Good to Great FOSS: Learnings from Africa

Prepared for IDRC by
Allen Gunn
Aspiration
10 March 2008

San Francisco Nonprofit Technology Center, 1370 Mission St., San Francisco, CA 94103
Phone: (415) 839-6456 • info@aspirationtech.org

Table of Contents

1. Introduction

The International Development Research Centre (IDRC) invited Aspiration to design and facilitate an event focused on Free and Open Source Software (FOSS) development in Africa. The workshop took place from the 24th to the 26th of October 2007 in Nairobi, Kenya.

The Acacia program at IDRC had funded a number of FOSS initiatives over the last 3 years and had come to the conclusion that knowledge of good practice in FOSS development was extremely varied, and that little documentation existed on how to establish and sustain a FOSS project in a developing country context.

The decision was been made to bring several of the Open Source projects that IDRC had funded together in a workshop to talk and learn about what constitutes good practice in developing an Open Source project in Africa. During the course of this 3-day meeting, projects met, demonstrated their tools, compared processes, discussed challenges facing FOSS practitioners in Africa, and considered what was required to sustain and grow the FOSS movement on the continent.

Goals and focus of this paper

The purpose of this paper is to document the state of open source software development in Africa from the perspective of the projects that participated in Good to Great FOSS. In addition, this paper includes an overview of best practices for open source development in the African context as detailed by event participants, as well as a summary of recommendations made at the event on how to better support and propagate open source efforts in Africa.

The paper is divided into several sections:

> **Challenges for FOSS in Africa** examines the current environment and the barriers to success for FOSS projects in the African context.
>
> **FOSS Community Processes** explores and contrasts the non-technical aspects of two African FOSS projects, including how they manage their development while governing and scaling their communities.
>
> **FOSS Development Processes and Development Environments** details the processes and tools used by participants at Good to Great FOSS to manage and deliver their projects.
>
> **FOSS Licensing** contrasts the different licenses used by the projects at Good to Great FOSS, explaining larger license categories and summarizing the related issues of license interoperability.
>
> **FOSS Business Models and Sustainability** surveys the range of of business models extant in the FOSS community, and summarizes a business modeling exercise done at the event.

Best Practices for FOSS in Africa provides a set of best practices for open source projects working in the African context.

Recommendations for a Better FOSS Future in Africa details ideas put forth by participants on how the FOSS movement in Africa might be better supported and developed.

Appendices are provided to document the agenda, the participants, and post-event assessments.

Participating Projects

A diverse group of projects participated in Good to Great FOSS, spanning a range of application areas:

OpenMRS (www.openmrs.org): OpenMRS is a customizable patient-centric electronic medical record (EMR) system. A forms-based system allows updates to be entered at each patient visit. Data can be entered by a physician during patient visits or by data assistants after the fact. The project has developed a rich and open data model for supporting the application. OpenMRS uses the OpenMRS Public License, which is very similar to the Mozilla Public License.

AVOIR/Chisimba: Chisimba is the Malawi word for "framework", and Chisimba is a rewrite of the KINKY project. Whereas OpenMRS is a dedicated application focused on electronic medical records, Chisimba is a web application framework which can be utilized for building a range of different software applications. The architecture is modular, enabling new functionality be developed independent of all other development.

DrumNet (www.drumnet.org/projects.htm): DrumNet is a supply chain communication platform that utilizes cell phones and the internet to allow end-to-end communications between supply chain players in African agricultural markets. DrumNet is a newer and less mature platform than OpenMRS or Chisimba, and does not as of yet have a community to speak of beyond the two primary developers.

Mifos (http://www.mifos.org): Mifos is open source software for microfinance. They are creating a new service model that will increase access to technology for all microfinance institutions, ultimately enabling them to extend their reach to the world's poor.

Tradenet (http://www.tradenet.biz): Tradenet provides market intelligence for farmers and traders worldwide. Tradenet facilitates communication along supply chains using the internet and mobile phones. The Tradenet platform is not currently open source, and Tradenet participated at Good to Great FOSS in an exploratory context.

2. Challenges for Free and Open Source Software in Africa

Central to understanding the state of free and open source software develoment in Africa is awareness about the challenges and issues facing open source practitioners on the continent.

Lack of FOSS awareness and propagation of myths: A fundamental challenge in driving FOSS adoption in Africa lies in addressing the gap between perception and reality. Confusion and mis-information persist, even as mature projects deliver value in a number of contexts. FOSS is unknown in large parts of the continent.

As FOSS advocates work to improve the perception and adoption, they spend substantial time countering biases. Two in particular stand out:

- **Attitude toward local products in general**: There is often a presumption of inferiority for products developed in Africa, and FOSS is no exception. The idea that Africans can develop top-tier offerings is foreign to many on the continent.

- **Attitudes towards FOSS in particular**: The notion that technology created in a developing country can be as stable or as powerful as those from corporations like Microsoft, developed in western countries, is something many Africans refuse to entertain. FOSS is often presumed to be second rate, and as such not worth of consideration.

Slow institutional adoption: In parallel with these biases come institutional barriers. Government and education, the two entities most able to shape perception and awareness regarding FOSS, remain largely resistant to FOSS offerings. Simply put, there is limited future upside for FOSS until it is taught in schools and utilized at regional and national levels. Factors limiting this uptake include:

- **Governmental factors**: There are host of political issues extant in convincing governments to adopt and support FOSS initiatives. Risk-averse politicians and bureaucrats juggling priorities don't want to take on the perceived unknowns of FOSS offerings. Governments seeking foreign investment and courting technology corporations to strengthen their presence in the region don't want to offend. And proprietary vendors have substantial resources with which to engage and shape governmental perceptions and priorities.

- **Educational Factors**: The education system as a rule does not adopt or acknowledge FOSS. In part this is dues to lack of awareness and understanding on the part of instructors. Where computers skills are able to be taught, proprietary tools are considered to offer more marketable and appropriate job skills.

Shortage in human capacity: The human talent required to power FOSS projects is in short supply. In particular the following gaps are the greatest limiting factors for increasing the viability of FOSS:

- **Lack of technical and developer expertise**: Fundamental math and logical thinking skills

are often not well taught to the majority of learners, let alone strong grounding in software development and skills specific to software development tools and technologies. This represents a catch-22 of sorts; unable to grow and establish themselves with existing talent, FOSS projects are unable to create a community ethos in which new members could learn and gain the requisite skills.

- **Lack of African FOSS Evangelists**: Just as important as technical skills in the growth of African FOSS is the lack of passionate advocates to educate and advocate about the benefits and promise of FOSS. Those who do advocate on behalf of individual projects or the movement as a whole often find it hard to make a sustainable living.

- **Lack of FOSS Communities**: Shortages in human talent, connectivity and awareness add up to a lack of FOSS communities. It is hard for FOSS projects to find participants, and equally hard for individuals to learn about and join FOSS projects. The lack of FOSS communities also means a lack of models for others to draw from, adapt and otherwise draw inspiration.

- **Retention of developers**: The shortage of software developers in Africa contributes to another challenge: retaining developers who have gained marketable skills while working on FOSS projects. Financial exigencies often compel FOSS developers to jump to proprietary corporate projects where compensation and job security are substantially better.

Cultural barriers: In conjunction with other challenges enumerated are a number of issues faced on cultural levels. Some stem from institutional dynamics mentioned above, while others reflect norms in some or all parts of of African society. These include:

- **Lack of exposure at younger ages**: Due in large part to the lack of educational adoption, youngsters are not usually exposed to FOSS at early ages. Initial technology experiences are often shaped by use of proprietary tools.

- **Lack of reading culture**: Reading skills vary widely across Africa, and the lack of strong reading skills is a fundamental barrier to the self-education and DIY (Do It Yourself) ethic that often powers FOSS learning and uptake in other contexts. In addition, most existing FOSS documentation and resources are available in a small number of developed-world languages, meaning reading skills must be in those languages if FOSS knowledge is to be accessed.

- **Rote culture**: Educational and vocational models in Africa tend to focus on the acquisition of rote skills, as opposed to critical thinking skills. The latter are essential to participation in FOSS communities where self-determination and innovation are the lifeblood of successful projects.

- **Lack of a culture of volunteering**: The vast majority of FOSS communities obtain their labor resources from volunteers who are inspired and want to move projects forward. Volunteering is not the norm in many parts of Africa, and thus the notion of volunteering in

FOSS efforts is a foreign notion to many. FOSS projects in Africa have a hard time recruiting free labor and sustaining the corresponding community energy engendered by such contributions.

