A Streamlined Process to Open Source Proprietary Technology

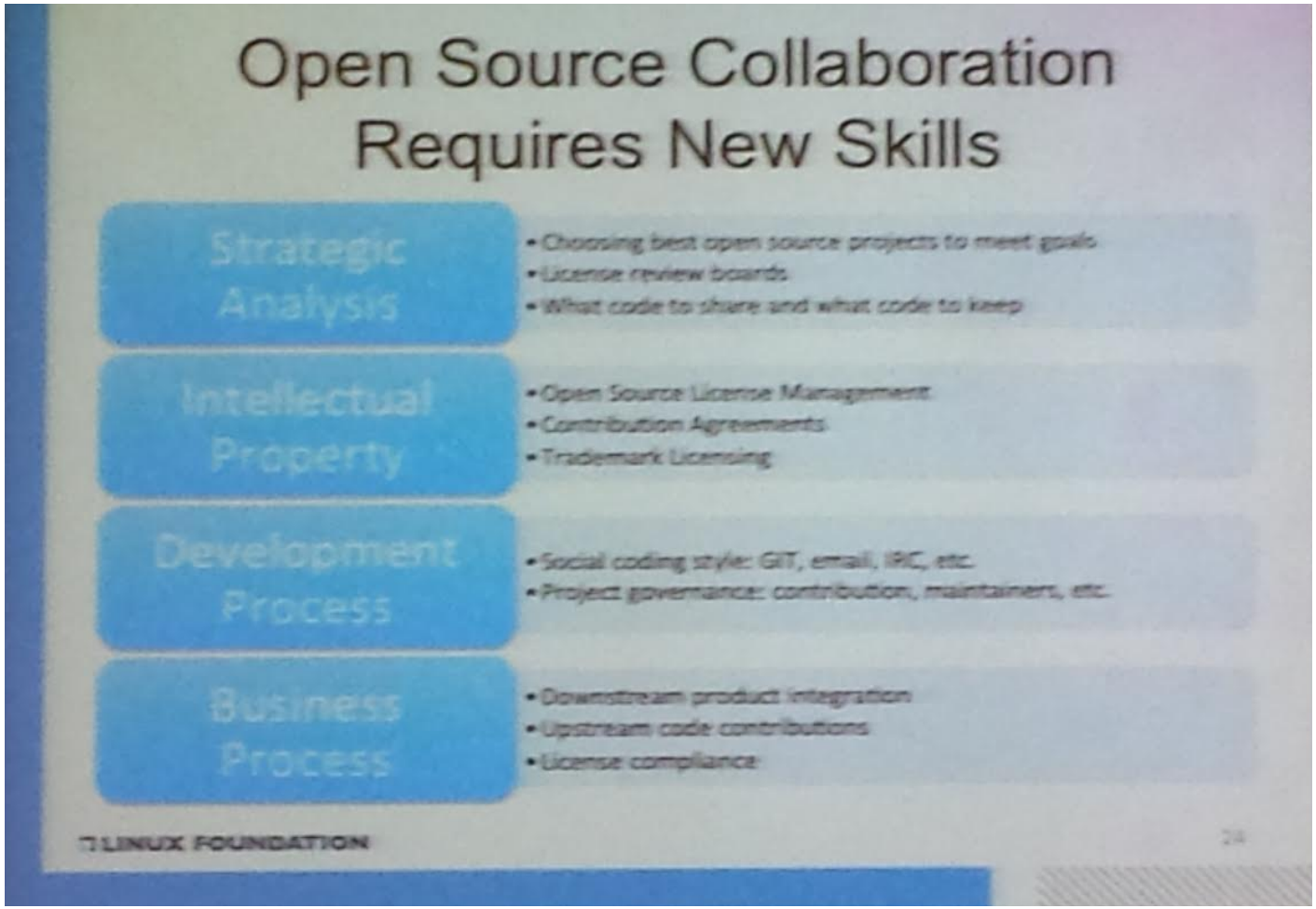
ソース内部技術をオープンするための処理

Ibrahim Haddad, Ph.D. I 工学博士　イブラヒム・ハッダド
Head of Open Source Group
Samsung Research America – Silicon Valley
@IbrahimAtLinux

A Practical Guide to Starting a New Open Source Project and to Open Sourcing Proprietary Source Code

Why?

