Decisions Made at the OpenLMIS Community Meeting 9/28 – 10/3

The following tasks will take place in order to get to OpenLMIS “2.0”

* Cleanup:  JSI to remove “proof of concept” tables
* Hide features (UI)
	+ Doc/describe how
* Have a default/demo config (following [github.com/clintonhealthaccess/lmis-moz](http://github.com/clintonhealthaccess/lmis-moz) pattern)
	+ Evaluate TW customization module
	+ For country-specific configurations
	+ Doc and recommend as good practices
* Doc and describe features
	+ Identify which are Global versus country-specific
* Merge in "accessibility"
* Published FAQ on current state?
* Endgame:  final test/stability run; merge to master
* Finalized branching strategy (e.g. "gitflow")

Need stable (will build) releases, will be tagged on the master branch. *How is TBD (and documented on Technical Committee Open Questions list)*

**How do we get to a “stable” place by the beginning of 2016**

1. We Commit to making 2.0 the upstream branch by end of 2015
2. Need CI and automated reviews  (possibly need a QA environment)
3. Updated workflows around pull requests/reviews
4. Test data set/configuration
5. Need 2.0 Demo instance (there are ones for TZ and MOZ projects)

## **Tasks: (Owner is in brackets)**

1. Nominate and establish technical WG
2. Accessibility from 1.0 is merged into 2.0 [VR]
3. Gitflow setup (rename '2.0' to 'master', etc.) [VR]
4. CI Server:  [VR]
	1. Need to pick a CI:  "travis", based on github permissions.  Most orgs currently use Jenkins
5. Sonar server [any org]
6. Publish rules in dev documentation, a "contribution guide"
7. Dedicated time for regular reviews, merge into master
	1. Public expectations between organizations, who has privilege to commit to master trunk?
		1. Write down the vetting requirements/criteria for this committer
8. Establish tech review board, and its arbitration process
	1. Publish Pull request process and expectations
9. Tool selection (issues, wiki, etc.)
10. Create test dB, demo environment

How does the community balance conflict of interest between project funders and in-country stakeholders (i.e. funders wanting re-usable features while Ministries want features that specifically match to their needs)?

*By allowing implementers to choose the implementation model (see implementation models document) that best reflects their time, money, donor & country constraints.*

Branch structure is as follows:

Shared Master (release). The technical committee will review pull requests from Dev to Master.

Shared Dev – There are no pull requests for Dev. If you push to the Dev branch you are stating that you think that code is “Core” OpenLMIS code and you think it’s ready and will build. If you push something that doesn’t build, you’ll be shamed and your code may be reverted. Dev will only be available to be written to for active partners / collaborators.

Master and Dev must always build; by this we say it is “Stable”. However, code does not enter Master only as a “module”, so Master and dev may have incomplete features.

Individual branches for organizations / implementations off of dev.

Should OpenLMIS try to attain a presence across the full stack (i.e. procurement to facility) or focus on being a well-built part of an application ecosystem?

*Broad community support that the domain of OpenLMIS is more limited - should not attempt to become the full stack! Should instead focus on those attributes and features which fall within the OpenLMIS domain. At the “edges”of thi domain, OpenLMIS should either:*

1. *Support interfaces for large players (i.e. DHIS2, Epicor, etc.) in the space or*
2. *Provide “light” implementations of these components*

*Resounding answer is that OpenLMIS is part of an ecosystem, NOT the ecosystem!*

What are the key attributes of the OpenLMIS software?

*To be completed by Mission, Visions, and Values OneText process.*

What activities are the responsibility of the proposed Marketing & Advocacy Committee?

*Communications*

* *Internal*
	+ *Knowledge sharing- documentation, information sharing, community support*
* *External*
	+ *General brand, messaging, awareness, shared assets (collateral, web, PR, etc.)*

What are areas for collaboration in the community?

* *Communication*
* *Product roadmap*
* *Community roadmap*
* *Sustainable thinking*
* *Implementation support (as outlined in implementation guide)*
* *On-going support (as outlined in implementation guide),*
* *Others?*

What are the capabilities of the OpenLMIS Community?

* *Fundraising / Revenue Generation*
* *Logistics Expertise*
* *Experience Design*
* *Product Management*
* *Evangelism*
* *Knowledge of other systems; Systems integration*
* *Supply chain domain expertise*
* *Academic partnerships – bridging academia + real world*
* *Development talent*
* *Health informatics expertise (i.e. what systems can we use that are feasible with local workforce?)*
* *Advocacy with Ministry of Healths - explaining value and making appropriate recommendations based on resources*
* *Others?*

How do we build the first iteration of the roadmap?

Run activity to isolate the features the OpenLMIS Community wants to be added to OpenLMIS, including prioritization. *The product manager will then build an initial roadmap down to feature level and present it to the product committee for feedback.*

*Update -- Will be announced and addressed in the Product Committee meeting in November.*

What is on the module level OpenLMIS roadmap?

***Prioritized List:***

* *Re-architecture / Modular Design*
* *Stock Management*
* *Act as a Data Source (Three-way tie)*
* *Mobile API / Reference UI (Three-way tie)*
* *Support an Upgrade Path (Three-way tie)*
* *Best Practice Processes*

*See OpenLMIS 2016 roadmap for more granularity.*

*JSI is going to document lessons learned around their OpenLMIS & DHIS2 integration in TZ. Ashraf is the touch point on this.*

Tactically, what marketing collateral can the community provide to support stakeholder's business objectives?