Resources and infrastructure: Central to the challenges faced by the FOSS movement in Africa are the resource and infrastructure bottlenecks that inhibit otherwise motivated and available community participants. These bottlenecks include:

- **Limited access to internet and learning resources**: The FOSS movement has blossomed in the internet era because human and information resources distributed across the goal have been able to connect seamless and serendipitously in the TCP/IP grid. Limited or expensive internet access in many parts of Africa has inhibited FOSS from permeating the continent and enjoying the ubiquity of access enjoyed in developed countries.

- **Shortage of computers and other hardware**: In parallel with connectivity barriers are fundamental hardware shortages. PCs are hard to come by in many parts of Africa, and what is available to those who might work on FOSS projects is often older and obsolete. Many PCs are available only in institutional contexts such as schools and offices, and are not available on a 24/7 basis. And even where personal computers are available, unstable electrical grids and other rugged conditions accelerate wear and tear on those devices and peripherals.

- **Difficulty organizing and communicating within FOSS communities**: The resource constraints listed above contribute to a fundamental gating factor in the growth of FOSS in Africa: members of geographically distributed projects are hard-pressed to stay in ongoing communication with one another, and thus have great difficulty planning, organizing and managing FOSS projects.

Market and competition dynamics: Even when open source products come to market, it is within a challenging landscape of competing offerings. In particular, the following factors make uptake of FOSS more difficult:

- **Proprietary software is easier to install and more accessible**: While FOSS continues to mature and diversify, both the perception and often the reality is that proprietary tools are easier to obtain and install.

- **Monopolistic give aways**: Proprietary vendors, seeing the threat that FOSS is posing in other markets, undertake aggressive give-aways to schools and government institutions at little or no cost to the recipients. This drives a ubiquity and pervasive presence of proprietary software in technology contexts, and makes it more difficult for FOSS offerings to get a fair look or equal consideration.

- **Software Piracy**: Even when vendor give-aways are not the problem, cultural norms that accept and encourage the piracy and mass reproduction of proprietary software blur the distinction between “free as in speech” and “free as in beer”. When all software is

effectively the latter, the benefits of FOSS are harder to convey, and the momentum towards continued use of proprietary tools is hard to offset.

Financial Sustainability: Based on all the foregoing factors and more, FOSS in Africa has rarely reached a point where it promises to be a financially sustaining enterprise.

- **Bootstrapping challenges**: Cost of FOSS development in the current context can tend be more expensive than commercial, off-the-shelf (COTS) software.

- **Access to capital**: Few practitioners or projects in the African context can afford to self-fund development of FOSS projects. Investors and venture capital are rarely available, so finding capital to underwrite startup, development and other projects costs is vexing. Risk aversion is high, and the perception is that FOSS projects come with inherently more risk than proprietary counterparts.

- **Difficult to calculate return on investment**: Those who might otherwise be inclined to invest in and underwrite FOSS efforts find it difficult to calculate return on their investment (ROI). Whereas proprietary software models involve well-defined revenue streams, FOSS ROI exists in the more intangible social contexts. And while FOSS tools can be superior to proprietary offerings, measuring the nature of that advantage is a challenge that remains vexing not just in Africa, but across the globe.

- **Lack of market for software**: As detailed in many contexts above, there is not yet strong market demand for FOSS offerings.

While all of these challenges remain extant, there is much exciting work being done in FOSS communities across Africa. The following sections detail the working of successful projects who participated at Good to Great FOSS.

3. FOSS Community Processes

An essential component of vibrant and sustainable open source projects is the community behind the code. Any organization or developer can release software under an open source license, but the relevance and lifespan of the project is usually determined by the participation of passionate users, developers and other stakeholders who take a role in using, testing, fixing, enhancing, and supporting the code base.

Open source community dynamics vary widely in the African context. This is due to a number of factors, many related to issues documented in the “challenges” section of this paper. Limited connectivity precludes the “always online” nature of open source communities collaborating in more developed contexts. With web access at a premium, online engagement is often limited to email, often precluding active participation in web-based communities such as SourceForge. African projects also face greater challenges in drawing attention to their work, and miss out on community engagement that would come with such exposure. And the limited number of available engineers in the region further compounds efforts to establish community.

That said, there are vibrant communities collaborating and developing code in Africa. And for these projects, the processes by which they govern, document, support and receive contributions from their communities is a fundamental indicator for both sustainability and success.

AVOIR Community Process

The AVOIR project describes their community process as lightweight. They draw a fundamental link between the architecture of product and the community process; a well-defined architecture makes governance easier because discussions, ideas and debates happen within that context.

They also make it a point to follow community process even when in same room, both to ensure consistency and to assure that those not present in the room enjoy the same access and transparency as they would as a result of purely virtual collaboration. There is an emphasis on well-defined guidelines at the process level, which in turn makes it easier for project members to interact with one another at the code level. An important lesson learned in the AVOIR community process is “keep it simple, don't have too many rules”.

Communication for the project is done primarily on mailing lists, which are relatively low volume. There are several mailing lists: one for the governing board, one for developers, and one for implementors who are installing and configuring the software in different environments. The developer list is active every day, with 50-60 messages/day from new students, proposal ideas, new business opportunities and other stuff.

The process for managing the project source code derives from the modular architecture of the platform. All code lives in 1 of 2 repositories, which archive the core platform and modules. AVOIR uses the CVS version control system. The community acquires new

developers through one of several avenues, but many register through AVOIR's GForge site.

A striking point of process and philosophy is that all developers are given commit rights, meaning they can check in code they write. The underlying belief is that it is easier to roll back bad code that might get checked in than it is to decide who should and who shouldn't have write access to the repositories. Over time, this hasn't presented any problems; developers don't generally make commits on others' code. The project encourages people to commit often--“as soon as it runs”--and before going home at the end of each day.

A critical value in the AVOIR community is the importance of bug tracking, which proves particularly important in a geographically diffuse community, with developers separate from users. The project needs to be able to track both bugs and feature requests, and follow set processes regarding feature additions. AVOIR uses Mantis. The current process is an important improvement over past models.

AVOIR also uses their bug tracking tool for project management. For paying customers, a project manager is hired, so it depends on the nature of where the development comes in.

OpenMRS Community Process

The OpenMRS Community is an unusual hybrid. In addition to the usual user and developers, there also community nodes per-installed-site. Most of the developers are also users of the system, or work for companies that use OpenMRS. In addition, there are multiple versions of the software extant, such as the version deployed for Partners in Health.

The community started out small, with 3 to 4 developers, as well as a “primary user” who provided great feedback and tested new releases. Things stayed that way for 2 years, and then Google Summer of Code brought in 10 new developers. It was a challenging process to get them integrated into the project, and up to speed on the development environment, but it catalyzed a maturation of both the project and the community process.

OpenMRS uses a wiki to store and maintain all the project documentation for the platform. Expanding the community required the project to revamp and redesign the front-facing page so visitors knew where to go; the redesign took a couple of months, but the net result was that the project evolved to support an external developer community.

The project also utilizes RSS as a communications tool. Feeds have been created to allow people to track changes and code commits, as well as posts to lists and forums, and posts to both the general and developer blogs. The feeds model has centralized the acquisition of information, and enabled both the core team and new developers to understand what is going on with the project.

For version control, anybody that asks to commit is given rights. The platform is modular, and new modules are welcomed. Core code contributions are reviewed before being applied, and when things break, changes are rolled back.

One important evolution of community communication involved the establishment of a separate mailing list for implementors. Developers exposed to routine implementations were getting bogged down. The implementation list works as a helpline. People can post questions and get answers, and the group learns each others' work patterns. This is helping the project to establish a bigger footprint in Africa

For project management, OpenMRS uses Trac, combined with the wiki. Trac manages tickets and features, and discussions take place in the bug tracking section. The project doesn't work on the basis of hard deadlines.

In terms of recruiting developers, they often start out as implementors. They need a feature, and the modular architecture allows them to build the need functionality and add it with minimal politics. The policy it to “take whoever comes”, and universities often yield connections. Promotion of OpenMRS is often done by word of mouth,and the yearly conference serves as a focal point. Project members attend other technology conferences and to present OpenMRS and solicit feedback, but they consciously refrain from “selling” the product.

Community process for other projects at Good to Great FOSS

Other projects at the event were not in a position to comment on their community process for a range of reasons. DrumNet is still in an early development phase, and have not yet distributed the code under a FOSS license. Tradenet has not made any decision to distribute any FOSS code. And Mifos did not participate in the community process discussion.