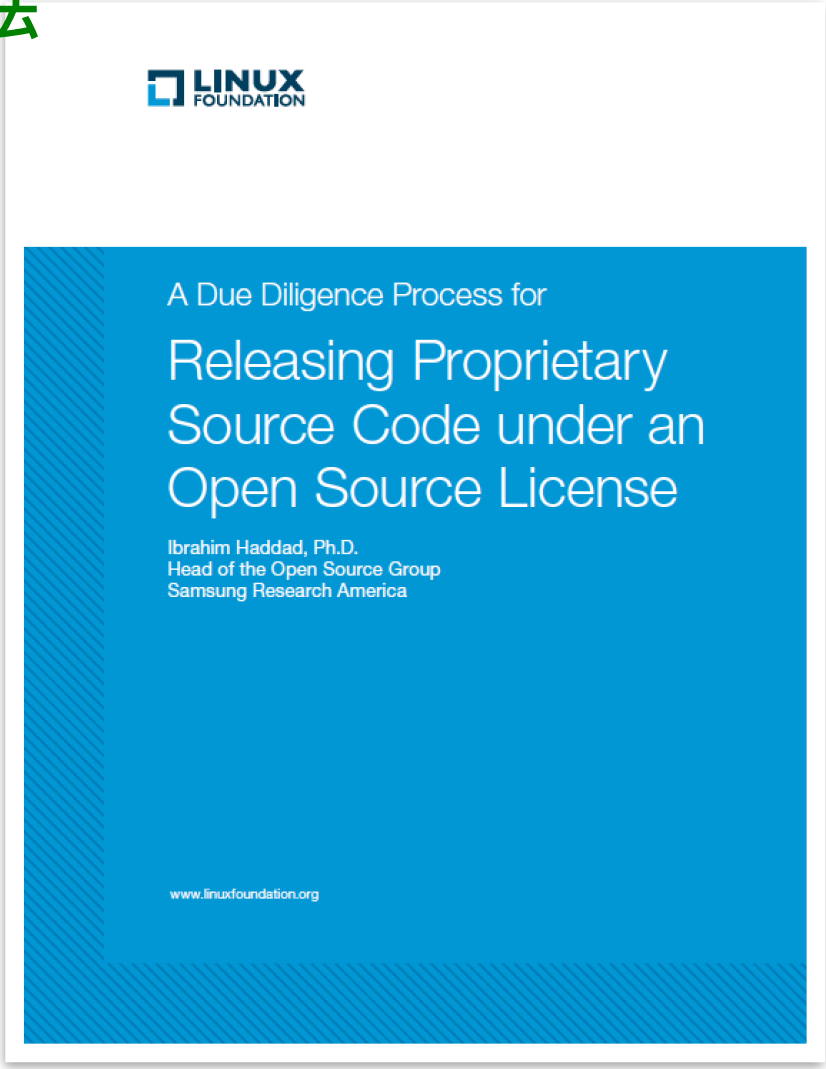
Jim Zemlin
Executive Director
The Linux Foundation
LinuxCon JP Keynote 2014



