OpenLMIS is a global initiative to support the development of shareable, interoperable, open-source software for electronic logistics management information systems. A community of countries and international stakeholders who believe in working together to solve common challenges designed OpenLMIS; countries and donors have pooled resources to create a non-proprietary product that is built on shared user requirements across countries. The end result is a more flexible and powerful information system than what any one country or organization could create individually. As an open source project, the software is available free of charge, and enhancements made by individual countries are contributed back to the community for others to use.

During the last five years, the OpenLMIS project, community, and software have experienced rapid growth, including six successful implementations, active interest from additional countries, and a strong and growing partner community. Software contributions from five partner organizations – PATH, CHAI, JSI, ThoughtWorks and VillageReach – have demonstrated a significant change from the one-off, business-as-usual LMIS software development practices that OpenLMIS was created to address.

At the same time, the strong uptake of OpenLMIS has exposed a demand for improved extensibility and modularity within the OpenLMIS software. From a donor perspective, a critical priority is the realization of the “build once, use everywhere” vision for OpenLMIS. Investment in a country project that generates new software development should contribute code in such a way that other countries needing similar functionality can use it. Over the last year, the OpenLMIS community has been actively discussing the need for a re-architecture of the platform and is now seeking decisive action on beginning the re-architecture project.

The current $2.5M OpenLMIS grant to VillageReach (May 2015 — May 2017) includes funding for continued OpenLMIS software development, including both sustainment and software re-architecture. In December, VillageReach presented a plan to BMGF detailing how the existing grant funding would be leveraged to re-architect the platform. This plan proposed a multi-year, incremental re-architecture (see Figure 1 below) and concluded that a full re-architecture could not be completed by the May 2017 grant end date. This proposal identified significant risks, including uncertainty about the feasibility of “untangling” coding throughout the system, a long wait time for countries interested in adopting OpenLMIS, and a sustained engineering effort for existing deployments to absorb new releases and refactor their code to comply with it.

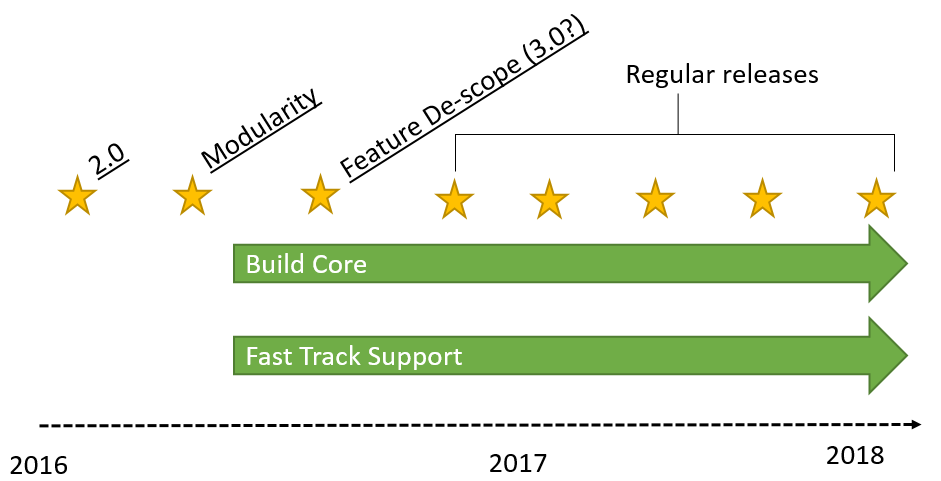


Figure 1: Incremental Re-Architecture Schedule (from Dec 2015)

BMGF raised concerns that during such a long engineering effort they would run the risk of investing in additional “one off” country-level OpenLMIS projects that did not contribute to the overall code base. The VillageReach team was asked to investigate a way to accelerate the re-architecture timeline and to quantify any additional resources needed to do so. This memo is the response to these asks.

In the intervening months, the team has evaluated technology frameworks; analyzed the full scope of variation across existing OpenLMIS implementations; gathered community feedback on the re-architecture proposal; held a weeklong community re-architecture workshop; conducted vendor evaluations; and created a detailed technical design. As an output of this work, we have landed on a technical design approach that will allow significant parallelization of work streams and a plan to deliver an OpenLMIS 3.0 Beta in October 2016 with a full, stable OpenLMIS 3.0 release in February 2017. The 3.0 releases will deliver a full re-write that includes the minimum, highest priority OpenLMIS features needed for a new country implementation. Further, the 3.0 releases will be delivered using existing OpenLMIS grant funding.

Once the initial 3.0 versions are released, however, it will take additional releases on the new modular platform to fully replace existing OpenLMIS functionality and to add the priority backlog features. At the community re-architecture meeting, participants created an OpenLMIS 3.x roadmap (see Figure 2 below). To deliver on the full plan 3.x roadmap, we are requesting supplemental funding.



Figure : Re-Architecture Workshop OpenLMIS 3.x Roadmap Activity

The remainder of this document details each of the projects necessary to deliver on the full re-architecture roadmap, including a mobile reference application and vaccine functionality previous proposed to BMGF by CHAI and VillageReach respectively. To give the foundation funding alternatives we have budgeted each project separately and presented a menu of choices:

1. **Re-Architecture Acceleration (OpenLMIS 3.0)**

Description: OpenLMIS Core Team to produce the re-architected Beta 3.0 and 3.0 stable release, including community support and new implementation support at current levels through February 2017.

Timeline: May 2016 – February 2017

Grantees: VillageReach (prime), SolDevelo (sub)  
 Budget: No additional funding needed, covered under current grant

1. **3.x Roadmap and Community Support**

Description: OpenLMIS Core Team funding to produce OpenLMIS 3.1 and 3.2 releases in 2017 and to continue to support the OpenLMIS community and new implementations through June 2018.

Timeline: March 2017 – June 2018

Grantees: VillageReach (prime), SolDevelo (sub)

Budget: $1,964,174

1. **OpenLMIS Mobile Reference Application**

Description: CHAI-led project to develop and pilot an OpenLMIS mobile user interface to support OpenLMIS requisitioning and inventory management at last mile service delivery points (SDPs).

Grantees: CHAI (prime), ThoughtWorks (sub)

Timeline: Oct 2016 – Nov 2017

Budget: $1,128,150

1. **OpenLMIS Vaccine Supply Chain**

Description: Addition of a flexible, configurable vaccine module to OpenLMIS supporting both informed push and standard vaccine supply chain models.