4. FOSS Development Processes and Development Environments

Open source projects are often characterized by the technology infrastructure on which they are built. Programming languages, application frameworks, database layers, and server technologies are some of the attributes used to differentiate and contrast different software offerings. Collectively, these technology components are referred to as the “application stack” for a project. In addition, the set of tools used in the course of software engineering—development environments, bug trackers, automation and testing tools, and documentation systems—are referred to as “development stacks”.

The projects at Good to Great FOSS spanned a range of technologies and approaches with both similarities and substantial differences in their stacks. The following section enumerates the essential technology traits of each participating project, including details about the respective application stacks as well as some of the development tools and processes used by each team.

OpenMRS

The OpenMRS application stack is based on the MVC (Model-View-Controller) pattern for software design. In OpenMRS, the view, or presentation layer, is based on JSP (Java Server Pages) containing HTML and Javascript. The model, or business layer, is written in the Java programming language, accessed through the reusable OpenMRS Java API. The database layer is based on the Hibernate technology to persist OpenMRS' Java objects to database tables. OpenMRS is served using the Tomcat application server, with the Apache Server acting as the front-end content web server; traffic is redirected by Apache to the Tomcat server.

OpenMRS has a modular architecture; modules are written to extend the functionality of the core OpenMRS code. The module architecture is custom for OpenMRS, but is very similar to Firefox and Eclipse extensions. Modules can be implemented to extend either the model or views of the application. The project has developed a "module repository" to aid people in locating and utilizing modules for their installation.

OpenMRS development stack uses a range of tools to support their development process:

- **Trac** is used for project management and bug tracking. The process is that anyone can self-create an account and then create a ticket.

- **Mediawiki** is used for all documentation about the project, and anyone can self-create an account and add/create/edit pages. The wiki also serves as the homepage of www.openmrs.org, so it is the entry point for developers and others new to the project. The wiki also serves as a hub for collaboration; a range of discussion take place on the wiki's “user” pages, such as what each user is currently focused on. As part of the process, discussion items are migrated to Trac tickets if feature requests emerge, or bugs are identified.

- **Subversion** is used for the source code repository and code version control. Subversion allows new code to be committed to the database, tracks changed files and their histories, and manages the unique number assigned to each “build” of the code. Subversion accounts are linked to Trac, and users are given permission to commit code based on several criteria. “Core team” developers have full commit access, whereas new developers that request accounts are only given the ability to commit to their module. New developers can submit small bug fixes to the alpha/trunk, but new features are developed in branches before being merged into the main trunk. Patches to address bug fixes and other code changes are attached to Trac tickets and submitted to Subversion by a "core" developer.

- **Eclipse** is the code development environment of choice; it is one of many options for Java Integrated Development Environments (IDE). Eclipse was selected because of available plugins for integration with Trac, Subversion, and reporting tasks. Mylar is the plugin used for integration between Trac and Eclipse. The IDE also supports integration with JUnit for testing.

- **ANT** is used to automate nightly builds and manage periodic releases.

- **JUnit** is used for testing the software, though test coverage is limited at the current time. Many OpenMRS implementations are running alpha code.

Project communication takes place in several primary channels. Two mailing lists, one for developers and one for implementers, are utilized for both notification and discussion. Real-time chatting is done using Internet Relay Chat (IRC) in the chat room #openmrs on irc.freenode.net. Discussion forums are used for focused asynchronous dialog on a range of topics, from general discussion to data models, the code base, usability and modules. Finally, weekly phone calls allow project members to be in direct dialog and discuss pending issues.

Management of the OpenMRS development process is done by categorizing and prioritizing three types of requests. “Project” requests, which are usually proposed by one of the developer, address significant changes to the overall platform, and usually involve large time requirements. “Feature” requests, which are more straightforward additions or enhancements, require less time than a project. Such features are usually engendered as posts to one of the forums or mailing lists, and are then converted to Trac tickets to formalize the request. Bugs, which entail small fixes to current code features, can begin as a post to a forum or list, or may be entered directly into a Trac ticket.

AVOIR/Chisimba

The heart of Chisimba application architecture is the modules. In the Chisimba application stack, each module has its own model, view, and controller objects; the application architecture can be envisioned as building blocks (See attached image). In addition, Chisimba modules are completely “skinnable”, meaning their appearance can be customized on a number of levels. Templates for content and layout can be specified, and each element has IDs and classes for each CSS (Cascading Style Sheet) customization. Version numbers are

managed by developers manually. In addition, the generalization of the view layer means that Chisimba applications can run on any device, not necessarily just web.

The Chisimba development stack includes the following tools:

- **Eclipse** is the IDE, in part because it allows for plugins for code completion, integration. It also supports built-in CVS commands to allow committing from within the IDE.

- **CVS** is used for code version control. It offers online tools for viewing what is happening on different branches/modules. In addition, it can be accessed using the command line from within Eclipse.

- **ANT** is used to automate the build process, for nightly builds and monthly releases

- **PHPUnitTest** is used to support the testing model for the Chisimba core, but is not required for modules.

- **Mantis** is used for tracking bugs, where they can be added, tracked, committed, and commented on. Statistics are generated per-developer on bug counts, severity of bugs and bug fixes.

- **Ohloh** is used to publicly show statistics and information about Chisimba, information which is automatically generated by scripts on Ohloh.

In terms of documentation, the Chisimba project has found that community members can't stay online long enough to edit the wiki, so documentation is done by editing offline and then committing documents to CVS. In addition, the “master documentation document” is also stored in CVS.

Communication is done primarily on mailing lists. The developer mailing list handles questions from developers about access and development, as well as posts from developers about new updates to modules. The implementers list is used by those who are deploying Chisimba applications.

DrumNet

The DrumNet Application stack is Java-based. Tomcat is used as the application server, and Spring is used as the application framework. Database components are managed by Hibernate, and JSP is utilized for user presentation. In addition, because communication with cell phones is a fundamental part of the DrumNet application, there is an SMS Gateway with a generic communication module,where messages come in through a mail server, and out through either HTTP or SMS.

The DrumNet development stack includes the following tools:

- **NetBeans** is the IDE, with plugins that support code completion and integration .
- **ANT** is used to automate the build process, with periodic builds and on-demand releases.
- **JUnit** is used for testing the core code.

Project documentation falls into two categories. User documentation is authored in OpenOffice and distributed PDF format. Code documentation is done using the Java-standard JavaDoc.

DrumNet code is not currently under version control.

Bugs not currently tracked in an automated fashion, but BugGenie is the anticipated tool when bug tracking is undertaken.

Project communications are done using a single developer mailing list.

Project Development Summary

The following tables summarizes the tools, languages, and other attributes of the projects described in this paper.

Aspect	OpenMRS	Chisimba	DrumNet
Wiki (document sharing)	Mediawiki used extensively	Use OpenOffice docs in CVS to share knowledge	OpenOffice documents, via email and server
Version Control	Subversion	CVS	n/a
Mailing lists	2 main lists, 1 for developers and 1 for implementors	2 main lists, 1 for developers and 1 for integrators	1 developer mailing list
Online Forums	Yes, used occasionally	Yes, rarely used	n/a
Bug tracking	Trac, for OpenMRS-specific problems, individual implementations have their own bug tracking, etc	Mantis, for Chisimba specific problems. Installations have separate locations for tracking issues.	BugGenie
IDE	Eclipse	Eclipse	NetBeans

Build automation	ANT	ANT	ANT
Server	Apache/tomcat or just Tomcat	Apache or LiteHTTP	Tomcat
Language	Java/JSP	PHP	Java/JSP
Testing	JUnit for both core and module, not required, not always done	Core is tested with PHPUnitTest (required for core, not required for modules).	JUnit
Usability testing	Working on establishing usability processes	Yes, at one point had a lab for usability testing. No progress recently (lost funding).	No
Load testing	No	Yes	No

5. FOSS Licensing

There was limited discussion of licensing at Good to Great FOSS. Licensing for FOSS projects in Africa varies, and appears to fall into the same general categories of license preferences and variations as the larger FOSS community.

To summarize those categories:

- The GNU General Public License (GPL) is the "original" FOSS license, setting forth the meaning of "Free Software" in legal terms. In lay person's terms, the GPL is defined by the "Four Software Freedoms":[1]

  - The freedom to run the program, for any purpose.
  - The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
  - The freedom to redistribute copies so you can help your neighbor.
  - The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

  At the core of the GPL is a "share-alike" ethic that specifies that one must redistribute GPL-licensed code under the exact same terms as it was obtained. In this way, GPL-licensed code can never become "less free".