Full Paper

オープンソースライセンスの下で独自のソースコードを公開する方法

Paper is available from:
http://www.linuxfoundation.org/

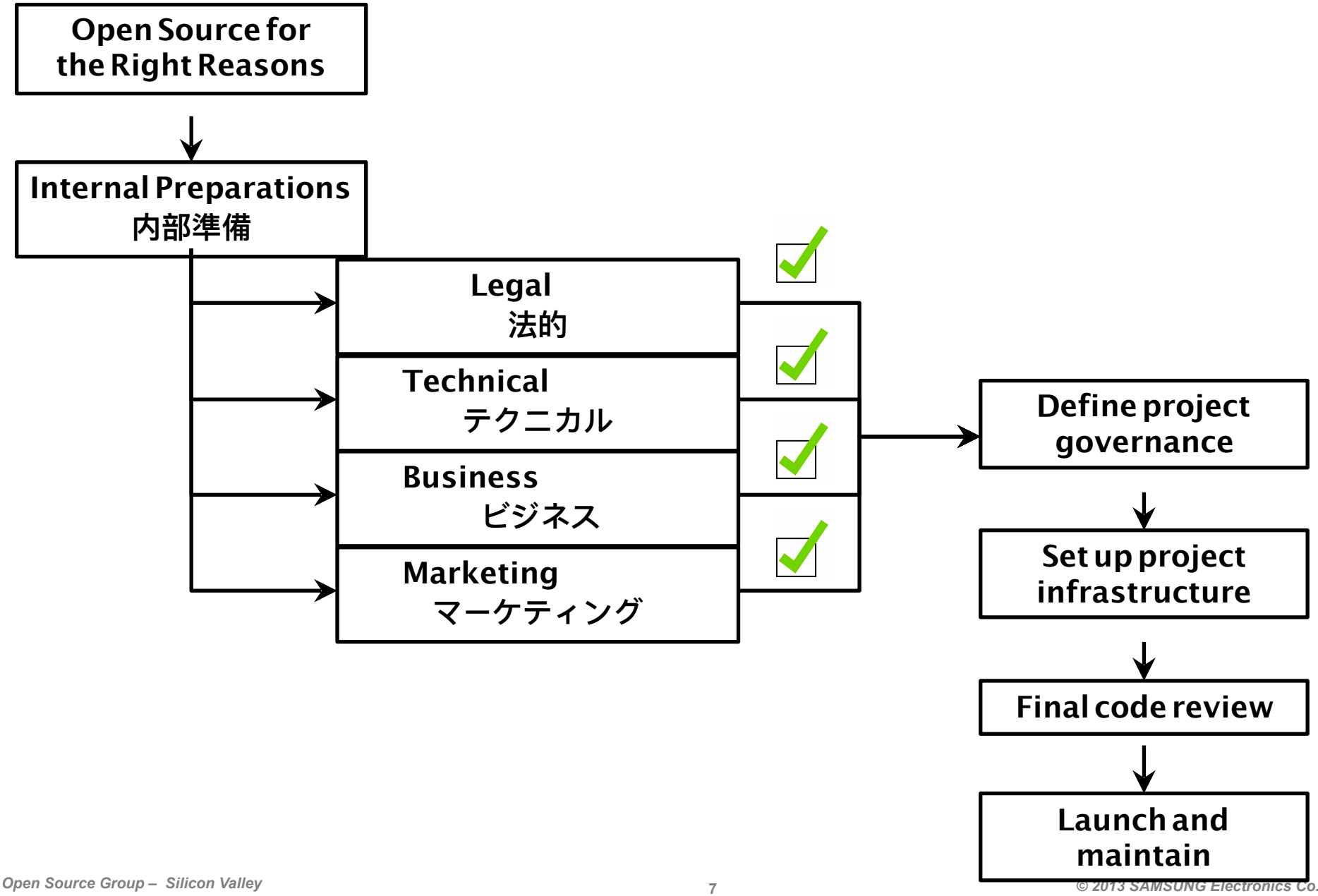


Content

Process Overview

プロセスの概要

High Level Process Overview



It starts with open sourcing for the right reasons

Business Preparations

- **Determine the overall strategy**

  - “Why are we doing this?”
  - Business case/value proposition

- **Ensure there is an executive champion for the project**

  - Commitment for resources (developers)
  - Commitment for funding (project infrastructure, events, travel, etc.)

Legal Preparations

- Patent Scrub
  - Identify IP in the existing source code to be made available
- Decide on License
- Decide on trademark
  - Use, requirements, and process
- Ensure a clean BoM for the code to be released
  - Identify all source code originating from outside your company
- Other items as necessary

License Selection Considerations

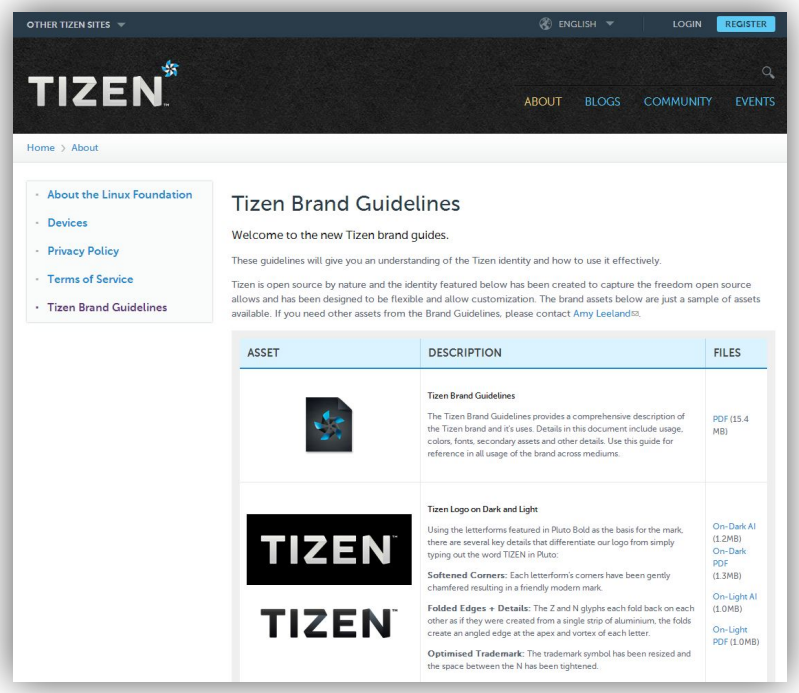
- **Use a mainstream OSI-approved license**
  - opensource.org
- **Some factors to consider in choosing a license:**
  - Do you want to relinquish any control over how your code is used and distributed?
  - Do you want to allow people to use your code in non open-source programs?
  - If somebody uses your code in their program and sells their program for money, do you want some of that money?
  - Do you want to ensure all future derivatives distribute source code?
  - Do you want your code to be usable in both open source and proprietary products?
  - If it's a library, should non-open source code be allowed to link to it?
  - Etc.

