2020 RwandaEMR Upgrade Toolkit

Table of Contents

Technical Upgrade Guide

Gap Analysis: What will be upgraded?

This upgrade aims to provide the same functionality currently available in RwandaEMR OpenMRS version 1.9 following the upgrade to OpenMRS version 2.3. The Components and Upgrade Analysis identified a) current modules to be retained, b) current modules to be modified/updated, and c) current modules to be removed. This analysis also includes target

modules that will be included in the upgrade. For a full listing of modules to be upgraded, please see 2020 RwandaEMR Gap Analysis.

What development work needs to be done?

Development work to be done will be initially determined through the upgrade gap analysis. The technical team set up a Jira project where details about the work to be done are documented in Jira tickets. The upgrade status can be monitored by module or by component using the 2.3 Upgrade Dashboard in Jira. (See more information about Jira & Issue Trackers below in the Issue Tracker Dashboard section, including a screenshot from the Rwanda EMR upgrade Jira Dashboard.)

Who will perform the upgrade?

The technical upgrade will be completed by a technical team consisting product owners, project managers, developers, quality assurance managers, testers, implementers, and stakeholders/sponsors. A description of these roles with their responsibilities is available here. Representatives of RBC, PIH, ICAP, Jembi, and the OpenMRS Community have each been assigned one of these roles, with modifications made to reflect the project needs.

How will the upgrade be done?

In order to upgrade to 2.3, it will be necessary to progressively identify, document, and ticket all of the migration issues that exist. The approach should be one of analysis, with an aim to identify an issue, and then clear that issue however necessary (not correctly) in order to rapidly expose the next issue, and so on until no more issues are encountered.

The process could be the following:

1. Review existing documentation and code
    a. Review Prepare for Upgrading from a Pre 1.10 to 1.10 or Later Version
    b. Review everything in rwandaemr/migration/src/main/resources/todo.txt and create migrations as needed for everything described in here as appropriate. As migrations are created, remove from the todo.txt file. Ultimately, we want to remove the todo.txt file completely and be left with just a number of files in the migrations folder that reflect the various steps. These can just have high-level TODO comments within them describing what needs to happen for now, and for each a ticket should be created under epic RWA-279
    c. Review work the Kaweesi did for RBC a few years ago here (this should not necessarily be taken as correct, just used to inform).
1. Work through the upgrade manually
    a. Create a new 1.x branch of the rwandaemr module, all the upgrade work to 2.3 will happen in the master branch
    b. Change the openmrs version to the latest OpenMRS release version (or Snapshot, tbd)
    c. Change the version of the rwandaemr itself to 2.0.0.SNAPSHOT

d. Build and install this version of rwandaemr to maven (work through any compilation errors and document and resolve them as appropriate)
e. Create a new SDK instance for this version, using Java 8 and this 2.3-SNAPSHOT distribution version
f. Try starting up this SDK instance, and as issues arise:
    i. Document the issue symptoms
    ii. Find the cause, and determine the correct technical approach for solving it
    iii. Create a ticket describing the above with an estimated story points
    iv. Work around the issue with brute force (eg. deleting or changing data in the DB, etc) in order to be able to get past it and identify the next possible issue. Try not to do too much that might affect our ability to find other migration issues (eg. don't delete * from orders)

The result of this process should be fully documented, ticketed, and estimated set of tickets required to prepare for a successful core 2.x migration.

What is the timeframe for the technical upgrade and implementation?

May 1 - September 21, 2020: Technical upgrade
September 21-25, 2020: Regression testing, UAT during bootcamp
September 28 - October 9, 2020: Complete upgrade fixes
October 12-16, 2020: Trial upgrade at initial pilot site
From October 26, 2020: Upgrade implementation to remaining sites

Note that roll out of the upgrade is contingent on sign off by RBC.

What is the quality assurance plan?