- A larger class of "Open Source" licenses have been certified by the Open Source Initiative, and are referred to as "OSI-approved" licenses[2]. These licenses contain a range of terms that distinguish them from the meaning and intent of the GPL, but a simple generalization can be made that the primary difference often lies in the share-alike terms, or lack thereof. Licenses such as the the FreeBSD (used for the FreeBSD operating system) and the Mozilla Public License (MPL, used for Firefox and other software distributed by the Mozilla Corporation) allow code distributed under those licenses to be linked and distributed with proprietary code. The nuances of what is and is not allowed by various OSI-approved licenses drive a vast trove of online debate and discourse.

The difference between the "free" software licensing of the GPL and the "open source" licensing of the larger class of OSI-approved licenses lies at the root of much friction in the FOSS community. The Free Software Foundation and other GPL-focused stakeholders believe that licenses other than the GPL subvert the essential "freedoms" of the GPL and the larger vision for a free software future, while proponents of other OSI-approved licenses feel the strict and unyielding nature of the GPL of limits their flexibility, innovation, profit, and other opportunities.

The situation has been further complicated with the publication of GPL Version 3 (GPLv3). This thoroughly overhauled version of the GPL license was published to address coverage gaps in GPL Version 2 (GPLv2), such as defining the status of GPL code adapted and

1 http://www.gnu.org/philosophy/free-sw.html
2 http://www.opensource.org/licenses

modified for “hosted” web applications which is never officially “distributed”. GPLv3 also contains a new category of restrictions relating to digital rights management (DRM), and states that GPLv3 code can not be employed in the creation of software that supports DRM. High-profile projects based on GPLv2, such as the Linux operating system kernel, have refused to adopt GPLv3. And the fact that GPLv3 is not backwards compatible with GPLv2 has only exacerbated an already complex licensing landscape.

A critical implication of balkanized FOSS licensing regimes is the issue of license interoperability. GPL code simply can not be combined or distributed with code licensed under non-GPL licenses; there is no license for the resulting aggregate which would satisfy the terms of both inherited licenses. In general, it is difficult if not impossible to combine code distributed under different FOSS licenses, though some vendors such as MySQL provide “FLOSS License Exceptions”[3] which mitigate interoperability constraints. In any case, FOSS developers must choose carefully in deciding under which license they should distribute their code.

One topic of discussion at Good to Great FOSS was the advisability of “dual licensing” approaches. In such frameworks, different licenses are granted for different uses of the source code. Dual licenses can both mitigate license interoperability issues (such as GPLv2 vs. GPLv3), while also providing the foundation for FOSS business models where commercial use of the code generates revenue.

In such dual-licensing scenarios, different terms are granted based on how the resulting code will be distributed. For new code which will be distributed under GPL or open source licenses, a corresponding GPL or open source license is granted. But for commercial vendors who distribute the licensed code with their proprietary products, and do not license and distribute their own source code under the GPL, a commercial license is granted, and usually associated with licensing fees or other revenue sharing. The MySQL database platform has a good example of dual licensing on their license page at http://www.mysql.com/about/legal/licensing/.

The projects at Good to Great FOSS operate under the following licensing frameworks:

- **Chisimba** is distributed under GPL version 2, which is described at http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

- **DrumNet** has not yet decided on the licensing framework for the project.

- **OpenMRS** uses the OpenMRS License, which is a variant of the Mozilla Public License (MPL), and is described at http://openmrs.org/wiki/License. A primary reason for not using MPL was the need to include expanded warranty and liability disclaimers specific to healthcare-related technology.

- **Mifos** is distributed under the Apache License, Version 2.0, which is described at http://www.apache.org/licenses/LICENSE-2.0. The Mifos licensing page at

3 http://www.mysql.com/about/legal/licensing/foss-exception.html

http://www.mifos.org/developers/wiki/MifosLicensing is an example of best practices in licensing, providing template headers for all types of source code files.

- **Tradenet** code is not currently distributed under any license.

While best practices for FOSS licensing are hard to generalize, the following assertions are safe to make:

- The GPL still represents the highest ideals of FOSS licensing, and should be considered in any licensing decisions. However factors including dependent code licenses, partnering agreements, target markets, business models and institutional constraints may prevent GPL from being the best choice. On the other hand, GPL licensing provides a "moral high ground" in FOSS distribution, and saves projects from having to explain and defend why they opted for "less free" licensing.

- Dual or multiple license approaches should also be considered when looking to increase uptake and adoption of FOSS projects. While such licensing models have the effect of "watering down" pure GPL offerings, they provide flexibility to those who otherwise might not be able to incorporate the available code. FLOSS License Exceptions such as those mentioned above also alleviate code interoperability blockages.

- In any case, creating a new FOSS license should only be considered as a very last resort. While unique institutional and legal requirements such as those associated with the OpenMRS project can mandate a specialized license, new licenses only clutter the landscape. All efforts should be made to not only use an existing license, but to use one which is in broad distribution, so as to maximize the reusability of the licensed code.

6. FOSS Business Models and Sustainability

The topic of business models and sustainability was an area of keen interest for many of the participants at Good to Great FOSS. Projects already distributing code under FOSS licenses were looking for insight on ways in which to make their projects more self-sufficient. Projects that had not yet committed to FOSS licensing were looking for justification and motivation to consider FOSS distribution models.

A range of business models were considered in the discussion:

- **Support-based revenue model**: Firms like Red Hat distribute their code as FOSS, but then enter into support contracts with customers. In this model, the market differentiator is quality and ubiquity of the support. Customizations and integrations with other systems can also generate additional revenue.

- **Consulting models**: In this model, business practices are built around FOSS applications which require customization and substantial experience in order to deploy. Consultants generate revenue based on the FOSS code by selling installation and development services to clients, as well as support and maintenance. Perhaps the most pervasive example of this approach are the FOSS content management systems (CMS) used in web site publishing. Applications like Drupal, Joomla! and Plone have rich ecologies of consultants who install, customize and support individual instances of the software. A key difference in this type of model is that the consultants aren't necessarily involved in developing and maintaining the core application code.

- **Data-centric model**: In this model the code is FOSS, but the application depends on non-open or difficult-to-acquire data sets. An example of this from the US software market is the DemocracyInAction platform, which is licensed under the Affero GPL. The platform is used for online advocacy targeted at media and congressional representatives, and the databases of legislative and media contacts are not distributed with the platform.

- **Hosting services**: Some businesses are built around value-added hosting using FOSS platforms: the value proposition is in the quality of hosting, excellent up-time, support, maintenance. In these scenarios, customers know they can migrate at any time, but in the meantime can delegate cumbersome responsibilities that require full-time attention (such as machine maintenance and security patching) to vendors with dedicated facilities. An example of this business model is www.opensourcehost.com, which provides Linux-based hosting of web applications for content management, blogging, site management and other services.

- **Hardware-based business models**: While cast in a different context than the projects at Good to Great FOSS, business models based on selling hardware running FOSS software are becoming more common. Perhaps the best known examples in this regard are the wireless network routers based on FOSS. These routers run stripped-down versions of the Linux operating system customized for networking. Linux-based cellphones are also making inroads in markets such as Japan. In these models, vendors make money on

hardware sales and/or subscriptions where FOSS is an ingredient in the product offering.

- **Dual licensing models**: As described in the FOSS Licensing section of this document, projects such as MySQL distribute their code under multiple licenses. The product is distributed as free GPL software to other GPL projects, but licensed with associated revenue to vendors who want to distribute the code in proprietary products.

A general theme in the business models dialog centered on "generic" versus unique software functionality: FOSS business models may tend to work better with applications where unique intellectual property and/or unique business strategy and logic are not embedded in the application. This is because it is clearly difficult to maintain a market advantage when the differentiating product attributes are available for any competing party to study and emulate. This is borne out in product sectors such as web content management systems; platforms such as Drupal, Joomla! and Plone offer very similar features, and compete on core architectural traits including module capabilities, underlying technologies, and usability. On the other hand, a project like the Asterisk VOIP (Voice over Internet Protocols) platform serves as a counter-example; Asterisk possesses a number of unique features and capabilities embedded in the code, but maintains its market edge by consistently out-innovating the competition.

The question was raised as to how well each of the discussed business models might work in the African context. The consensus was that making money in Africa is hard in general. Since the majority of IT expenditures in developing countries are made by government and large telecoms, the implication was that each model should be vetted for viability in those markets. But it was also pointed out that trying to have government understand the difference between FOSS and proprietary offerings is quite difficult.