Source Code Preparations

- **Clean up the code**
  - Ensure coding style is consistent
  - Clean up internal comments, references to internal code, etc.
  - Remove any code not part of open sourcing plan
  - Remove dependencies on non-public components
- **Add open source license and copyright notices**
  - Add license notice in source code files
  - Add license text as a file in the root directory
  - Update copyright notices in source code files
- **Prepare the code for new external users**
  - Provide documentation and use case examples
  - Ensure it compiles and builds on target platforms
  - Fully document all APIs
  - Etc.

Marketing Preparations

- **Design project logo, color scheme, website, collateral, etc.**
- **Formalize branding guidelines**
  - Do you need a compliance program?
- **Register social media accounts for the project**
- **Register domain names**
- **Etc.**



Project Governance

プロジェクトガバナンス

Project Governance

- **Companies vs. individuals**
- **Define the project structure that enables decisions on:**
  - Participation guidelines and requirements
  - Architectural changes
  - Nominating maintainers
  - Final arbiter on disputes
  - Suspension of participants
  - Etc.
- **Often similar to a board of directors**
  - Typically represents mix of project contributors

Governance Decisions

- **All projects should define and communicate, at minimum, processes for:**
  - Submitting code, patches, feature ideas, etc.
  - Technical conflict resolution
  - Release management
  - Etc.

- **A multi-company project may need more formal governance**
  - Decide on how governance is structured, who can participate, for how long, etc.

Basic Recommendations

- **Governing body should represent various participating entities**
- **Democratic system**
- **Clear decision-making process**
- **Clear path to resolve disputes**
- **Flexibility to adapt to changing project needs**
- **Clear means to add new or replace members**

➔ For good examples, check the governance of projects hosted at the Linux Foundation

Project Maintainer

- **A formal position of leadership within the project**

Define the “Project Maintainer Role”

- Setting the criteria for accepted / rejected code
- Reviewing submitted code / accept and reject based on pre-defined rules
- Tracking dependency issues
- Notifying developers of source code changes that may affect their packages
- Managing source code security issues
- Working closely with team developing the source code
- Working closely with QA team testing the source code
- Dealing with reported bugs in a timely manner
- Preparing binaries – packages of the source code

Required Project Infrastructure

必要なプロジェクト･インフラ

Project Infrastructure

- Essential components:
  - Web site
  - Wiki (User contributed documentation)
  - Git or other source code repository system
  - Mailing lists
  - IRC
  - Bug and feature tracking

Project Infrastructure Examples: Front Page

What the project does

Where to find the code

Who is involved

LTSI Workgroup | The Linu

ltsi.linuxfoundation.org

LONG TERM SUPPORT INITIATIVE

Home | What is LTSI? | Developers | Users | Downloads | Participate

The LTSI is an ecosystem-wide collaborative project hosted at the Linux Foundation to create and maintain a common Linux base for the use in a variety of CE products and to enable faster contributions upstream and better alignment with the mainline kernel.

Embedded Linux Conference

February 15 - 17, 2012 ▪ Hotel Sofitel San Francisco Bay ▪ Redwood Shores, CA

February 15 - 17, 2012 · Hotel Sofitel SF Bay, Redwood Shores, CA

The Embedded Linux Conference (ELC) is the premier vendor-neutral technical conference for companies and developers using Linux in embedded products.

