Community Development Swim Lane

Background

As a community, OpenMRS understands there is great value in investing in ongoing (omni-present) development effort dedicated to cleaning up general bugs and feature requests that are highly voted (i. e., important to implementations). We also know new developers have a better experience when there are more experienced developers around to help answer their questions. Therefore, we have created an explicit "Community Development" swim lane that is always running (and sometimes gets its own sprint) within which developers in the community can work together to learn how to work with OpenMRS and, together, try to knock out some of the bugs & feature requests that matter most to our implementations. Recognizing the importance of this swim lane, we (OpenMRS) ensure that an experienced developer is involved in the community development swim lane. The lead developer in the community development swim lane should try to find about 20% -25% of their time to work on tickets in the ongoing OpenMRS sprint at the given time.

Swim lane? Why a swim lane?

In swimming pools (whether for practice or during competitions), swimmers are placed into lanes. This serves to keep people from bumping into each other and, for developers, the idea of a "swim lane" keeps them focused on their task/goal. Swim lanes are a popular metaphor used in managing business processes.

Description

- Every week there will be at least one developer (hopefully many) working in the bug fixing swim lane. We call these folks '**Community Developers**'. This can (and purposely) happens while other devs are working on the current Development Sprint.
- We assign a full-time OpenMRS developer to lead the community development process. See calendar below to identify who this person is on any given week.
- The goal of the community development swim lane is to tackle both high-priority bugs and long-standing issues. This keeps the queue of them from building up and requiring fewer all-team bug fixing sprints.
- The community developers should send a *report* out to the community at the end of each week with a brief overview of what all tickets were completed during that week.

> When possible, the community developers should try to avoid the low complexity & intro tickets, if they are able to tackle others, leaving those for new devs and volunteers wanting to get their feet wet.

- Community developers should look at bugs across all openmrs-supported projects and modules. That list currently is:

```
project in ("OpenMRS Trunk", "OpenMRS Standalone", "Release Testing Helper", "HTML Form Entry Module",
 "Reporting Module", "HTML Widgets", "Reporting Compatability Module", "Serialization
XStream", "XForms Module",
 "Patient Flags Module", "Form Entry Module", "Metadata Sharing Module", "Module Maven Archetype")
```

- The list of all OpenMRS pull requests.
- You should also regularly look at error reports at . Look for validity/reproducibility, and, if appropriate, convert them into bug issues in the appropriate JIRA project(s):

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due

No issues found

Join the Community Development Swim Lane

We assign a full-time OpenMRS developer to lead the community development process. See the calendar below to identify who this person is on any given week. Look for community developers in IRC or send a note to our Developers mailing list.

Dashboard

The queue of tickets that the community developers should choose from is here: https://tickets.openmrs.org/secure/Dashboard.jspa?selectPageId=11753

Choosing tickets

The most important box is the yellow "Open Blocker Tickets". This list should always be as small as possible. Any unassigned ticket should be top priority; claim them; Some in the list might just need a second person to review; review those. Some might have an assignee that has languished; ask them if you can claim the ticket from them.

The bottom left box is general open bugs. These are ordered by votes. The popular tickets are on top. Choose any off the stack.

The middle box are tickets that have been submitted but need to be confirmed, tested, reproduced, or just moved into the "Ready For Work" state in the right project. These should be relatively quick tasks. Anything needed to get them out of the "Needs Assessment" state.

The far right column contains unclosed tickets with patches attached to it that need to be reviewed and tickets that have been committed and just need to be reviewed a final time before being closed (or asked for Rework if needbe)

Community Development Lead Calendar

Team Calendars