Several challenges in successful FOSS business models for Africa were enumerated:

- Support for FOSS offerings is still costly in Africa; the tipping point where demand drives down prices has not yet come. Microsoft, for example, is lower quality, but it's more cost effective.
- There is a pervasive issue of “certific-itis”; the Africa markets tend to prioritize vendor certification over unique and strategic technical knowledge and tools. Microsoft has a large practice in certifying Microsoft expertise, and FOSS products like Linux lag in this regard on the continent.
- Available cash to bootstrap FOSS ventures is also less available to private enterprise; NGO's can use grants and other sources in ways that companies might not.

As an exercise, participants in the discussion decided to build out a scenario using the DrumNet project and talk through business models and sustainability. While session time constraints did not allow the scenario discussion to fully play out, many interesting ideas were exchanged in the dialog.

The scenario was laid out as follows:

- DrumNet wants to connect four actors: buyer, farmer/producer, banks, and input suppliers (fertilizer, etc). They want to have a seamless interface between these suppliers in a Market / Finance / Information (MFI) model.

- They are developing a platform to use phones and the internet to connect these actors. In theory this is a win-win: banks loan and make money, farmers get support and sales, buyers get better organized local markets, suppliers sell product.

- The particular use case for the discussion was as follows: Farmers join the network, indicating the number of sunflower acres they intend to plant. SMS and email are sent to suppliers, telling them what quantity of supplies to ship to each farmer Suppliers send goods to a stockist, and farmers pick up goods. Farmers then report on crop progress and yields, which are then sold through the network at harvest.

The overall opportunity is that agricultural markets are poorly served, and a system like the one described above can introduce efficiencies and transparency into the market place. The struggle is in defining the business model: is it transactional, is it ASP (Application Service Provider), or is it training and support?

A matrix was then developed to consider how to apply various business models for the scenario based on a FOSS platform:

Model	DrumNet Value-Add	Revenue Sources	Business Risks	Advantages
Single hosted Instance of the platform	Market and Actor knowledge and relationships, Trust Factor, System knowledge	Transaction & interest fees, Certifications, Training & doc, Access fees/sponsorships	Cloned by competition	Goodwill, Trust, leadership by example, 3rd party contributions
Franchised instances of platform (contractual)	Markets knowledge, Strategic consulting, Organization development	Consulting, % of volume, Support, Customization	Contract non-compliance via hacked code or fake data	If successful, model is well positioned to scale.
ASP model: Sell hosted application to players who facilitate markets	Content/data integration, scale in markets (a/k/a critical mass)	Hosting fees, meta-consulting, partnering, advertising	Cloned by competition	If successful, model is well positioned to scale.
Data-centric: Differentiate in non-open data used with platform	Unique data, “Secret sauce” that is difficult to replicate.	Selling data, consulting on data	Reverse engineering	Harder to clone, new aftermarkets possible with innovative use of data.

All participants agreed it would be interesting to see where DrumNet actually takes their business model, and what licensing they decide upon to support their approach.

7. Best Practices for FOSS in Africa

The following represent best practices for FOSS development in the African context, as shared by participants at Good to Great FOSS. This enumeration focuses primarily on practices critical or unique to the African context, or which vary from conventions in developed country contexts.

Community Development: Central to the long-term success of any FOSS project is the ability to grow the involved community by recruiting and retaining new developers and other contributors. The following were called out as critical to successful community development:

- **Active engagement**: Just making project source code available will rarely draw in developers; there are too many interesting projects extant for code alone to be the primary attraction. Project principals must actively engage interested parties and find ways to honor their interest. In addition, contributions and other efforts should be acknowledged in a fashion that is visible to the project community.

- **Transparent process**: A differentiating factor in FOSS projects is the degree to which project decisions and management are done in a trackable and publicly visible fashion. Having well-defined processes for decision making, planning, and project management encourages new participation, and empowers all project members to understand and contribute to project evolution. An important corollary for processes that guide distributed collaboration is to follow those same processes even when meeting in the same room.

- **Well-defined entry points**: Projects need to make sure their online presence is friendly and inviting to the uninitiated. A well-designed project home page that summarizes the project, provides latest news, and conveys a sense of community activity is essential to recruiting new community members. In addition, for those wanting to get more involved, information about how to do so is critical in order to engage developers and other contributors in their moment of interest.

- **Accessible Documentation**: While documentation is the bane of many programmers' duties, projects with good documentation and documentation processes have a better chance of engaging new community members. The ability to browse project architecture, features, and processes can help in convincing new members to get involved. An important decision projects in the African context need to make involves online versus offline document management processes. While wikis provide a powerful, collaborative way to keep shared knowledge current, they are more bandwidth intensive. Projects like Chisimba opt to follow an offline documentation model, where OpenOffice is used to author and maintain documents, which are then checked in to the version control system along with source code.

- **Offering intangibles**: While it is difficult for new or small projects to do so, there are “intangible” benefits that can draw in developers and new community members. Two in particular are worth noting: “fame” and “buzz”. FOSS developers are often motivated

by a desire to grow their reputation in the FOSS world; projects which can provide developers with high visibility or public acknowledgment of their contributions have greater appeal. Projects with substantial and growing user bases have the additional ability to offer developers a chance to see their work enjoyed by a larger audience. In addition, projects focused on “hot” or “buzz” markets or emerging technologies are more likely to draw in new community members. An example of “buzz” technology in the African context is the integration of internet and cell phones; such a bridge is at the cutting edge of communication in Africa, offering both compelling technological and social benefits in bridging digital communication divides.

Communication and collaboration: Staying in contact is often a challenge for FOSS projects in Africa, owing to limited internet access. A multi-channel approach was recommended at Good to Great FOSS, in part to account for varying degrees of connectivity by different project participants:

- **Email**: Because it is more bandwidth-friendly than the web, email should usually be the core communications channel for important project discussions and notifications. Email messages should be distributed via mailing lists (as opposed to individual addresses and “cc” lists), both to give recipients control of how messages are received as well as so that archives are generated which then allow new project members to browse and review project history. A common convention for Good to Great FOSS participants was to manage a pair of mailing lists, one for developers and one for those implementing or deploying the actual software.

- **Internet Relay Chat (IRC)**: A valuable synchronous counterpart to asynchronous email is IRC. IRC allows for rapid response to questions, as well as quick turnaround on decision making. While not accessible to those for whom a full-time internet connection is not possible, IRC is low-bandwith and uses minimal resources on the client machine. The host irc.freenode.net is a popular venue for establishing and maintaining an IRC channel for many FOSS communities.

- **Discussion forums**: Web-based discussion boards can be used for focused asynchronous dialog on a range of topics, from general discussion to data models, the code base, usability and modules. But owing to the higher bandwidth needs of HTTP-based discussion pages, critical messages and conversations should be handled or mirrored on primary email lists.

- **Rich Site Syndication (RSS)**: Projects like OpenMRS use RSS to provide feeds documenting changes in all aspects of the project, from wiki page updates to bug fixes. RSS is a bandwidth-friendly and efficient way to let project members track project goings-on without necessarily spending lots of time loading web pages.

- **Telephone**: Where feasible, weekly or regularly scheduled phone calls allow project members to be in direct dialog and discuss pending issues, and greatly enhance community energy. Where bandwidth and usage agreements permit, VOIP (Voice over Internet Protocols) tools such as Gizmo and Skype can enable such conference calls

to be done at no charge to participating parties.

Outreach and marketing: In parallel with active community engagement, FOSS projects must make themselves to visible to the larger FOSS community. Projects should register and be discoverable on several key sites:

- **Sourceforge.net**: The de facto standard for hosting FOSS projects, SourceForge is an import online venue in which to establish a project identity, even if code is archived elsewhere.

- **Freshmeat.net**: The web's largest inventory of FOSS and Unix/Linux software.

- **Ohloh.net**: Ohloh is a new entry in the global FOSS landscape, but it offers compelling value. Ohloh tracks very specific project details, including lines of code, number of developers, and rate of development. Projects registered on Ohloh can convey their level of activity and community growth.

Projects should also make sure entries on these sites are well maintained, and convey the sense of an active and vibrant project community to those who encounter them.

Code architecture and Management: At the heart of any FOSS project is the source code itself. Decisions and practices established early in the lifecycle of a code base can influence the sustainability, relevance and attractiveness of a FOSS project. The following are guidelines which were discussed at Good to Great FOSS:

- **Manage code through a version control system**: The use of Subversion or CVS to archive and manage source code is an essential trait of successful FOSS projects. Such systems control who has permission to edit which files at any given time, track version history for each file, and support naming and marking to denote which version of each file was used in a given build or release. These repositories also make possible truly distributed development by coordinating changes from various contributors, and supporting “rollbacks” when code checked in by one developer breaks or inhibits functionality in other parts of the code.