Grantees: VillageReach (prime), JSI (sub), CHAI (sub), PATH (sub)

Timeline: Aug 2016 – Nov 2017

Budget: $951,989

Note that Project #1 on the menu (Re-Architecture Acceleration) is already fully funded, but funding for Project #2 (3.x Roadmap and Community Support) is a pre-requisite for the completion of either Project #3 (Reference Mobile Application) or Project #4 (Vaccine Supply Chain).

The OpenLMIS project team at VillageReach will coordinate these four projects, including designing solutions architecture and managing dependencies as needed. Key milestones for project delivery are shown below.

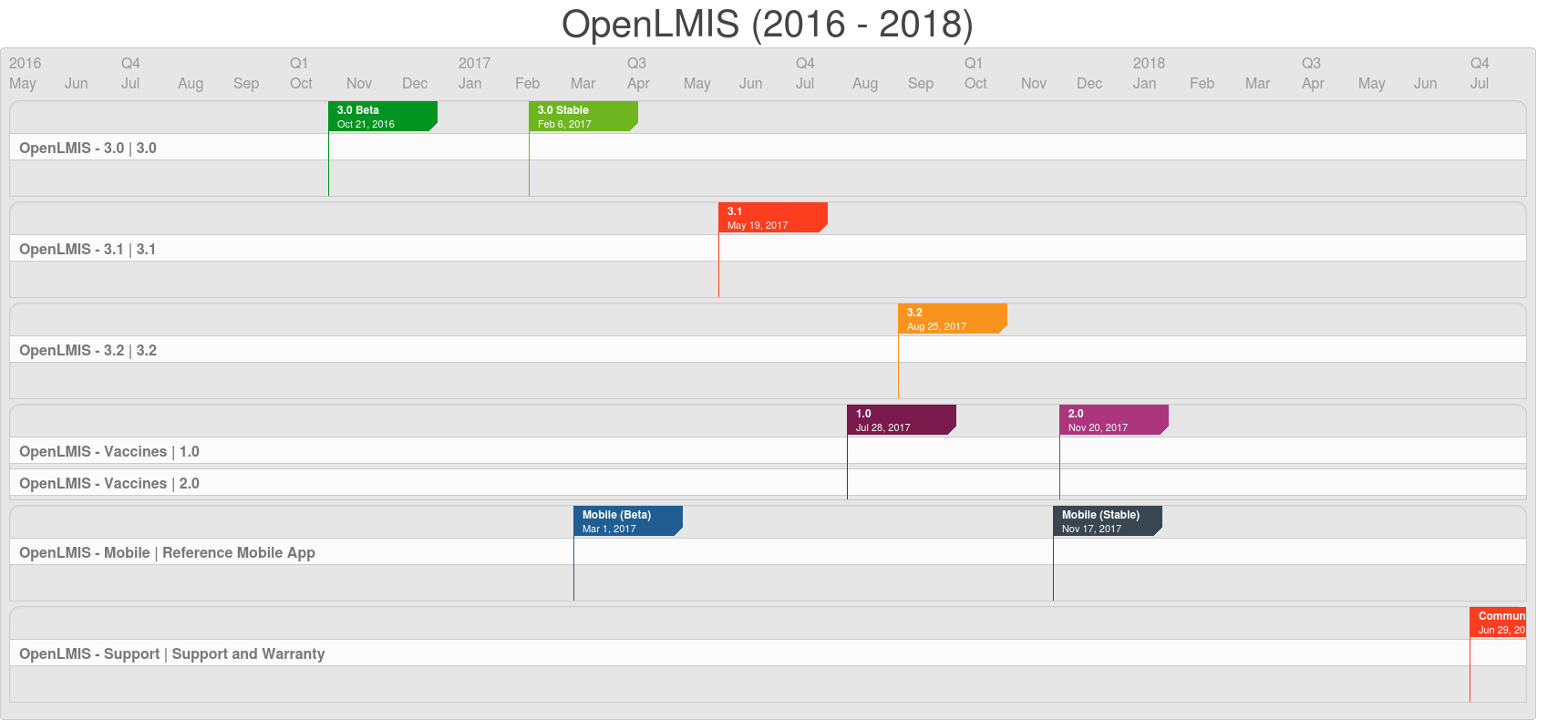


Figure 3: Proposed OpenLMIS 2016 – 2018 Roadmap

1. **Re-Architecture Acceleration: *Time-based releases; Modular & Extensible Architecture***

The increasing level of country interest in using OpenLMIS—in addition to partner interest in contributing software to OpenLMIS—requires a solution that meets the needs of countries looking to implement OpenLMIS soon (e.g. Malawi, Uganda, Senegal, potentially Nigeria) while simultaneously ensuring that any new feature development will be done following the new modular architecture and can thus be re-used. Getting OpenLMIS 3.0 out as soon as possible will also prevent further fragmentation in the OpenLMIS deployments and obviate the need to pay for migrations.

The VillageReach OpenLMIS team led a community-wide effort to collect and analyze the [features in use in each existing OpenLMIS implementation](mailto:https://openlmis.atlassian.net/wiki/display/OP/Feature+Variability+Matrix#https://openlmis.atlassian.net/wiki/display/OP/Feature+Variability+Matrix). This information was used as input for a [weeklong design workshop](https://openlmis.atlassian.net/wiki/display/OP/March+2016+Re-Architecture+Design+Workshop), hosted and facilitated by VillageReach in Seattle March 28 – April 1, with CHAI, Thoughtworks and PATH in attendance. (JSI was unable to attend, but we held several calls to review progress and findings.) Together, we reviewed the existing features, discussed gaps in the current feature set, and defined the scope for the ‘Beta’/3.0 release as well as several, significant point releases to follow. The analysis of the current and envisioned future feature set informed the target architecture and scope of a 3.0 release to address the best way to maximize existing investments.

The overall objectives—to better enable contribution and maximize shared benefit by promoting code reuse and transferrable customizations—have not changed, but we have moved from a feature-based release schedule and monolithic ‘core’ architecture to a time-based schedule with modular architecture. Instead of an incremental approach, we are recommending a rewrite.

The rewrite option has several advantages over the incremental approach:

* Faster time-to-market for a modular, extensible 3.0 architecture
* Less disruption to existing implementations; the incremental approach would require a more sustained—and expensive—refactoring effort over time
* Cleaner, more extensible model and ability to more easily introduce new technologies for automation
* Ability to work on parallel development/clearer demarcated code

Project Approach

To quickly achieve the objective of a complete re-architecture, VillageReach and SolDevelo (a software development partner) will rewrite the existing OpenLMIS software in multiple, concurrent work streams from the ground up, reusing existing code (e.g. calculation algorithms) whenever possible, decomposing the existing monolith into logically and physically separate, headless services.

