For example, overlooking such an index variable in a for-loop has caused infinite looping.

5 HCSL Components

5.1 HCSL Web Browser

The original HCSL public system is accessed through a client Web Browser (e.g. Netscape, Mozilla Firefox, or Internet Explorer) using the URL http://www.nist.gov/hcsl/ in the web browser.

On the Home page, there is a navigation menu on the left hand side that provides access to various HCSL functions. The menu consists of following items:

- Home
- Login/Logout
- Register
- Search Content
- User Support
- Background
- Related Activities

Each menu item is implemented either in html or as a java servlet. Some of the html files also include embedded JavaScript codes; thus, JavaScript must be enabled in the browser. This paper describes how these functions are implemented and only provides a cursory description of their usage.

Anyone can browse and search the contents of the HCSL repository without a login. However, to add, update, or delete content, a user must be registered with a login-name and password, and be a member of one or more groups or subgroups. When a registered user logs in to a group or subgroup, the menu changes to offer a "Member Area" selection, which allows the user to manage their content within the group or subgroup to which they belong. The user can then add new content or update existing content, and may also be able to update content created or “owned” by other group members depending on the assigned access privilege. The administrator of a group can also manage the membership, contents, and subgroups within the group.

5.2 Search and Update Functions

Figure 2 depicts the main program modules for the content search and update pages.

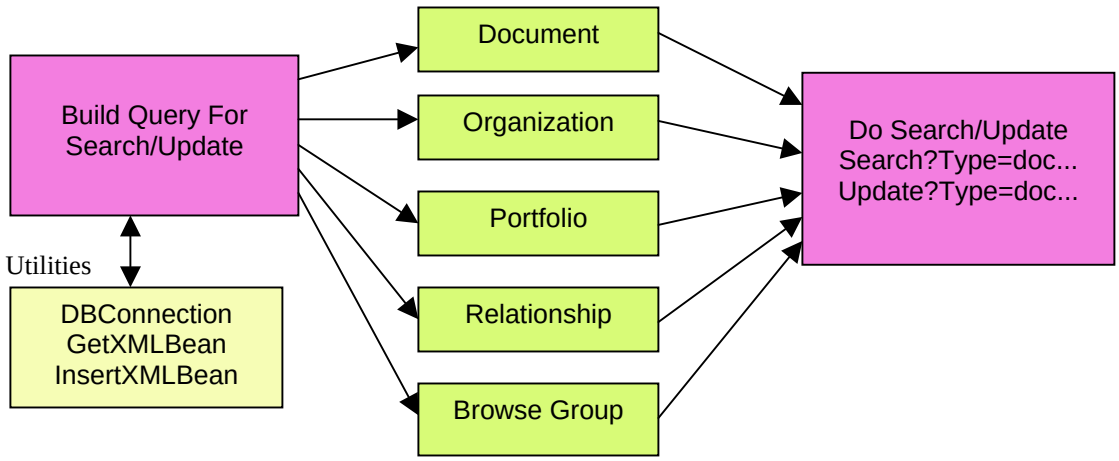

Figure 2. Search and Update

Internal links within the database for search and update are changed to relative links instead of absolute ones, so that the contents of the database can be ported directly to different platforms. For example,

- some text, and
- some other text.

5.3 System and Data Access Control

Both system and data access controls are provided by HCSL, and are associated with a user-type. Access controls are role-based, based on the user type. The HCSL has three primary types of users with associated roles and privileges:

- System Administrator (SA),
- Group Administrator (GA), and
- Registered User (RU).

The System Administrator is a person with overall responsibility for configuring and managing the HCSL system. A Group Administrator is a group member authorized by the SA to be the group administrator for managing the members and contents within the group. A Registered User is a user who has registered with the HCSL, and has defined a user name and password for log in purposes. A registered user can request to join one or more Groups or Subgroups in which they are interested. A fourth user-type, called "Guests" or Non-registered users, does not need a username or password to browse or search the repository. However, Guests can not add, update, or delete HCSL information; this can only be done by registered members of groups or subgroups.

HCSL uses Groups and Sub-Groups to establish organizational structures within HCSL for managing and controlling user membership and contents of each Group and its Sub-Group(s). The Group or Subgroup administrator manages the membership and contents of a respective Group or Subgroup. Each member of a Group and Sub-Group may submit contents, and decide who may access, view, add, delete, and update their contents. Typically, most contents will be public and viewable by all. However, there may be circumstances where a Group or Sub-Group member wishes to keep their contents private, or only allow viewing by members of the Group or Sub-Group. Later, the Group or Sub-Group member may make their contents public and allow viewing by all HCSL users.