- **Leverage an established stack**: Build new projects on software stacks which themselves have strong communities and mature, stable code bases. This approach has a range of benefits; documentation and support are in greater supply, maintenance of such components happens in a consistent and timely fashion, and project developers are able to focus more exclusively the code they are actually developing, rather than what they're building upon.

- **Use design patterns**: The use of design patterns to create well-architected, well-factored code is essential to maintenance and scaling of code bases. In particular, the MVC pattern (Model-View-Controller) enforces code discipline that separates “business logic”, which models the problem space of the application, from “presentation logic” that renders the user interface and interacts with the user. This

separation, mediated by “controller” classes, insulates the model code from evolutions in front-end technology and conventions. An additional benefit of such design patterns is that business logic can be repurposed and utilized on a range of client devices, from traditional computers to cell phones and other handhelds, not to mention gizmos which have yet to be invented.

- **Employ modular architecture**: Code base design informs the architecture of participation for any FOSS project. Projects with monolithic code bases run into scaling issues as more developers become involved, because incremental changes made during new feature development need to be integrated into the central code base, but are often out of sync with changes being made by other developers. Modular architecture, where a smaller central “core” code base is extended in functionality by the addition of modules which “plug in” to the core, is a more sustainable and scalable coding approach. Not only can development happen in parallel on a number of modules without adversely impacting other module development, but platforms implemented in this way can be deployed in different configurations for different needs and varying resource constraints.

- **Release early and often**: Developers should resist the instinct to hold out for “big” releases, and instead get code in the hands of users and other developers as quickly as possible. This establishes tighter feedback loops, and keep the project more in touch with its ultimate audiences.

Tools: While a complete enumeration of development tool sets required in FOSS projects is beyond the scope of this paper, several specific tools bear mentioning for their essential nature in properly supporting and automating FOSS projects:

- **Project management**: Projects need a central tool for managing tasks, assignments, and timelines. While FOSS project management processes vary widely, the need for effective and usable project management is without dispute. In addition, such tools greatly aid in transparent project management, providing a dashboard of “who's doing what” and allowing newcomers to both comprehend project dynamics and more easily get involved.

- **Bug tracking**: Sometimes lumped in as part of project management, and other times deployed as a stand-alone application, bug tracking tools are the immune system of FOSS projects, tracking what's broken and what has priority in pending fixes. Tools like Trac or Mantis span the features of both bug tracking and project management.

- **Build automation**: A critical indicator of a code base's ability to scale is its ability to be built from scratch on a “clean” machine. Build automation tools such as ANT automate the process of “pulling” source code from a central repository and assembling the requisite components to generate an executable version of the code. In addition, such automation tools allow builds to be done on a daily or regular basis, which enables developers to quickly identify when new code has “broken the build” and requires mitigation.

8. Recommendations for a better FOSS future in Africa

Good to Great FOSS participants spent the final session at the event brainstorming how to better support and grow the FOSS movement across Africa. The following represents a “blue sky” set of ideas, where no constraints were placed on thinking in terms of costs or other limiting factors.

Advocacy and awareness raising: Several ideas were put forth with the goal of driving greater understanding of both the potential and the current benefits of FOSS:

- **Leadership and champion development**: FOSS projects and developers currently must serve as their own advocates and evangelists. Intentional efforts to recruit and train champions able to articulate the benefits of FOSS to a range of sectors would distribute some of that work away from those already trying to deliver tools, and diversify the ecology of FOSS stakeholders.

- **Development of evangelism resources**: Advocacy would be greatly bolstered by the availability of quality materials to support outreach work. These assets would include a list of the benefits of FOSS software, talking points for FOSS-vs-Proprietary discussions, case studies, how-to's for evangelists, and lists of relevant resources such as mailing lists and online communities.

- **Success stories of private FOSS initiatives**: Examples and descriptions of successful FOSS projects, both African and western, especially those not delivered under government or educational auspices would be a very useful resource to those advocating about the sustainability and potential of FOSS undertakings.

Community Building: FOSS projects in Africa often lack peer awareness, and according to practitioners, there is not a strong peer sharing ethic. The following recommendations by participants at Good to Great FOSS seek to address these issues:

- **African virtual FOSS community**: An online community space designed to connect FOSS practitioners in the African context would fill a large gap in connecting projects that otherwise have no common place to meet and collaborate. First, by listing existing communities and organizations and their respective areas of focus, such a hub could better connect what is already being done while providing any additional mailing lists or wikis which were needed. Second, by providing documentation and best practices on collaboration models, hiring processes, licensing, community oversight and software development, the site could lower barriers to entry for those wishing to get started or scale their FOSS projects. Third, such a community would provide a a better forum for developers to exchange ideas and solutions. While such a community would not be readily reachable by all FOSS practitioners due to connectivity issues, it would provide a starting point for a more connected FOSS community across Africa.

- **FOSS project mapping**: It is still very difficult to know what FOSS projects are extant in Africa. A mapping of FOSS projects working in the African context would offer an

invaluable inventory of who's doing what, and offer a range of stakeholders insight into where they might plug in and offer support, as well as avoid duplicating existing efforts.

- **Events and workshops**: While FOSS communities typically operate in virtual and distributed fashion, there is no doubt that in-person convenings are an invaluable source of knowledge gain and community strengthening. Ideas discussed include an annual FOSS fair, tied in with awards and a job fair in each country, to bring together developers and other FOSS advocates in a environment of capacity building and social networking.

Institutional support: FOSS projects are often challenged from the outset by a lack of institutional support. The lack of both financial and tactical resources hinders both success and growth. Ideas proposed to counter this dynamic include:

- **Venture fund for FOSS in Africa**: The establishment of a pool of venture funding for African FOSS projects would enable promising startups to seek seed capital. In addition, the process of seeking such funding would force those projects to analyze their business environment and sustainability much earlier in the development cycle than is normally done.

- **Government strategy and policy**: Any success at bringing governments into the fold as proponents could have far-reaching benefits for FOSS in Africa. But lobbying is challenging, and compelling cost/benefit analysis is hard to come by. In addition, influence peddling and corruption are ongoing concerns; proprietary vendors have resources and motivation to manipulate policy in their favor.

- **Infrastructure**: As countries and regional governments continue efforts to better connect their constituents, the increased connectivity will allow more of the ubiquitous collaboration that has characterized successful FOSS projects all over the globe. In addition, efforts to get PC's into more homes will increase the number of potential participants in the FOSS ecology.

Enhanced educational resources: Increased opportunities for both young learners and adults to encounter and understand FOSS tools and concepts are central to any growth plan. Two specific ideas were considered during the course of the Good to Great FOSS brainstorm:

- **FOSS-based learning curricula**: While there are already noteworthy examples of FOSS in African schools, the vast majority of curricula are still devoid of FOSS elements. Efforts to better integrate FOSS into elementary and high school learning programs would greatly increase the potential for FOSS to thrive in the African context.

- **Technology centers with FOSS programming**: Providing telecentres and other technology venues with FOSS tools and training materials could great learning gateways for adults to gain FOSS skills and knowledge.

9. Conclusion

The promise of FOSS in Africa is great, and the number of successful projects and thriving communities is ever increasing. At the same time, unique challenges faced in the African context make establishing, scaling and sustaining FOSS projects harder than in more developed contexts.

Good to Great FOSS 2007 served to capture a snapshot of the state of a diverse set of African FOSS projects. It is hoped that the details captured in this paper can be used as a benchmark against which future project and community updates can be compared.

Those wanting to undertake FOSS projects in the African context should consider the learnings shared by projects at Good to Great FOSS. At the highest level, these can be summarized as follows:

- Be mindful of the challenges face by FOSS projects in Africa. Design processes and community models appropriate for the environment, and be fault-tolerant to factors such as poor connectivity and developer attrition.

- Establish transparent and open community processes that encourage new participation and sustain environments of trust and accountability. Follow processes consistently, and make it easy and engaging for new members to get involved.

- Follow industry standard development processes built on robust and mature stacks and components. Limit risk by reusing what already works, and let successful FOSS projects and existing models inform project decisions.

- Adopt Best Practices in community development, collaboration, outreach, code architecture and tool selection to maximize indicators for success.

The good will and passion in the African FOSS community know no bound. As connectivity and other resource barriers continue to be surmounted, the great potential reflected in the current ecology of African FOSS projects will be realized and replicated across the continent.