Before arriving at our current architecture, VillageReach reviewed the architecture of other notable systems in the open-source health space and met with multiple teams and individuals: Grameen’s MoTECH team; Darius Jazayeri, Principal Architect of OpenMRS; Thoughtworks, who did most of the engineering on the first version of OpenLMIS; SolDevelo; and InSTEDD.

We framed our discussions using the existing data of whatis variable across the OpenLMIS implementations *today* (i.e. how implementers have actually customized OpenLMIS); how certain existing country-specific implementations could be made to be re-usable; and what made the most sense in terms of complexity and future flexibility.

There are pros and cons for the various approaches we reviewed, the discussion of which could be the subject of a memo of this length. In the end, we opted for the middle ground between a pure microservices approach and a monolith with extensions. No architecture can provide high enough guardrails to alone enforce governance of good design that promotes re-usability through the code; rather, a combination of reasonable guardrails, a well-lighted path for development and contribution, and a dedicated community ready to engage will be the safest course.

The following is the technical and project approach for accelerating the completion of the 3.0 release.

Technical Overview

As shown in Figure 4, the new architecture provides extensibility to accommodate several different approaches for feature development and/or customization:

1. *Independent Services* that communicate through IPC (RESTful APIs) for different functional areas. E.g. Requisition Service, Informed Push Service, Inventory Management Service, etc. An *Independent Service*:

* Defines the interface and default behavior
* May expose Extension Points that a Module may utilize to extend its behavior
* Provides Data Store, logging, and authorization services to an Extension Module
* Is hosted in own code repository, e.g. *openlmis-serviceName*
* Is a Spring Boot application that runs its own HTTP server and exposes a RESTful API
* Is extensible through the use of Extension Module(s)
* Is hosted in 1 + M (1 for Spring Boot and M for the Extension Modules) Git repositories
* May expect a PostgreSQL DB connection setting to be provided to it through the environment
* Is packaged and deployed as an Image or Container for rapid deployment

1. *Modules* for extension *within* an independent service. For example, as an Implementer, I want to add a new calculation for order quantity to the Requisition Service; add a new accounting policy to the Inventory Management Service, etc. A *Module*:
   * Has a Spring context that uses a Service's extension point(s) to define new behavior
   * Is packaged and deployed as a JAR through Maven
   * Is hosted in own code repository, e.g. *openlmis-serviceName-moduleName*

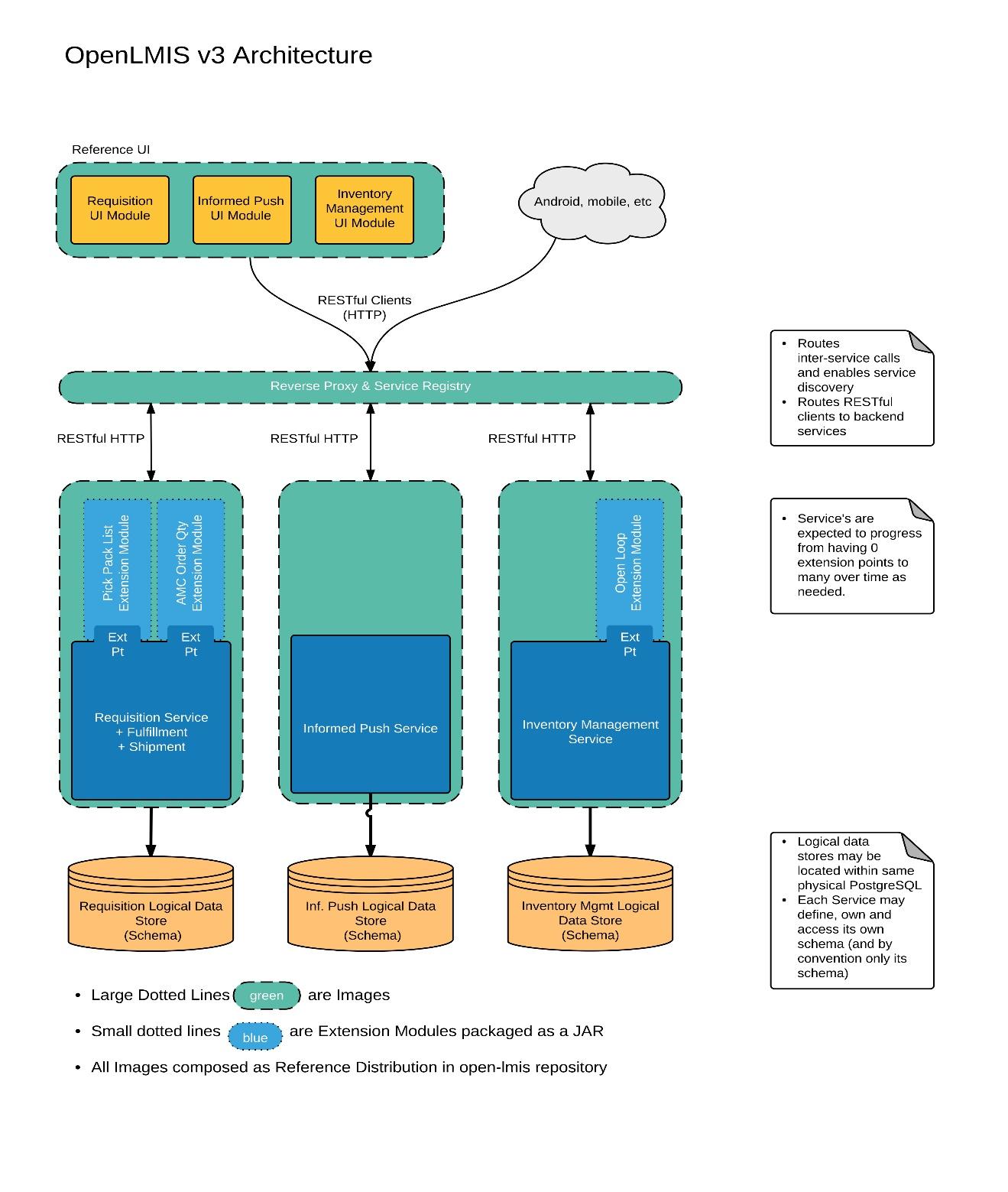


Figure 4: Headless Services with a Reference UI

To date, much of the forking of the OpenLMIS project was due to the need to customize the UI along with adding new features. In the new architecture, the *Reference UI*, which comprises the collection of UI modules corresponding to services, will be packaged and deployed separately from the services and should be thought of as a reference point from which an implementer can start to customize a particular implementation; forking of this repository is expected. The *Reference UI*:

* Uses Modular JS framework to compose Reference UI Modules
* Is Packaged as JAR/TAR/ZIP
* Is Hosted in own code repository, e.g. *openlmis-reference-ui*