Access controls are applied during updating or deleting individual record in database tables. Although database servers typically provide such access control, these controls are not usually portable from one database system to another. Thus, data access controls in HCSL are implemented in the web server and are independent of individual database controls.

5.4 Cocoon

Cocoon is a web development framework based on the concept of component pipelines. It keeps concerns separate and allows parallel evolution of all aspects of a web application. It operates similar to a UNIX pipe, but instead of passing bytes between STDIN and STDOUT, Cocoon passes Simple API for XML (SAX) events.

The three types of Cocoon pipeline components are: generators, which take a request and produce SAX events; transformers, which consume SAX events and produce new SAX events; and serializers, which consume SAX events and produce a response. A Cocoon pipeline is composed of one generator, zero or more transformers, and one serializer.

The Cocoon in HCSL is a servlet connector that allows you to call Cocoon from your servlet engine or application server. It is installed beside existing servlets or JSPs. (See Figure 1). HCSL uses FileGenerator to act as a parser, reading a file (or any other URL) and producing SAX events from it. The SAX event is passed to an XSLT Transformer to transform the SAX stream depending on a given XSLT stylesheet. Finally an XMLSerializer produces an XML response.

5. 5 Porting to New Databases

HCSL has two jsp-servlets for porting from one database to another:

- "readTables" retrieves table definitions from an existing database and formats them into "create table" format for creating the same database tables in the new database.
- "readTablesContents" retrieves the contents of an existing database and formats them into individual "insert into" records to reload the contents into the new database in a new platform.

Both jsp-servlets require the system administrator to check on the jsp code to ensure that they are connected to the existing database from where the database is ported. If the database connection is not the right one, it can be changed to the right one by commenting and un-commenting appropriate lines of code in both jsp codes. As the table names are pre-defined in the jsp code, the administrator also needs to check the existence of the tables in database and possibly add new tables to the list.

6 Conclusions and Remarks

The HCSL prototype provides an open source web-based capability to those interested in finding healthcare information. The prototype contains both sample and real information to demonstrate its capabilities and is available for users and organizations to add additional information. The full value of the HCSL will develop as additional information is added. NIST envisions that the HCSL will become a comprehensive resource of relevant health informatics and healthcare information for the wider community of healthcare stakeholders.

In porting existing code to open source databases, we have changed portion of the code so that the connection to database is only coded once and that proprietary Oracle codes are replaced with equivalent Java programs for working with different databases. We have also eliminated redundant codes and streamlined the HCSL system.

Variables inside functions should always be defined, so that they stay local and do not default to those defined in the calling functions to interfere with those variables. Even though Java is a strong type language that will throw an undefined symbol error message, it defaults to the same variable if it happens to be defined in calling function. Overlooking such an index variable in a for-loop has caused infinite looping.

The HCSL can be used as a tool for other uses. It was built as a 3-tier web-based application providing user and content access controls. Implementers can replace the database tables with their own and modify the query process accordingly. NIST will make the prototype HCSL software and data available for download and use by any organization.

7 References

[1] HITSP – Healthcare Information Technology Standards Panel (ANSI); http://www.ansi.org/standards_activities/overview/overview.aspx
[2] AHRQ – Agency for Healthcare Research and Quality (part of HHS); http://www.ahrq.gov/
[3] CHI – Consolidated Health Informatics (eGov Initiative); http://www.whitehouse.gov/omb/egov/c-3-6-chi.html
[4] HIMSS – Healthcare Information and Management Systems Society; http://www.himss.org/ASP/index.asp
[5] ADA – American Dental Association; http://www.ada.org/
[6] HL7 – Health Level Seven (ANSI accredited consortium developing HC standards); http://www.hl7.org/
[7] NCPDP – National Council for Prescription Drug Programs; http://www.ncpdp.org/
[8] IEEE 1073 – Health Informatics; http://www.ieee1073.org/
[9] http://tomcat.apache.org/tomcat-5.5-doc /index.html
[10] K. Loney and G. Koch, "Oracle8i – The Complete Reference", Oracle Press, 2000.
[11] http://borg.postgresql.org/docs/8.0/static/index.html
[12] http://dev.mysql.com/doc/refman/5.0/en/index.html