Appendix I – Good to Great FOSS Agenda

The following represents the final agenda at Good to Great FOSS.

For additional background on each session, as well as session notes for bulleted session titles below, see the proceedings recorded at http://wiki.goodtogreatfoss.org/index.php/Event_Agenda.

Wednesday 24 October

9:00 **Opening Circle**

The event started with introductions, agenda overview, participant guidelines and reflections on the days ahead.

10:00 **Spectrograms**

This highly interactive session encouraged everyone to get their voice in the mix. Participants were invited to react to "controversial" statements by positioning themselves along a line on the floor that spanned the room and ranged from "strongly agree" to "strongly disagree", and to state how they felt on a range of issues relevant to open source software development in the African context.

The following spectrograms were explored, and participants were invited to interpret them any way they wanted to:

- Open source is *the* way to go in Africa
- Doing open source in Africa will always be too hard
- Open data standards and open programming interfaces are more important than open source

10:45 **Coffee break**

11:00 **Agenda Discussion Breakouts**

Participants broke into small groups to discuss their goals for the gathering, and how they wanted like the sessions to focus on their interests, questions, and needs.

12:30 **Lunch**

14:00 **Afternoon Sessions 1**

Three discussions were held in parallel, and participants elected individually which one to attend:

- Challenges of FOSS in Africa
- Collaboration Success Stories and Challenges
- Intro to FOSS

15:15 **Coffee break**

15:30 **Afternoon Sessions 2**

- FOSS Community Models and Processes

16:45 **Closing Circle**

Thursday 25 October

9:00 **Opening Circle**

9:30 **Morning Sessions 1**

- FOSS Business Models and Sustainability
- FOSS Development Process and Toolsets

10:45 **Coffee break**

11:00 **Morning Sessions 2**

- Distributed Development

12:30 **Lunch**

14:00 **SpeedGeeking**

15:15 **Coffee break**

15:30 **Afternoon Sessions 1**

- FOSS Requirements Gathering

4:45 **Closing Circle**

Friday 26 October

9:00 **Opening Circle**

9:30 **Morning Sessions 1**

- FOSS Licensing
- FOSS, Mobiles, and More

10:45 **Coffee break**

11:00 **Morning Sessions 2**

- Mobile Phones

12:30 **Lunch**

14:00 **Afternoon Sessions 1**

- Ideas to take promote usage and awareness of FOSS In Africa

15:15 **Coffee break**

15:30 **Closing Sessions**

Participants used the last slot on Friday to reflect on what had been accomplished and discuss where things went next for each project as they left the event.

16:45 **Closing Circle**

Appendix II – Good to Great FOSS Participants

Name of Participant	Institution and URL
Allen Gunn	Aspiration, www.aspirationtech.org
Anjna Shah	PRIDE Africa, www.drumnet.org
Ben Wolfe	OpenMRS, www.openmrs.org
Bishar Mohamed Duble	University of Nairobi, www.uonbi.ac.ke
Chris Seebregts	University of KwaZulu-Natal South Africa, www.mrc.ac.za
Christopher Kasangaki	MOSSFA Project, www.mossfa.net
Daniel Orwa	University of Nairobi, www.uonbi.ac.ke
Derek Keats	University of Western Cape, www.owc.ac.za
Edith Adera	International Development Research Centre (IDRC), idrc.or.ke
Enver Ravat	University of Western Cape, monasa.co.za
Gladys Githaiga	International Development Research Centre (IDRC), idrc.or.ke
Isaac Okeyo	Jomo Kenyatta University of Agriculture and Technology, www.jkuat.ac.ke
Jonathan Campaigne	PRIDE Africa, www.prideafrica.com
John Wainaina	Jomo Kenyatta University of Agriculture and Technology, KISE.jkuat.ac.ke
Joseph Gatheru	Jomo Kenyatta University of Agriculture and Technology, KISE.jkuat.ac.ke
Joseph Mugoma Okomba	VERVE, www.verveko.com
Justin Miranda	Partners in Health, www.pih.org
Luke Ouko	VERVE K. O. Ltd, www.verveko.com
Mark Davies	BUSYLAB, www.busylab.com
Mark Steudel	PRIDE AFRICA, www.prideafrica.com
Nancy Ndung'u	International Development Research Centre (IDRC), idrc.or.ke
Robert Njoya Kinuthia	PRIDE Africa, www.prideafrica.com

Appendix III - Good to Great FOSS Evaluation Form Results

Outcomes

1. **Did this event change your thinking about Free and Open Source Software Development in Africa? If so, how? If not, can you elaborate?**
   - Yes, the capacity for OSS development in Africa is there, what lacks is the drive and motivation
   - Not really, but it gave me insights into how we can create synergy among different initiatives
   - Yes, opportunity to work with commercial entities
   - Yes. Collaboration may not be easy/forthcoming. Mixed models allow commercial involvement in FOSS strategies
   - Yes, by learning that FOSS does not mean you can not live out of it; you can gain revenue through FOSS
   - Yes, I see a more developed open source community/movement that I previously thought
   - Yes, I am convinced that FOSS is the way to go
   - Yes, stronger sense of the role of OSS in business
   - Yes it did. I now know that FOSS is possible in Africa
   - Yes, definitely. I think OSS may be perceived differently in different contexts. Getting to understand some of the concerns that plague African developers and organizations
   - Yes, definitely. Business models was a huge eye opener
   - Yes, I got a deeper understanding of the power within FOSS to create my own solution and to understand what business model options are available and their implications
   - Yes. It helped me appreciate FOSS as less of a risk than I imagined
   - Yes. I have now realized there are other people involved
   - It did. I have gained confidence in the idea of community of FOSS developers
   - Yes. I have come to understand that it is possible to do business with OSS and how to do it
   - It expanded my thinking definitely + encouraged me to go and explore how we can interpret open source into Drum Net
   - Yes. It gave me a more concrete understanding of where it is applicable and not applicable
   - Yes. I realized that there is enough interest from so many different spheres that FOSS stands a chance of success.

2. **What were the two or three most useful things for your about the event?**
   - Learning environment, interaction environment and how great minds think alike
   - Finding synergy; learning some new application domains; the facilitation approach
   - FOSS business models; facilitation techniques
   - Models, support network
   - Licensing models for FOSS; business models for FOSS
   - Several takeaways for charges with OpenMRS; several avenues for recruiting devs. With Africa
   - Speed geeking – wish it was slow geeking as it was very interesting learning about existing projects, licensing
   - Meeting of minds and the generation of new perspectives
   - Business models for FOSS; FOSS culture change in Africa
   - Learning stories from others groups about how they manage projects; debating about OSS and its viability in Africa; discussing how to handle developer communities
   - Business models; different non-technical aspects of FOSS; different projects of FOSS that are in fact very similar to speed geeking
   - Understanding basic software development processes; not feeling intimidated by the perceived technical nature of the workshop due to the excellent facilitation process
   - Success/sustainable models; speed geeking; projects
   - Collaborating with others; learning about possible solutions to common problems
   - The openness; the sharing of information and experiences; the simple and relaxed approach
   - Success stories from participants; knowledge on OSS
   - Networking, learning about what others are doing, meeting new friends
   - Business models, meeting other attendees, being involved in new projects

- That I got an opportunity to meet possible collaborators that I got to learn that the problems I faced in FOSS were not unique.

3. **How do you see yourself applying what you've learned at the event? Are there specific next steps you anticipate?**
   - Creating awareness to friends about OSS and doing more on OSS development
   - Through the initiation of a practical project
   - Try to develop open source sustainability models involving commercialization
   - Yes. PRIDE Africa, AVOIR, OpenMRS, CODEFACE
   - Involving students in FOSS by creating FOSS projects; involving students in summer of code; popularizing FOSS
   - Contacting the devs and profs for collaboration
   - This is re-affirming our investment in OSS for development initiatives
   - I look forward to talking to and encouraging undergraduate computer science students to use the AVOIR framework for their academic projects
   - Yeah! Think we learnt some useful things from the AVOIR project ref. testing, deployment etc.
   - Yes. Applying some of the business models
   - I will certainly have a better conversation with potential project partners
   - Joining FOSS community; collaborating more freely
   - More participation in open source forums
   - Study some of the code of OpenMRS, the SMS gateway, the Drum Net application and be involved in the codeface initiatives
   - I am planning to have an elaborate understanding of OpenMRS and see how I can customize and to use the Chisimba framework for other purposes other than e-learning
   - Working on the collaboration exercise we agreed to carry forward
   - Information learning about business models and being able to provide informed advise about them
   - I am going to be following up with the new contacts I have made and will participate in mailing list.