Each Independent Service will likely have a UI, and these components will be modularized in a similar fashion. A *Reference UI Module:*

* Includes the web-components (JS, HTML, CSS) that make up a Service's (e.g. Requisition) user interface
* Is packaged and deployed separately from the Service
* Is hosted in own code repository, e.g. *openlmis-serviceName-refUI*

OpenLMIS 3.0 will also produce a *Reference Distribution*, which is the recipe or instructions to assemble a collection of services that comprise the ‘standard’ deployment. This recipe will evolve over time to include new features and services as they mature, and an implementer would be free to fork and customize this recipe for their own particular needs. A *Reference Distribution*:

* Is composed of Image/Container composition file(s) for wiring Independent Services together and deploying the Reference UI
* Includes a Reverse Proxy and Service Registry so that all of the Independent Services may communicate
* Includes script(s) to manage deployment configuration
* Is hosted in own code repository, e.g. *openlmis-distro*

Detailed information on the proposed architecture can be found on the OpenLMIS wiki: <https://openlmis.atlassian.net/wiki/display/OP/OpenLMIS+v3+Architecture+Overview>

Key Activities

This project will include the following key activities and deliverables:

1. **3.0 Beta Release (October 2016).** Beta release of re-architected OpenLMIS platform, including the following features:
   1. Modular Architecture and Extension Framework
   2. Reference UI (Web)
   3. Requisitions, including Offline Support
   4. Orders
   5. Inventory Management (25%), including GS1 support[[1]](#footnote-1)
   6. Program Data (fully extensible and customizable)
   7. Export for Order Conversion (ERP Integration)
   8. Built-In Reporting (50%)
2. **3.0 Stable Release (February 2017).** Stable 3.0 release suitable for country deployment, including all beta features as well as the following:
   1. Fulfillment/Shipping
   2. System Management/Configuration (via UI)
   3. Reference Data Import/Uploads
   4. Inventory Management (50%)
   5. Built-In Reporting (complete)
   6. Reporting Framework for *ad hoc* Custom Reports (50%)
3. **Quality Assurance and Automated Testing.** A key tenet of Core OpenLMIS is that it is highly reliable and tested. The new version will include automated test coverage as well as fully documented manual regression testing scripts, where applicable.
4. **Technical, End User and Administrator Documentation. S**upporting documentation to enable developers, implementers, users, and administrators to understand and use OpenLMIS will be created over the course of the project and will be made available on the OpenLMIS Website and wiki. This activity includes contracting with a technical writer for user and administrator documentation.

Project Delivery Approach & Timeline

In planning the project approach, we held in-depth discussions with multiple potential software development partners. SolDevelo was selected based on their solid past performance history with VillageReach, MoTECH and the International Rescue Committee; ability to quickly scale up or down capacity; cost; and willingness to work under the close direction of VillageReach.

In order to accelerate project delivery, the re-architecture project will parallelize development of epics utilizing a scaled agile approach. As shown in blue in Figure 5 below, agile development teams of 3-5 staff will each be dedicated to development of specific epics. VillageReach will staff one team internally and scale development with additional SolDevelo teams as needed during various phases of the project. All work will be overseen and coordinated by the lead VillageReach architect and technical project management, who in turn will be accountable to the OpenLMIS community.

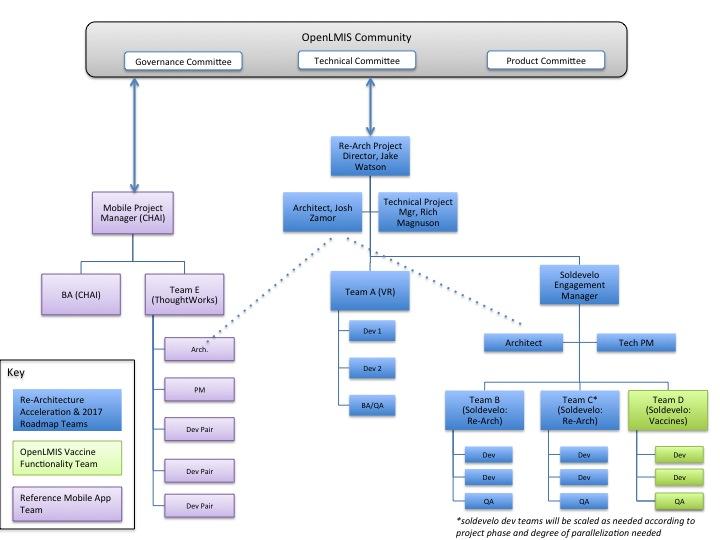


Figure 5: Team Structure

At the outset of each project work stream, team leads will co-locate to develop a detailed design and a work plan. Once development commences, all teams will follow the same sprint schedule and use the same project management tool (the publicly accessible OpenLMIS Jira site). “Scrum of scrum” meetings will be used to coordinate work and dependencies across teams.

A key component of the current OpenLMIS grant has been building a core OpenLMIS software development team that serves as a consistent resource and “knowledge bank” for the OpenLMIS development community. Outsourcing of all OpenLMIS software development during the initial OpenLMIS software development resulted in a lack of institutional knowledge and ability to support the open source community. The proposed approach combines the cost effectiveness of outsourced software development with an in-house team to provide technical leadership, project stewardship and quality control.

The project schedule below (Figure 6) outlines the development schedule for each 3.0 epic, as well as release milestones.