* *Community-centered messaging rather than organization specific OpenLMIS messaging*
* *A country / Ministry of Health roadshow showing off OpenLMIS features*
* *At USAID it’s more word of mouth than printed material (so none?)*
* *From the ThoughtWorks perspective, the questions tend to revolve around technology.*
	+ *Therefore, blurbs about specific use cases are helpful - more so than say, a long feature guide. Implementers might only need a specific piece/part of the document/messaging specific to a particular country/audience.*
	+ *Demo videos are helpful. Make it feel like it’s in use and a real thing “tangible”.*
	+ *Incorporate the “open source” message well as it’s a strong motivation.*
	+ *ThoughtWorks is fine with using non-ThoughtWorks branded technology b/c it’s in the best interest of the customer and TW has expertise around it.*

Should integrations be branded as OpenLMIS?

*No, not unless the community agrees with the outside partner (i.e. OpenLMIS & Dimagi)*

How do we ensure code delivered back to core is of high quality?

*Multi-tiered answer!*

* *Code review as outlined by technical group*
* *Use implementation guide to remove pressures from delivering fast + re-usable*

Who reviews the pull request? This comes with overhead

*Someone on tech committee or another resource*

What happens if technical WG decides a pull request should not be merged back?

* *If code/feature is not core/global, then the implementing team needs to create an extension point or some other mitigation*
* *Could have a "code review window", scheduled times when others are available as reviews*
* *In OpenMRS, three people are dedicated as full time reviewers, etc. for a period*
* *VR could have more active pull request review roll, as VR has some level of community funding*
* *Could merge, say weekly. Accept or reject (with comments back to owners),*
* *Also review ad hoc pull requests*
* *Decide release tags.  Master is not strived to be stable 100% of the time*
* *Concern of overhead, esp. if rejections are involved.  This is part of the open source tax.*
* *A technical working group ("security council") resolves disputes.  Needs an escalation process*
* *Good practice is to avoid long running feature branches*

What are the standards for merging into master?

* *Feature alignment - is the feature applicable to global/core?*
* *Code quality check(coverage, tests pass, code style standards, etc.). This can be automated.*
* *It is not checking low-level stuff, e.g. code formatting.*
* *Need CI, Sonar server, etc. to help with automation*

Who watches out for the integrity of OpenLMIS?

*A combination of the roles of the Product Manager and Technical Architect. The Product Manager is responsible for prioritizing work on OpenLMIS features that fall within the core domain of the product, while the technical architect is responsible for the quality of design.*

How many tiers of support are there?

* *T1- Basic - User can't log in, can't go to next screen, etc.*
* *T2- T1 support followed the script, but didn’t get an answer for the user*
* *T3- Possibly a bug – may need to go back to developers; could include new feature requests if there is a maintenance fund*
* *T4- Dedicated admin?*

Who is currently contributing code (as of 10/1/2015)?

* *ThoughtWorks for the Mozambique implementaion*
* *VillageReach, CHAI, JSI and PATH to VIMS / eLMIS in Tanzania*

What is the scope of the technical WG?

* *Arbiter for disagreements on pull request reviews, etc.*
	+ *How are decisions made?  What if there's a tie?*
* *Forum for technical discussions*
* *Final architecture decisions*
	+ *Decisions made via democratic vote*
* *Tech review of new projects; act as a "check" for the product management function*
* *Regular architectural review for roadmap and tech debt*
* *Decide on tech/coding standards, which determines what code and features can make it into the master branch*
* *Owners on the relevant OpenLMIs github repositories*
* *Tool selection*
* *Responsive to community input*
* *Reviews and approves membership (in collaboration with governance group)*

Where should OpenLMIS be one year from now?

* *Ashraf:  a site that shows OpenLMIS as a production release (similar to DHIS), details the features that OpenLMIS offers, brochure and other doc is available.  Materials like install/config guides are available, and a demo site for folks to investigate the application.*
* *Developers have easy access to documented ("published") APIs, dev environment setup, coding standards and Dev Guide and infrastructure (how to contribute, good practices, etc.).   Dev onboarding process is defined and easy to find.*
* *Doc specific to implementers is available.*
* *Tech Committee exists and functions.*
* *Jeff:  Collaboration environment for implementers, a place for discussion and advice*
* *Published road map.*
* *Architecture:  OpenLMIS has an extensible architecture.*
* *Shared infrastructure for issue tracking, etc.*
* *Ashraf described a sort of internship program to build capacity and skill for in-country staff.  Donors could fund this.*
* *Support forums for devs, implementers.  (DL/listervs, etc.)*

Who owns the responsibility of the above?

## **Coding standards:  once established, what do we do with the existing code?**

* *First, don't  make code worse (new code must adhere to standards)*
* *Legacy code is refactored opportunistically, typically if a developer is updating exist code/files.*

*Committee meetings should be open for anyone to join and participate in discussions. The time and dates of these meetings should be posted on the wiki ahead of time. Visitors are encouraged to participate in discussions, however, it is the responsibility of the committee members to make decisions, by consensus if possible, and otherwise by majority.*

*The Technical Committee is responsible for the OpenLMIS repository and making all decisions regarding it’s upkeep and maintenance.*

How will committees share their plans with the larger community?

*Open meetings, notes posted in meeting minutes.*

We should create a toolkit for implementers ([similar to OpenHRIS](http://www.ihris.org/toolkit-new/))!

*Yes! Who can take on this responsibility?*

How do we measure success on the “re-architecture?

*Product and technical committees to work together to define:*

* *From the product side, what pain points do implementers and end-users see?*
* *From the technical side, what issues are implementers & developers seeing?*

*Representatives from both groups to meet and describe what we want the outputs of the re-architecture to be and how we get from here to there.*

Should additional implementations start?

*Yes! The community and implementers need to be transparent that OpenLMIS is an evolving system though.*