4. **Have you established new collaborative relationships that will continue after this workshop? If so, please elaborate?**
   - Yes, through business contacts, email address (exchanging)
   - Yes, between AVOIR and the other projects and will continue through planned collaborative projects
   - Possible with TradeNet/Drum Net
   - Yes
   - Yes. With OpenMRS, Drum Net and AVOIR
   - Yes, with Dan at University of Nairobi; Joseph for OpenMRS work
   - I know that if we have a problem, I can communicate to anyone in the groups and they will respond
   - Yes, great to have established collaboration with Mark Davies
   - Yes, I did. I intend to play around with OpenMRS and see if I can convince the regional government hospital in my province to use it
   - I believe so. I think there are many opportunities but I'm afraid we might not be able to achieve some of the broad initiatives that were introduced. I am willing to provide my support to make these initiatives successful
   - Yes. On mobile applications
   - Yes, with Chris on OpenMRS mobile for Rwanda. Encouraging Drum Net to talk to the other partners e.g. TRADENET, OpenMRS mobile on the core application & Drum Net business model
   - Yes. We will work on a project to solve mobile phone interaction/interface
   - Project base
   - Hope to build a business relationship with Mark Davies to use his platform. Collaborate with Enver on "selling" the Mifos light in South Africa
   - Yes. I have met a friend who has promised to help me completely get into OSS and as such, we have exchanged contacts
   - Definitely – with TRADENET most specifically, but also with OpenMRS and AVOIR
   - Yes. Definitely will be working/collaborating with other projects using their platforms, or partnering to share resources
   - Yes. (i) going to be able to promote software I have learned about in detail more confidently; (ii) going to be able to "pin down" people on commitment they have made with a more human touch

Agenda

5. Was the agenda at a level appropriate for your needs?

- Yes, the agenda came from the participants including myself
- Yes, we defined it!
- Yes (X9)
- Yes, but would have liked other larger projects for us to derive knowledge from
- It was perfect
- Very appropriate
- Yes. I think it covered our needs
- Absolutely, it was very responsive to the needs of the participants and at a comfortable level for a non-techy as concepts were explained in simple terms
- Yes. It was nice that we could determine it our way
- Great – very collaborative, how could we say different

6. What was your assessment of the overall facilitation of the event?

- Very, very excellent. Quite a skill for the facilitator and the overall facilitation
- Awesome and groovy
- Excellent (X4)
- Superb – best I have ever experienced
- Great way to get all needs met
- Excellent, beyond expectations
- Rockin!
- Excellent. It was unique
- The facilitation was amazing. Very
- Very well coordinated and excellently and expertly facilitated
- Excellent, flexible, responsive, respectful, ensured balanced power among the participants
- Fantastic
- Cool
- Cannot find a better facilitator – Gunner is great, a big gun
- Great facilitation, energetic, relevant, flexible, fun
- Facilitation was brilliant A+

7. Which session(s) offered you the most benefit?

- Development and deployment of ISS
- They were all excellent
- Speed geeking, workgroups
- Licensing models; FOSS tools; business models, speed geeking
- Open source tools
- Various licensing models under FOSS
- Was only here for Friday; AM session on Mobile and so OSS and Business was excellent
- Speed geeking, 3 priority things that can scale up FOSS in Africa session
- Discussion on distributed development teams
- Business models and deployment of mobile telephony
- Introduction to FOSS; Licensing regimes; Business models; Speed geeking
- Project for collaboration; sustainable models ideals
- Collaboration
- Speed geeking and the group discussions
- Speed geeking
- All were beneficial
- Business models, discussing way forward, new projects
- FOSS business models; next steps after learning workshop.

8. **Which session(s) offered you the least benefit?**

- None (X12)
- How to spread open source
- All sessions were important
- It seemed to be just right
- The final group discussion on day 1
- I am not sure there was any
- User requirements
- The grid and seeing how people shifted (they didn't), session on tools because they were same old same

9. **Was the number of sessions offered too many, too few, or just the right amount?**
   - Just the right number/amount (X6)
   - Perhaps the last one was a bit rushed
   - Just right (X6)
   - Right
   - Could have done with more
   - The number was good enough, maybe have longer breaks right after a "controversial" topic for discussions (speed geeking)
   - Right amount given time

10. **What was not useful or something you would recommend omitting from similar events in the future?**
    - Additions: have more participation (participants)
    - All useful
    - None (X3)
    - Was all very useful
    - Can't remember
    - There could have been invitations from business people
    - Keep it so for now

11. **What was not covered at the event that you would have liked to see included in the agenda?**
    - Explore collaborative opportunities
    - Examples relevant to me
    - Government policies that can help popularize FOSS
    - Can't think of any right now (X2)
    - More on the psychology of collaboration
    - Simple practical exhibition on how to develop a software application from scratch
    - A simple business case on a successful FOSS in Africa
    - For the number of days, you couldn't do more
    - Participants would have been given a chance to share on future projects
    - Given time, the coverage was adequate
    - Success stories.

12. **Was the wiki a useful part of the event? Do you see yourself using it in the future?**
    - Yes. Yes, I see myself using it in the future.
    - Not sure
    - Yes (X 3)
    - Yes, for 6 months
    - Yes, reviewing topics I missed
    - Yes. And yes if it keeps updated
    - Yes, a good record of the event easily accessible and shareable afterwards
    - Never used it before but this Workshop has motivated me to use it
    - Not yet. But I'm positive that it will be useful in the future
    - Yes, I will use it for reference on the materials
    - Not used it before but will definitely be using it

- Did not use it much. I will use it in future
- Yes. It has and will continue to be a useful tool
- Need to learn more about wiki
- Initially, it didn't, but summary of notes at the end may be useful
- Very useful. Definitely

Logistics

13. Please rate the following statements on a scale of 1 to 5, where "1" is the lowest and "5" is the highest:

		Average totals (out of 5)
The conference facilities were appropriate	-	4.2
The food and beverages met my needs	-	4.4
The hotel accommodations met my needs	-	3.9
The pace of the day was comfortable and effective	-	4.6
Event administration (travel, documentation, finances) was effective	-	4.7

14. Please offer any other comments about the logistics that might help us improve future events:

- The holiday inn would have been much better. Luggage was lost
- Keep the same
- The information flow be a key at all stages

Overall Rating – average was 8.8 (out of 10).

15. Comments from Participants on their experience were as follows:

- I learnt many things and still have more room for learning many new things.
- One of the best organized and productive workshops ever
- The matching of event, location and agenda with opportunity for open discussion
- If I had a clear strategy (business case) to go FOSS and short term protection missing – more biz talk would have helped
- Good facilitation, great venue, good participants
- Didn't get all I was hoping out of it, but gathered enough feedback etc to justify time
- It met more than my expectations. The facilitation was fantastic
- It was great learning more about how other OSS and non-OSS projects work, about business models for OSS. I wish there had been more projects so we had a broader range of OSS experience
- Enriched my knowledge deeply, was flexible and responsive to my needs and not too technical
- It has opened a gateway to me who is NOT an IT person but manages ICT mediated project
- There is need to bring more diverse audiences especially from business world
- Addressed all the issues to my expectation
- The only thing is not being able to attend all sessions
- Close to perfect other than the location (traffic).

Other Questions

16. Who was not here who should have been here?

- African (champions) mentors in OSS to the young generation
- Guide – if supplied with mute button
- Government representation from ICT; more successful FOSS implementers and users from the African Continent
- Other OSS partners but this was a good number to start with
- My boss, Dr. Elijah Omwenga. He would have gained great FOSS idea that he could convince the University to adopt
- A few more African OSS communities/projects
- Dr. Felix Akorli would have contributed well on the mobile stuff – is a mobile engineer
- Ministry of Information and Communication officials; mobile phone providers
- Business executives
- As far as FOSS methodology and real life experience think that some attendees got more out of the discussions. It would have been nice to have a higher level speaker
- Business (software companies).

17. What other suggestions might you offer the event organizers and facilitators to improve on the event?

- Hotel in centre of town
- Invite successful role models
- Gunner is one of the best
- Like the small group discussions and then reporting back to the larger group
- Next session should be in Jamaica or Bahamas!
- Offer a little bit more practical (speed geeking) sessions
- Do a follow up even within a year if possible
- Representation of local developers and institutions of higher learning
- Getting more training on the Wiki by participants. Somehow motivating people to talk on the Wiki in preparation for the workshop.