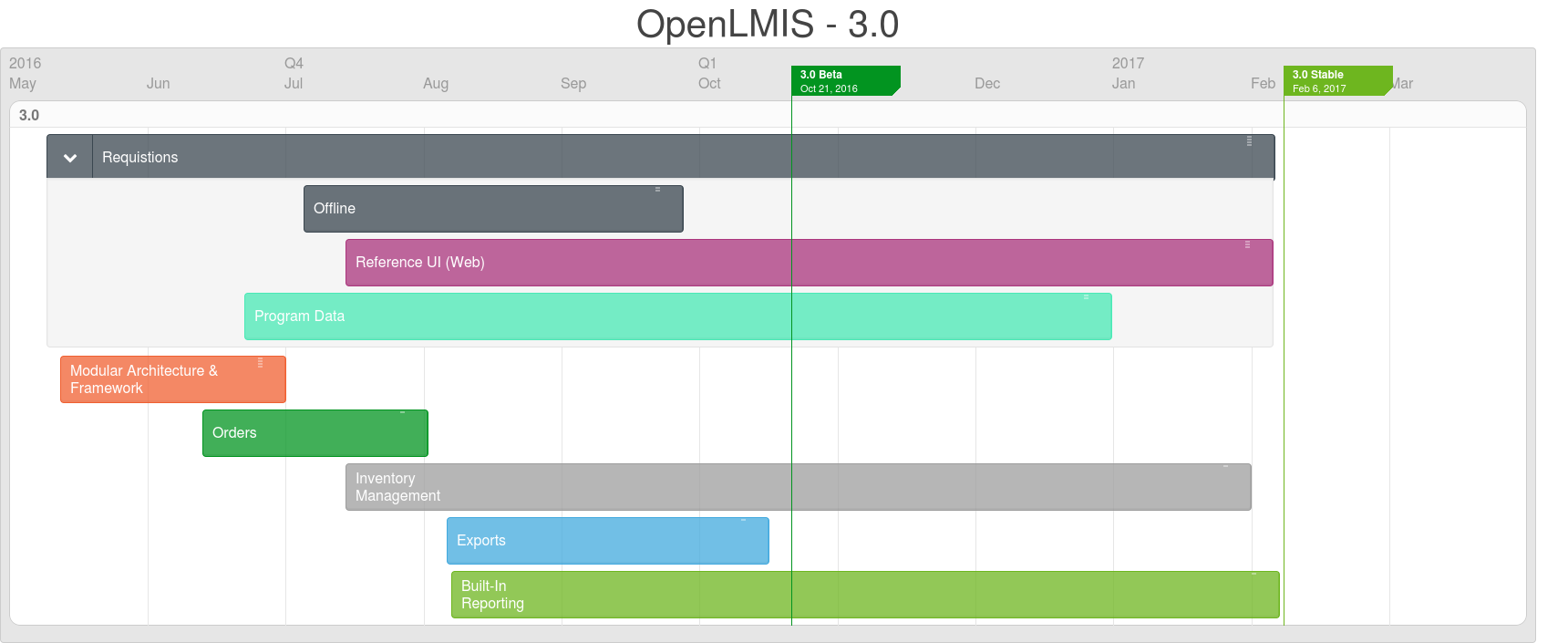


Figure 6: OpenLMIS 3.0 Timeline

Budget (No additional funding necessary)

The cost to fund accelerated delivery of OpenLMIS 3.0 and maintain the current level of community/implementation support activities is approximately $1,850,000. This will be funded completely through the existing OpenLMIS Critical Mass grant by increasing the burn rate on existing funds, by bringing in the SolDevelo team, and re-allocating funds between budget categories (e.g., moving travel to subcontracts). To date the OpenLMIS Critical Mass grant has been underspent due to a diversion of resources to the VIMS project throughout 2015, leaving unspent funds that can be targeted towards the re-architecture acceleration. If the foundation team concurs, we will request this budget reallocation when submitting our May 2016 grant progress report.

Note that by increasing the project burn rate **all funding will be exhausted at the end of February 2017, and no technical support or community management will be provided after that date.** Delivery of these services through June 2018 is included in the *OpenLMIS 3.X Roadmap* project proposal that follows.

1. **OpenLMIS 3.X Roadmap**

The *OpenLMIS 3.x Roadmap* project will continue delivery on the re-architecture project roadmap. This will include OpenLMIS 3.1 and 3.2 releases to build out functionality that will complete the re-architecture scope. In addition, this project will fund community support through June 2018. After OpenLMIS 3.0 is released, we believe that there will be a continued need to educate and train implementers on the 3.x architecture on how best to extend and customize it; continue to support and nurture the nascent OpenLMIS community until such time that it is self-sufficient; and support countries evaluating OpenLMIS implementations. Thus, we are seeking funding for the OpenLMIS team through the June 2018 to engage and support these efforts. Resourcing will be scaled down as the OpenLMIS project matures and moves toward a more sustainable cost structure.

Key Activities

This project will include the following key activities and deliverables:

1. **OpenLMIS Deployments or Expansions in 3-5 geographies:** This funding will support continued advocacy for country implementations and pilot support to implement or expand OpenLMIS in 3-5 geographies. We are open to discussing an increase in the target metrics based on the expanded project timeline.
2. **OpenLMIS 3.1 Release (May 2017):** The 3.1 will further expand the OpenLMIS product feature footprint to include the following epics:
   1. Inventory Management (complete)
   2. Built-In Reporting (complete)
   3. Reporting Framework (complete)
3. **OpenLMIS 3.2 Release (August 2017):** The 3.2 release will complete migration of features from the 2.0 to 3.x platform, including delivery of the following epics:
   1. Dashboards
   2. Inventory Management, Offline Support[[2]](#footnote-2)
   3. Equipment Management (including Cold Chain Support)
   4. GS1/Barcode Scanning Support
4. **Warranty & Stabilization Period (September – June 2018):** The OpenLMIS core technical team will provide stabilization and support in the post-3.2 release period, including:
   * 1. Bug fixes
     2. Technical advising for implementations, upgrades and migrations
     3. Community support
5. **Quality Assurance and Automated Testing.** A key tenet of Core OpenLMIS is that it is highly reliable and tested. The new version will include automated test coverage as well as fully documented manual regression testing scripts, where applicable.
6. **Technical, End User, and Administrator Documentation.** Supporting documentation to enable developers, implementers, users, and administrators to understand and use OpenLMIS will be created over the course of the project, and made available on the OpenLMIS Website and wiki. This activity includes contracting with a technical writer for user and administrator documentation.
7. **Community Management**: The OpenLMIS team will provide community management including a 2016 and 2017 community meetings (likely in Africa), facilitation of committees, and expanding the set of partners contributing to the OpenLMIS community.

Project Approach & Timeline

The delivery approach for this project will continue the strategy outlined in the OpenLMIS Re-Architecture Acceleration section above (see p. 9). VillageReach will lead software development, augmented by resources at SolDevelo, and teams will coordinate using a scaled agile approach. The activity timeline for this period is show in the diagrams below:

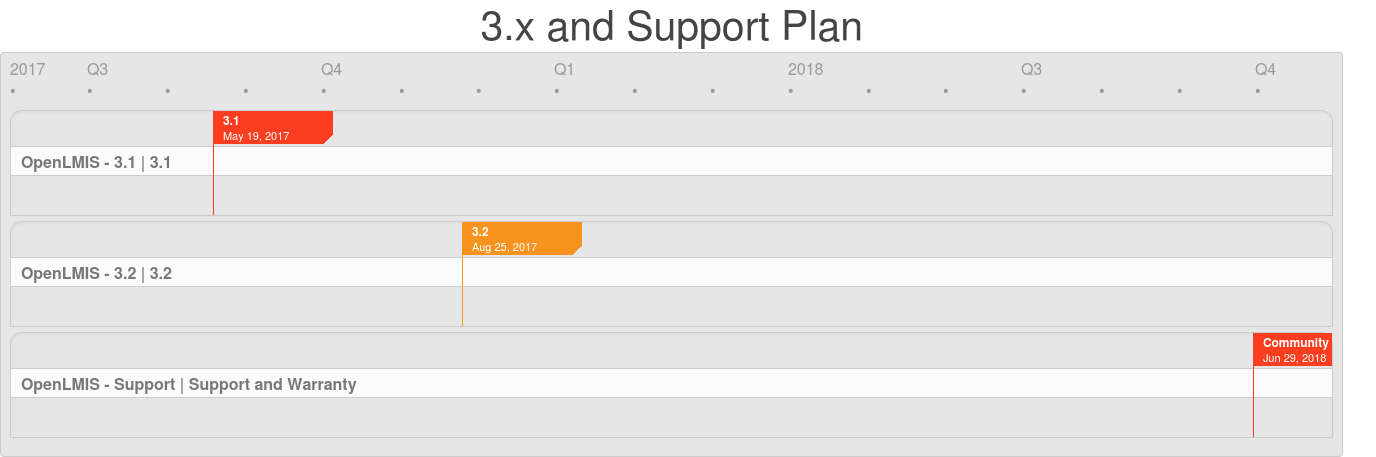


Figure 7: 3.x and Support Timeline

**3. OpenLMIS Mobile Reference Application**

The OpenLMIS product vision includes both a Web-based and mobile (tablet/phone) user interface. While OpenLMIS Web serves the needs of users at the district-level, regional-level and hospitals, those at the last mile of the supply chain require a lower-cost, offline-enabled mobile solution. Ministries of Health in several countries—including Uganda, Malawi, Nigeria and Zimbabwe—have particularly identified the access to quality, accurate, and timely facility-level logistics data as a key challenge.

The OpenLMIS 3.0 platform will provide a core set of APIs that can be integrated with any existing mobile stock management tool. While many mobile-based products exist, no tool currently meets the requirements of multi-program stock management for 100+ commodities or provides the integration with automated requisitions and higher-level systems. CHAI piloted facility-based Android systems in Mozambique using CommCare Supply and in Nigeria using DHIS2, and has conducted in-depth analyses of systems across countries to inform requests from MOHs. This analysis has identified a number of gaps:

* Focus on tracer commodities – not routine replenishment (e.g. CommTrack)
* Proprietary and dependent on specific development teams for customization (e.g. Logistimo)
* Unable to integrate with higher level systems in an easy manner (e.g. CommCare Supply),
* Does not incorporate business logic needed for inventory management (e.g. DHIS2)

As a result of these learnings, CHAI Mozambique has since built and piloted a facility-based Android application on OpenLMIS with good results and strong buy-in from users. Given the demand and significant overlap in facility LMIS requirements across countries, CHAI proposes building a reusable OpenLMIS reference mobile application for facility stock management on OpenLMIS.

Similar to the OpenLMIS reference Web application, this application will provide flexibility to configure and customize a similar application for implementations across multiple countries in lesser time and at much lower cost. Countries can also leverage and use enhancements made by other countries and organizations at much lower cost than developing functionality individually. Building on OpenLMIS will ensure integration with a robust LMIS for higher levels of the supply chain to automate requisitions and reporting, and facilitate connections to ERPs and other reporting systems such as DHIS2 with the relevant APIs.

Project Approach

The reference mobile application will be built based on business requirements for facility stock management across several countries to ensure relevance across countries and inform design decisions around configurability and user interface. A multi-country requirements gathering workshop will be organized with participants from Ministries of Health and local country teams from across at least three countries[[3]](#footnote-3). The workshop will span two weeks and will dive deep into processes and desired user interfaces for requisitions; receiving and issuing stock; adjustments; reports; and country-specific functionality.

Functionality

The reference mobile application is expected to cover the following functionality. All functionality will be available offline with periodic data syncs to the OpenLMIS server to facilitate the requisition and receipt processes, and reporting.

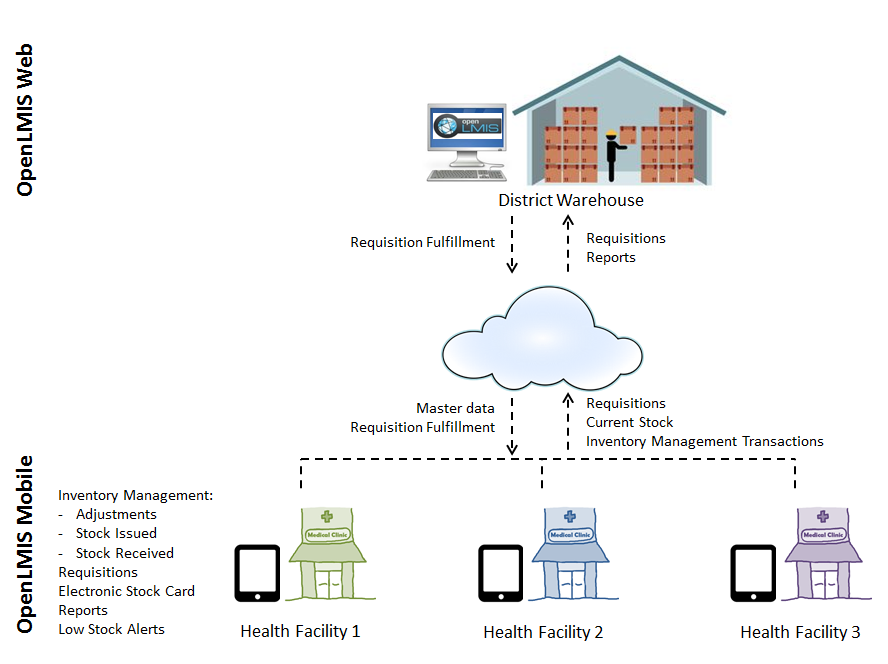


Figure 8: Mobile Reference Application

Key functional areas include:

* Requisitions: place orders to relevant higher-level stores for commodities in each program, based on a schedule or on an emergency basis. The system will support both pull and push systems.
* Receipts: receive stock through fulfillments of orders or from other sources such as donations, private sector, or stores that do not have OpenLMIS
* Issues: record stock issued to wards, to other facilities, or community health workers, to update stock balance
* Adjustments: record losses (with loss reasons), physical stock counts based on a schedule, and other adjustments
* Reports: the system will cover basic reports, such as stock on hand, low stock, monthly consumption. This will also include an electronic stock card to reflect paper stock cards and for audit purposes.
* Alerts: receive alerts when stock is below minimum or stocked out, and notifications for when orders or physical stock counts are due
* Batch functionality: be able to track and record transactions by batch numbers

Technical Overview

The reference mobile application will mirror the domain model of the OpenLMIS platform. Similar to the core system, it will be developed as two primary parts: a Product Core will contain all the reusable components, including the domain model, data and data relationships, and a set of standard operational processes to cover the stock management requirements mentioned above. This product core will also include infrastructure support for security, synchronization, automatic updates, error handling, etc., and the management and authentication of user accounts. In addition, a Reference UI will contain default UI representations, process configuration options, configuration of validation and calculation logic, and look and feel. For a more detailed technical overview, please reference the included document “LMISAndroidApplicationProductization.pdf”.