ELC is embedded Linux experts talking about solutions to your embedded Linux problems. ELC consists of 3 days of presentations, tutorials and Bird-of-a-Feather sessions. There are over 50 sessions to choose from, on a wide variety of topics.

Submit a proposal today!

LTSI Blog

The latest from LTSI.

Introducing the LTSI Project

Submitted by brian on December 26, 2011 - 00:42

On October 25, 2011, The Linux Foundation, the nonprofit organization dedicated to accelerating the growth of Linux, announced it is hosting a new project created by its Consumer Electronics (CE) workgroup.

The new project, the Long Term Support Initiative (LTSI), provides for both an annual release of a Linux kernel suitable for supporting the lifespan of consumer electronics products and regular updates of those releases for two years.

more blogs

LTSI Releases

LTSI Trees.

- Current LTSI Tree: 3.0.0
- Current LTSI Staging Tree: 3.0.0
- Current Industry Staging Tree: 3.0.0

more releases

Participating Organizations

See who is involved in LTSI.

HITACHI, hp, IBM, intel, NEC, Panasonic, Qualcomm ATHEROS, RENESAS, SAMSUNG, SONY

Project Infrastructure Examples: Get Involved

Participate in LTSI | LTSI W

ltsi.linuxfoundation.org/participate-in-ltsi

LONG TERM SUPPORT INITIATIVE

Home | What is LTSI? | Developers | Users | Downloads | Participate

- Home
- What is LTSI?
- Developers
- Users
- Downloads
- Participate
  - Guidelines for Participation
  - Blog
  - Events
  - IRC
  - Mailing Lists

Home › Participate in LTSI

Participate in LTSI

Introduction

The LTSI is an ecosystem-wide collaborative project hosted at the Linux Foundation to create and maintain a common Linux base for the use in a variety of CE products and to enable faster contributions upstream and better alignment with the mainline kernel.

It offers a framework for collaboration between CE industry players and the kernel community with the goal of strengthening the Linux technical platform.

Ways to Participate

- Join the discussions on the mailing lists or IRC channel
- Submit bug reports
- Submit bug fixes
- Submit ideas for new features
- Submit code for new functionality
- Improve documentation
- Help review / janitor the code prior to upstream submission
- Attend our meetings and events

THE LINUX FOUNDATION

Project Infrastructure Examples: Processes

Understanding the OpenM

www.openmama.org/developers/understanding-openmama-development-process

OpenMAMA

Search

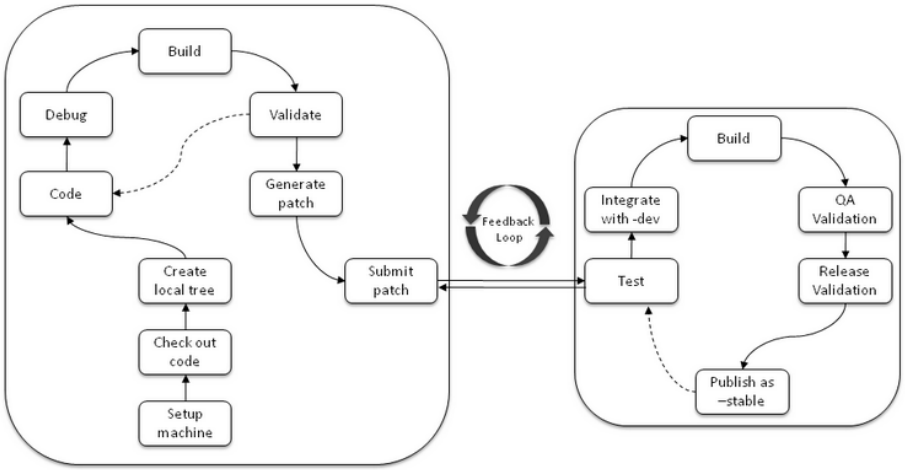
Log in / Create account

Home | What is OpenMAMA? | Developers | Users | Downloads | Participate

- Home
- What is OpenMAMA?
- Developers
  - Architecture
  - APIs
  - Bug Tracking
  - **Development Process**
  - Feature Requests
  - Patch Submission
  - Signed-off-by Process
  - Using git
- Users
- Downloads
- Participate

Home ›

Understanding the OpenMAMA Development Process



Driven by participating developers

Driven by maintainer

OpenMAMA development is modeled extensively upon the proven Linux kernel development process, where developers modify project code by submitting patches as emails to a public mailing list. It is the OpenMAMA maintainer's responsibility to decide whether submitted code should be integrated into the mainline code, returned for revision, or rejected.