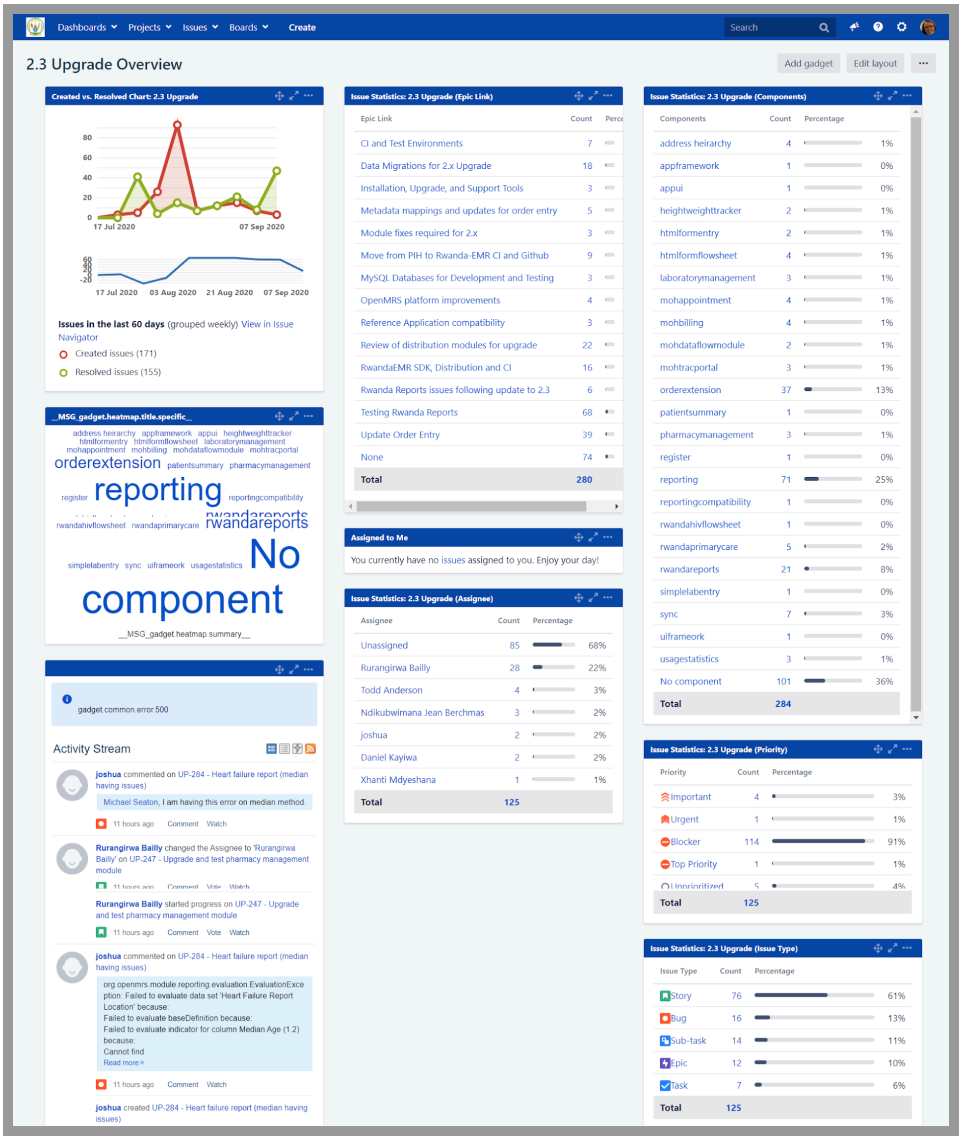
Quality assurance will take place throughout the upgrade process, consisting of automated tests during continuous integration, regression testing, UAT testing, and pilot testing. Sign off on regression testing, UAT testing, and the pilot will indicate that the upgrade meets technical, clinical, and RBC standards and that the upgrade can be widely implemented.

Quality assurance documents and tools

Quality Assurance Strategy
Regression Testing Checklist
OpenMRS Rwanda 2.3 Test Case Template
OpenMRS Rwanda 2.3 Regression Test Case Template
User Manual to guide UAT
Upgrade Data Quality Audit/Validation

Issue Tracker Dashboard: Jira Example

The Rwanda Upgrade Jira Dashboard has closed permissions, but here is a screenshot of what the dashboard looked like. The goal was to show which Epics (features or modules) needed more attention vs were in progress vs were done, who was doing what work actively, and where Bugs were cropping up.



We're not using Jira - should we switch?

Jira is such a big, complex tool that we do not recommend switching to Jira only for the purpose of an upgrade, nor just for the purpose of Dashboarding. Ideally you should use whatever issue tracking tool your development team has already adopted into their workflows.

We're using Jira already - how would we make a dashboard?

Here is the dashboard how-to-guide from Atlassian:
https://support.atlassian.com/jira-core-cloud/docs/what-is-a-jira-dashboard/
If you decide to set up a Jira Dashboard for your team, we recommend the following gadgets, since they help answer the following questions:

- *Issue Statistics* (by Epics): Which feature/theme areas are we addressing? Where are the most issues? (e.g. “Patient Registration”. Your team may prefer to use Labels or Components to group themes like this.)
- *Issue Statistics* (by Assignee): Who is actively working on the issues identified?
- Issue Statistics (by Issue Type): How many Bugs do we have?
- Issue Statistics (by Priority): What Blockers do we have? What Urgent issues are left?
- *Created vs Resolved Chart* (including the *Unresolved* trend line): Are issues are piling up or being solved?