Field Test Approach and Country Interest

The development of the reference mobile application will be immediately followed by configuration and field-testing of the app for one country, to ensure strong coupling between development and country needs. Based on strong MoH interest, this first country is likely to be Uganda. Various programs at the Ugandan Ministry including the National Medical Store (NMS), UNEPI (Uganda National Expanded Program on Immunisation), the Pharmacy Department, Child Health, Reproductive Health and the Family Planning departments have expressed interest in an electronic LMIS at the facility level with a basic system at the district level, followed by a full functional scale-up of OpenLMIS across all levels of the supply chain. CHAI is discussing funding for this deployment with DFID Uganda.

The Malawi MoH has also been highly interested in a deployment of an eLMIS at the facility level for requisition submission as one of their top priority project roadmap items. In 2015 MoH and eLMIS Steering Committee have chosen OpenLMIS as the system that will replace the current supply chain system to submit electronic requisitions for all programs (except vaccines) at district and higher levels. It would be highly complementary to run a facility-level pilot using the reference mobile application.

Project Approach & Timeline

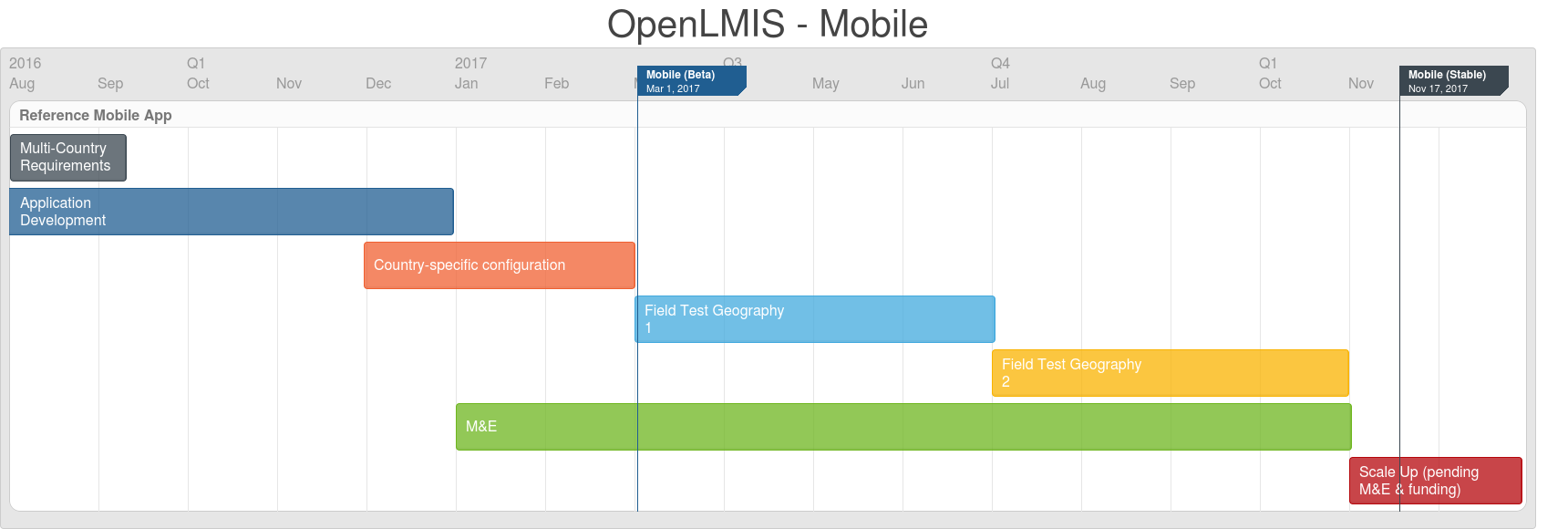


Figure 9: OpenLMIS Reference Mobile Application

CHAI would be the lead on this proposed work, in close collaboration with the VillageReach OpenLMIS team. The development of the reference mobile application will be done in parallel to the core OpenLMIS re-architecture work, using the team scaling strategy outlined on p. 10. Development on the mobile application will begin as soon as the OpenLMIS 3.0 Beta is released in October 2016. At this point in the project, a reference Web interface will have already been delivered, and the mobile application will consume the same APIs already developed for the Web interface.

VillageReach, CHAI and ThoughtWorks collaborated successfully to build the Mozambique mobile application (VillageReach providing backend stock management functionality, and ThoughtWorks developing the mobile application and customizing the Web UI to meet project needs), and a similar approach would be used for this project. Collaboration will be managed via the OpenLMIS Technical Committee, and the OpenLMIS Product Owner via the OpenLMIS Product Committee will coordinate product decisions. The budget below assumes resources already allocated for the core OpenLMIS project team (software development manager, product owner, architect, etc.) are fully funded through the re-architecture project and will be able to allocate a percentage of their time to supplemental projects like this mobile reference application to collaborate with the CHAI/Thoughtworks team.

**4. OpenLMIS Vaccines Functionality**

The current global momentum to accelerate next-generation immunization supply chains (NexGen iSC) has illuminated the need for quality, re-usable electronic logistics management information systems that meet the specific requirements of immunization programs. Through ongoing work with immunization information systems in Mozambique, Tanzania, Benin and Nigeria, VillageReach is seeing a consistency in requests coming from EPI programs and their partners regarding the functionality and reporting needed to manage vaccine logistics and program data.

These requests include a common need across countries for flexible vaccine data collection tools; a consolidated view of vaccine service delivery and logistics data; cold chain inventory management; flexible demand planning tools; integration with/replacement of the DVD-MT; and vaccine stock management. VillageReach and UNICEF’s NexGen iSC work—and the work of partners in other countries—are likely to uncover similar requirements and requests. By systematically collecting these requirements and developing an OpenLMIS feature release in a fully configurable way, additional countries will be able to implement OpenLMIS to support their iSC with lower financial and technical barriers to entry.

