Development of a Framework for Laboratory Information Tracking in Support of Research and Clinical Operations

B. Craig, B. Tibbils, K. Topper, S. Griffith, L. Watkins, K. Doheny
Center for Inherited Disease Research (CIDR), Johns Hopkins Genomics (JHG), IGM, JHU-SOM, Baltimore, MD

Introduction

The Center for Inherited Disease Research (CIDR) part of Johns Hopkins Genomic (JHG) provides high quality next generation sequencing (NGS), genotyping and statistical genetics consultation to investigators working to discover genes that contribute to disease. CIDR’s current genotyping Laboratory Information Management System (LIMS) was designed to handle the tracking of detailed information for large numbers of samples going through specific genotyping assays. This LIMS has over time become difficult to maintain and extend as newer resources and NGS services have emerged. A third party LIMS was selected for NGS applications that addressed the widely different set of tasks and information to be stored for sequencing services. Annual licensing costs and requirements to re-write custom code and upgrade to newer releases to maintain support add to the difficulty of sustaining two LIMS deployments. These deficiencies combined with the expansion of research and clinical genomic services at JHG (jgenomics.jhmi.edu) have highlighted the need to create a new LIMS to support the breadth of current and future protocols. These rapidly changing laboratory protocols require reliable tracking of large amounts of workflow information including a record of sample movements, identification of robot resources, reagent association and user comment tracking. Here we describe the development of a MySQL database backed Java framework consisting of tools to reduce disruption to lab processing with module hot swapping, limit database schema complexity and increase reusable graphical user interface (GUI) components to flexibly build and extend workflows for any protocol driven technology.

Motivation for LIMS Infrastructure replacement

- Genotyping Service LIMS
- Developed with 1 Illumina Infinium workflow in mind
- Complex code management
- Uses outdated software and hardware resources
- Sequencing Service LIMS
- Pay for annual user license
- Version updates require extensive time commitments
- Difficult to maintain two LIMS deployment strategies

Current LIMS deployment strategies

<table>
<thead>
<tr>
<th>Topic</th>
<th>Current In-house Genotyping LIMS</th>
<th>Current 3rd Party Sequencing LIMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age (years)</td>
<td>12+</td>
<td>7</td>
</tr>
<tr>
<td>Database design</td>
<td>Highly normalized MySQL schema</td>
<td>Proprietary MySQL schema</td>
</tr>
<tr>
<td>Application Logic</td>
<td>HTTP requests -&gt; Java code validation -&gt; JDBLC -&gt; HTTP responses</td>
<td>Proprietary business logic software</td>
</tr>
<tr>
<td>Client Interface</td>
<td>JavaScript web interface</td>
<td>Proprietary JavaScript web interface</td>
</tr>
<tr>
<td>Downtime</td>
<td>Any modification requires code redeployment</td>
<td>Business logic software updates and Modifications to UI fields</td>
</tr>
</tbody>
</table>

New LIMS Framework Decisions

- **Database Design**
  - MySQL 5.7 (Figure 1)
  - Use of LONGTEXT datatypes
  - Store submission data in XML format
  - Simplified table relationships
- **Application Logic and Client Interface combined**
  - Java 8 with JavaFX GUI (Figure 2)
  - Customized and reusable components
  - Consistent phase navigation look and feel
  - "You are here" visualization
  - Real time and submit only validation notifications
- **Java Database Connectivity (JDBC)**
- **Serialization of Java objects into XML**
- **Automating code deployment**
  - Gradle scripts
  - Compiles and validates source code
  - Creates Java Archive (JAR) files
  - Creates deployment directories
- **Downtime**
  - "Hot swapping" JAR files
  - Reflection and database validation
  - Eliminates downtime

General LIMS Framework Goals

<table>
<thead>
<tr>
<th>Project Goals</th>
<th>Maximize Flexibility</th>
<th>Simplify Database Interactions</th>
<th>Update Deployment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building reusable tools</td>
<td>Storing XML metadata</td>
<td>Automating deployment</td>
<td>Enabling hot swapping</td>
</tr>
<tr>
<td>Designing by forward thinking</td>
<td>Simplifying table relationships</td>
<td>Removing multiple tiers</td>
<td></td>
</tr>
<tr>
<td>Lowering memory footprints</td>
<td>Accelerating data reporting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Specific LIMS Framework Goals

<table>
<thead>
<tr>
<th>Increasing</th>
<th>Decreasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turnaround time for new LIMS jobs</td>
<td>Complexity of code maintenance</td>
</tr>
<tr>
<td>Robustness of audit tracking</td>
<td>Deployment errors and bugs</td>
</tr>
<tr>
<td>Scope of the LIMS to include clinical and R&amp;D operations</td>
<td>User frustrations</td>
</tr>
</tbody>
</table>

Discussion

The missions of CIDR and JHG are to map a genetic path to better health by integrating expertise, enabling data driven discovery and informing patient care. To accomplish this mission, services offered are constantly evolving requiring accelerated development of information tracking mechanisms. Originally designed for very specific workflows, the current LIMS infrastructure can no longer evolve at the pace needed to accomplish this goal. Building a new LIMS from the ground up gives the opportunity to evaluate new database types, deficiencies in the current LIMS and software development practices. Leveraging extensive database experience gained in the decision to use MySQL to make use of XML data types to store submission data in a graph like interconnected way (Figure 1). Conversion of user supplied data from Java objects to XML for database insertion, GUI repopulation and reporting is supported using the XStream Java library (x-stream.github.io). Each submission is treated as a new timestamped record in the database to extend audit tracking mechanisms. User feedback strongly shows that wet bench protocols can be optimized if GUI validation and code were more flexible. To address this deficiency dynamic phase GUI elements with methods for global and lab customized validations have been created. Validation failures are presented to users by contextual pop-ups (Figure 2). Review of current LIMS implementations reveal now obsolete Java library versions and code duplication to support lab and vendor protocol changes over time. These shortcomings are addressed by using a current JDK version, creating and updating internal cross project Java utilities and when possible using Maven (search.maven.org) to manage external library dependencies. Development work now focuses on creation of jobs and phases to mirror current genotyping production protocols. Through the use of updated technologies, forward thinking and inclusion of highly technical individuals this project will be integral to driving research and clinical operations into the next decade.