Where can we go for more help about Dashboarding during our Upgrade?
You are welcome to reach out to OpenMRS Director of Product Grace Potma for support with dashboard creation, whether you are using Jira or another tool.

Upgrade Implementation Guide

What is the plan for implementing the upgrade?

The Implementation Plan Proposal outlines the approach for implementing the upgrade, the main activities to be completed during centrally and on-site, and who will be involved in implementing the upgrade.

Where will the upgrade be piloted? Where will the upgrade be implemented?

The upgrade will be piloted at up to 3 sites in order to identify any code and/or database conflicts and where any additional work may be needed. The pilot also serves as an opportunity for the upgrade team to follow SOPs and make any adjustments prior to implementation.

The upgrade implementation schedule for all sites to be upgraded can be found in EMR Rwanda Upgrade 2020: Facility Monitoring.

Upgrade Procedures

The upgrade itself comprises of the following steps:

1. Backup database
2. Database Migration script run
3. Upgrade to Ubuntu 18
    a. Java 8
    b. Tomcat 8
    c. MySQL 5.7

4. Deploy 2.3.x OpenMRS upgrade

These steps are provided in detail in the RBC Database Upgrade Steps (Kibagabaga)

Upgrade Scripts and Tools

The OpenMRS 2.3 Upgrade Installation Guide v0.1 provides details for completing each of the above steps in order to upgrade from OpenMRS version 1.9 to 2.3.

Upgrade Data Quality Audit/Validation

Approach

Possible options include:

- Add the dataintegrity module to the distribution, and attempt to use that: https://wiki.openmrs.org/display/docs/Data+Integrity+Module
- Integrate with PowerBI to produce useful dashboards of data integrity checks
- Add a custom report(s) or a custom page to the rwandareports module that can be used for this purpose.

The purpose of validating data quality during an upgrade is to ensure that data migrated from the old database to the upgraded database is consistent. Any discrepancies identified during this process should be investigated and confirmed that the discrepancy is due to the upgrade. In this case, the issue should be reported and fixed by the designated developer. Routine data quality assurance processes and procedures should be followed post-upgrade.

This approach consists of both manual and automated components and is described below. This Data Quality Audit Process outlines the procedure for using these two data quality validation tools during the upgrade process.

Manual Data Quality Audit

A manual data quality audit compares data on a select report and patient data recorded in the Data Quality Audit workbook before the upgrade to the same report and patient data run after the upgrade. In addition to a monthly facility report, some possible facility and patient level data to include in the data quality audit include:

Facility

- Total number of records
- Total number of patients

Patient

- Unique patient ID (as recorded in the EMR system)
- Sex
- Date of Birth, DOB (dd/mm/yyyy)

- Total number of encounters
- Date of First encounter
- Date of Last encounter
- Date of first ARV pickup from the current facility
- First documented regimen
- Last documented regimen
- Date of last lab encounter
- Date of first lab encounter
- Date of last documented viral load (if available)
- Last documented viral load result

Data Integrity Module

The Data Integrity Module provides a way to create, manage and review the quality of data audits. Users can view or download the results of an integrity check, ignore certain known results that should not be included in the total count, see a chart representing the history of an integrity check's findings and link directly to encounters, patients, observations and more from the results. It is meant to provide a common place for finding and dealing with multiple common data quality issues.

Some potential integrity checks are:

- Patients with no or more than one preferred identifier
- Patients with no name
- Patient Identifiers with an incorrect check-digit
- Unused Concepts in the Concept Dictionary
- Unused or duplicate Field Objects for Forms
- Unused Locations
- Encounters having no Observations
- Pregnant male patients
- Patients who have a wrong encounter type (e.g.: A pediatric patient with an adult encounter type or form)

Support

Support for facilities will be provided by RBC in partnership with ICAP.

Issues will be escalated according to the following hierarchy:

1. Clinical staff ask ITs for support
2. ITs create Jira tickets

Clinical staff will have access to a Whatsapp group. Most issues brought up to the Whatsapp group should then be logged in Jira by an implementer/developer.

Resources

PIH RwandaEMR Distro Guidance
OpenMRS Preparing for an Upgrade Wiki page
Upgrading OpenMRS Wiki page
UgandaEMR Upgrade
Nigeria Migration Guidance
Nigeria Data Migration Tool
eSaude Data Migration System
eSaude Data Migration Guide
    Dummy Data Mapping
eSaude OpenMRS Data Migration Tool Module
Spreadsheet Module 2

Information about module for previous RwandaEMR Upgrade
2016 RwandaEMR Upgrade module
Upgrading OpenMRS