OpenLMIS 2.0 does not have a module ready for deployment in the next country requesting these features. Much of the current EPI functionality in OpenLMIS 2.0 was developed as custom, country-specific work and was not created in a globally configurable way. As demonstrated by the Tanzania Vaccine Information Management System (VIMS) project, the lack of a process to collect and document multi-country user requirements up front and a commitment by all partners and donors to build re-usable, flexible, and configurable software, results in software that will only be country specific.

Because this cross-country requirements work has not yet been completed, **OpenLMIS 3.0 has not been scoped to include any specific vaccine management functionality** (for informed push or standard distribution models). This OpenLMIS Vaccine Functionality option would add a globally reusable vaccine module, configurable to the needs to specific countries, in the OpenLMIS 3.2 release (August 2017).

A key component of this project will be leveraging vaccine supply chain requirements already developed for specific country projects, including VIMS; the BID Initiative; Mozambique and Benin informed push systems; the VAN project (including both global and country-specific work); and the initial OpenLMIS CRDM process. Requirements documents, process maps and software artifacts from these projects will be analyzed to form a base set of OpenLMIS vaccine requirements. This work stream will include sub-contracts to CHAI, PATH and JSI to support analysis and joint requirements development. These requirements will be further refined through country-level discovery and a multi-country requirements workshop.

Functionality

The bulk of the work on new vaccine-specific functionality for OpenLMIS falls into the following areas:

1. Front-end data collection flexibility and configurability for different vaccine supply chain designs and associated workflow variations (i.e., push and pull distribution models)
2. Vaccine reporting and analytics
3. Vaccine demand planning (ideal stock amount, population data management, replenishment calculations)
4. Vaccine stock management, including accounts for doses vs. vials, and robust tracking of expiry dates, batch and VVM status (optionally via bar coding)
5. Vaccine cold chain equipment management and integration

The project may also including the following components, based on prioritization:

1. Facility-based mobile module to support vaccine stock management and vaccine receiving
2. Mobile (tablet-based) module to support informed push vaccine distribution
3. Integration with vaccine registry tools

Key Activities

* Collect comprehensive vaccine supply chain management requirements from partners and ministries of health across three to five countries, utilizing both existing requirements and requirements discovery
* Engage global stakeholders (WHO/UNICEF Hub, Gavi, etc.) to integrate global guidance on NexGen iSC, product requirements from the Gavi Data for Management Priority Working Group and country-level requirements
* Conduct requirements validation workshop with partners and MoH representatives to prioritize requirements
* Create high-level design, emphasizing code reusability and modularity
* Develop OpenLMIS vaccine functionality, ensuring global configurability and reusability
* Release documentation and support OpenLMIS community to deploy solution

Project Approach & Timeline

Vaccines functionality will be built out using the modular design for OpenLMIS 3.x, extending OpenLMIS to support specific vaccine-program needs. During high-level design the solution architecture will be defined, including determining what services, modules and extension points are needed to support vaccine supply chain needs.

Software development will be coordinated with overall OpenLMIS 3.x release schedule, as shown in the high-level schedule below. The project is expected to take 16 months. A major contributor to the length of this project is the time needed to work with country and global stakeholders on common requirements for vaccine management.

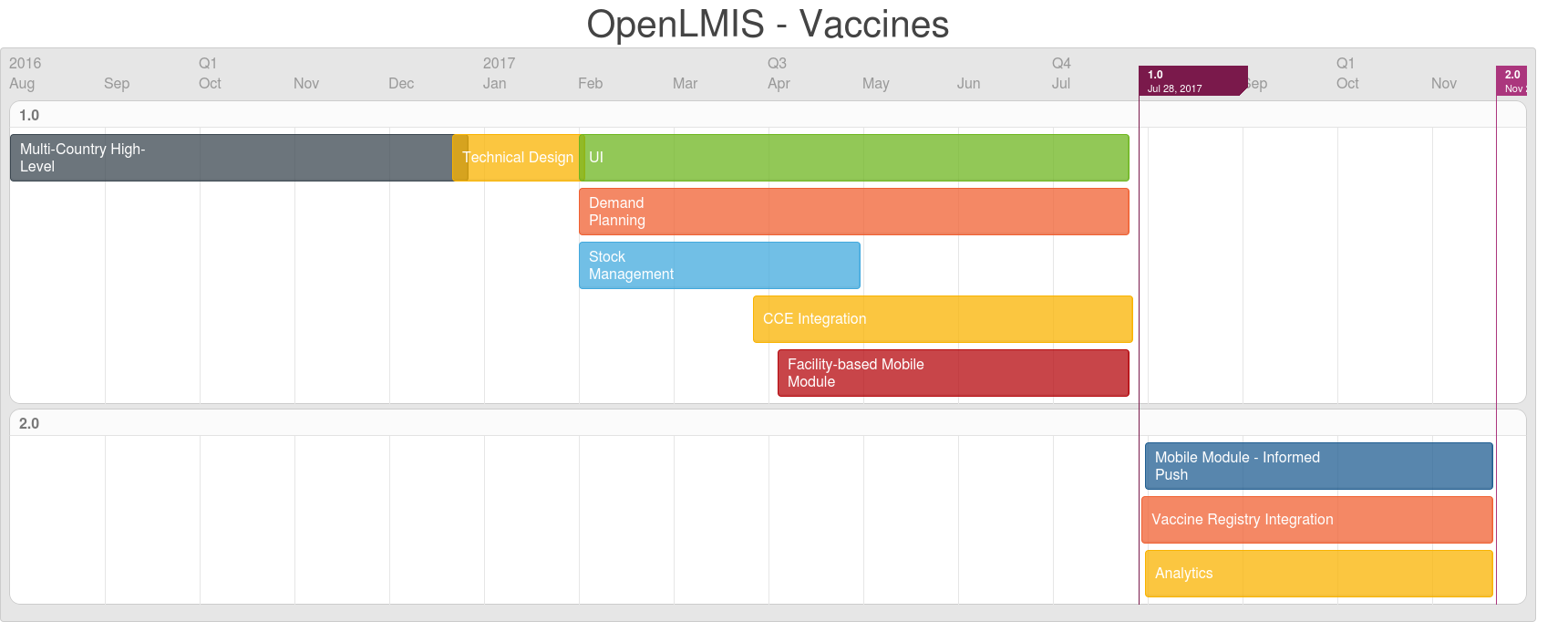


Figure 10: OpenLMIS Vaccines

1. Support in the Data Model only; no feature development at this time. [↑](#footnote-ref-1)
2. E.g. Pick Pack, Offline Stock Adjustments [↑](#footnote-ref-2)
3. The three countries to participate in the multi-country requirements gathering will likely be Uganda, Malawi, and Zimbabwe, based on current MOH interest and priorities, donor interest, and in-country capacity. [↑](#footnote-ref-3)