Laboratory Information System Interoperability (Design Page)

Primary mentor	Jan Flowers / Jim Sibley
Backup mentor	Bill Lober
Assigned to	Rahul Akula

Background

OpenMRS allows you to store and access patient-level medical data in a flexible and configurable way. But a true, large-scale electronic medical record requires multiple software applications, all interoperating together. A large hospital might use OpenMRS for the clinical patient record, and a separate Laboratory Information System to store detailed information about specimens and samples. Or a national-scale medical record might have OpenMRS running as a centralized database-of-record, with multiple mobile applications connecting to it.

What these situations have in common is that when systems interoperate, one typically behaves as a Master Patient Index, allowing others to query it to find patient records.
At the MEDINFO 2010 conference, we demonstrated the potential for interoperability between OpenMRS and OpenELIS (an open-source Laboratory Information System). We wanted to describe and show a scenario where OpenMRS would be providing the master patient index and OpenELIS would be able to access patient demographic information from the OpenMRS database to avoid the necessity for dual data entry of that information into both systems. As the framework for this interoperability, we decided to implement the standards developed by the Integrating the Healthcare Enterprise (IHE) initiative. We successfully created a working demonstration of the concept, but more work is needed to make this interoperability link suitable for production use.

Purpose

The purpose of this project is to complete the work started on developing the Patient Demographics Query (PDQ) Supplier module for OpenMRS. The PDQ Supplier module should be able to receive queries from external applications, parse those queries and perform searches against the patient tables in the database and finally, return demographic content for patients matching the search query criteria.

Required Skills

- Strong Java development skills
- Knowledge of HL/7 a plus

Objectives

1. Fully parse and handle basic query parameters (first and last name).
2. Use dynamic/configurable values instead of hard-coded ones for the required message segments.

Extra Credit/Wish List

1. Handle additional types of queries other than just first and last name (address, gender, wildcards).
2. Handle PDQ continuation queries ("give me results 11-20", etc.).
3. Add ability to register a patient.
4. Add the ability to restrict the ability to query to specific institutions only (via configuration options page).

Project Plan

Milestones

- Mid-Term milestones
    1. Remove all hard-coded pieces and use standard HL/7 parsing (HAPI).
    2. Get the module to a point where it can successfully handle a simple query using first and last name and return the appropriate results over HTTP(S).

- Final milestones
    1. Add extra credit/wish list features (see above).
    2. Investigate the possibility of using sockethl7listener module (or similar), or other transport options instead of (or in addition to) HTTP(S).

Timeline

April 25 – May 23

- Set up development environment
- Go through the existing implementation of PIX/PDQ

May 24 – June 13

- Fully parse the message using HAPI

June 13 -- July 1

- Handle basic query parameters (first and last name).

July 1 – July 18

- Get the module to a point where it can successfully handle a simple query using first and last name and return the appropriate results over HTTP (S).

July 18 – July 25

- Implement extra credit/wish list features

July 25 – August 1

- Documentation of the project

August 1 – August 22

- Make further improvements if required

Resources

- PDQ Specifications (section 3.21)
- OpenELIS Information, Demos, Source Code, and VM
- Project Progress Wiki