Handling Observation Exceptions (Design Page)

With the evolution of the FHIR standard, this design should probably be adjusted to align more closely with FHIR – e.g., using the cancelled Observation.status along with Observation.dataAbsentReason from FHIR's Observation model instead of HL7 v2.x's "X" status approach.

Primary mentor	Burke Mamlin
Backup mentor	TBD
Assigned to	TBD

Background

Within OpenMRS the majority of data collected about patients are stored as observations. Examples of observations are: the patient's weight, the patient's pulse, the patient's answer to the question "How many children live in your home?", or a lab result like serum potassium level. In most cases, we collect these observations as data and stored them in the system without any problems. In some cases, observations cannot be completed, but there is value in recording the exception rather than simply omitting the result. While exceptions for laboratory tests are most common, exceptions are not limited to laboratory tests. Some examples of common exceptions would be:

- Inadequate specimen (e.g., blood received, but not enough blood to run the test)
- Wrong tube (e.g., some test tubes contain preservatives to preserve the blood for testing; if the wrong type of tube is used, the test cannot be run)
- Not applicable (e.g., a form requires name of spouse, but the patient is single)
- Patient unable to answer (e.g., some required questions on a form must be answered by the patient, but the patient is unconscious)
- Patient refused
- Test cannot be performed (e.g., a patient is sent for an MRI, but found to have claustrophobia or metal in their body)
- ~~Pending in Lab (e.g., the lab has received the specimen, but does not have a result to report while testing is underway)~~ "Pending" is a status, not an exception.

Currently, OpenMRS has the ability to store observations but lacks a standardized way to represent observations with exceptions. Implementations are left to come up with other ways to handle exceptions – e.g., record the exception as a separate observation or omit the exception altogether (possibly losing valuable information).

Purpose

The goal of this project is to add a standard mechanism for handling observation exceptions to the OpenMRS API so that implementations can – in a standard & predictable way – handle and record observations even when they are incomplete, refused, or have other exceptions.

Domain Expert(s) / User(s)

TBD

Required Skills

- Strong Java skills
- Familiarity with the OpenMRS API
- An understanding of how Concept & Observations are modeled within OpenMRS

Objectives

- Add an attribute to indicate observations with exceptions and specify the exception (e.g., pointing to a concept like "PATIENT REFUSED" in the concept dictionary)
- Adjust existing API methods to filter out (omit) observations with exceptions
- Add new API methods that include observations with exceptions
- Adjust the observation management screen(s) to allow exceptions to be recorded
- Update the HL7 ORUR01 handler class to know about exceptions and add the right kind of observations
- Provide some documentation for module developers to understand how they can adjust their code to include/exclude observations with exceptions

Design Ideas

Data Model Changes

Explicitly mark observations that have exceptions through the use of a status field. For example, add one or two attributes to the obs table:

- (+) obs.status *(varchar 2)*
    - The default status for observations is final ("F")
    - Exceptions would have a status of "X"
    - This allows for future expansion to other status codes – e.g., "P" for pending results (in lab)

        - Aligns closely with FHIR's Observation.status field
- (+) obs.exception_code *(Foreign key to concept.concept_id)*
    - This would point to a concept like "Inadequate specimen" or "Specimen hemolyzed"
    - We might want a new "Observation Exception" concept class to categorize these concepts
    - This would align with FHIR's Observation.dataAbsentReason field
- (+) obs.exception_value *(varchar 255)*
    - An optional text field in which a exceptional/invalid value could be placed
    - Based on this discussion in 2013

Backwards Compatibility in API

For backwards compatibility (since existing code assumes all observations have values), have the existing API work as it does (omitting exceptions) and add new methods to get results including exceptions – e.g., getObservations(..., boolean includeExceptions).

Upgrade Strategy

This change will need an upgrade strategy, since the `obs` table can be massive and changes to its structure can be prohibitively slow. Simply changing the structure of the `obs` table in a liquibase changeset (as we normally do for small changes) could prevent implementations from being able to upgrade.

Ideas for upgrade strategy:

- Create a SQL script that makes the needed model changes in the most efficient way possible and is applied before upgrading OpenMRS
- Create a helper tool/module that an implementation would run before upgrading OpenMRS
    - Makes a copy of the `obs` table with new columns,
    - Adds temporary stored procedures to copy any new inserts/updates
    - Copies records from old to new obs tables in the background in a resource-throttled manner (e.g., so it can run over days or a week without disturbing usage of the system)
    - Once all rows have been copied, the obs tables are swapped
    - The tool should work with MySQL and MariaDB, but ideally would work for most common databases (e.g., Postgres)
- Or demonstrate a simpler upgrade step (e.g., liquibase change) can be done by large implementations by testing it on a massive obs table and proving that it can complete in a timely manner.

Extra Credit

- Add support for observations with exceptions to other aspects of the OpenMRS web application – e.g., patient dashboard, encounter management, etc.
- Adjust the default HL7 handler to recognize observations with exceptions and store them appropriately.

Resources

- The OpenMRS Data Model and data model browser.
- Original OpenMRS JIRA Issue for this topic