Individual developers maintain a local copy of the OpenMAMA codebase using the git revision control system. Git ensures that all participants are working with a common and up-to-date code base at all times. Each developer works to develop, debug, build, and validate their own code against the current OpenMAMA codebase, so that when the time comes to integrate into the mainline project, their changes apply cleanly and with a minimum amount of merging effort.

When a developer is confident that their code is ready for integration, they generate a patch, sign off on their code, and email it to the openmama-dev@lists.openmama.org mailing list. The OpenMAMA maintainer watches the mailing list for patches, tests the code, and accepts or rejects patches accordingly.

When a patch has been accepted, it is integrated into the -dev branch, which means that other developers will now be able to see the new code when they update their local git repository. It is built, tested, and if no additional issues arise, published as part of a -stable OpenMAMA release.

After the patch has been accepted, it remains the developer's responsibility to maintain the code throughout its lifecycle.

Project Infrastructure Examples: Processes

Submitting Requirements

www.openmama.org/developers/submitting-requirements-feature-requests

OpenMAMA

Search

Log in / Create account

Home | What is OpenMAMA? | Developers | Users | Downloads | Participate

- Home
- What is OpenMAMA?
- Developers
  - Architecture
  - APIs
  - Bug Tracking
  - Development Process
  - **Feature Requests**
    - Sample Email
  - Patch Submission
  - Signed-off-by Process
  - Using git
- Users
- Downloads
- Participate

Home ›

Submitting Requirements & Feature Requests

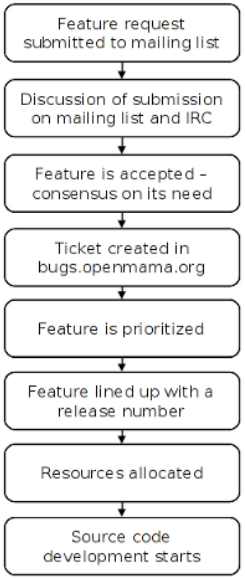
OpenMAMA is an open source project that accepts contribution from its user and development communities. Input can be in various forms: reporting bugs, submitting patches, submitting requests and code for new features, etc.

As with any open source project, when project contributors are actively involved in the project by providing useful feedback and high quality code submissions, you will build trust with the project maintainers and increase the effectiveness of your interactions. As you build trust within the community for the quality of your work, you will be given increased opportunity to help set direction.

The feature request process is meant to ensure that requests are captured, discussed, prioritized, and if accepted, planned for development and release in an orderly fashion.

Before you open a feature request, please make sure you are using the right process.

New Feature Request Process



Tracking Requirements and Feature Requests

Requirements and feature requests are tracked and prioritized in public via http://bugs.openmama.org, prior to acceptance into OpenMAMA. This ensures a common understanding of which features have been requested, their relative priority, their development status and when they are planned for release.

Major feature requests go through multiple stages of feedback as explained below. This ensures that a request is strategic to the direction of OpenMAMA before resources are dedicated to implement it, and also that there is a broad understanding of why the feature is needed and that there is willingness to maintain the feature through its lifecycle.

1. Feature requests begin with a proposal to the openmama-dev@lists.openmama.org mailing list or as a feature request in http://bugs.openmama.org by the person who has the idea and whoever is likely to do the work implementing the requested feature. This is to notify others, solicit feedback, and gain acceptance for the idea, and come to a consensus on next steps.

   Please use the OpenMAMA standard format when posting feature request emails.

2. Next, create a feature request at http://bugs.openmama.org (only if you have not done so originally). When you create the request, please be sure that you click "Show Advanced Fields," and select "Feature request" as the category, and include as much detail as you can.

   Additional feedback may be provided in the comments area.

3. The OpenMAMA system or subsystem maintainer will evaluate the request, and determine whether it is a candidate for a future release. There will be discussions between the requester and the development team on the mailing list. If the request is approved, a target release will be set, and development can begin in earnest.

Submitting the Feature

Project Infrastructure Examples: Sign-off-by

Signed-off-by Process | Op

www.openmama.org/developers/signed-process

openMAMA

Search

Log in / Create account

Home | What is OpenMAMA? | Developers | Users | Downloads | Participate

- Home
- What is OpenMAMA?
- Developers
  - Architecture
  - APIs
  - Bug Tracking
  - Development Process
  - Feature Requests
  - Patch Submission
  - **Signed-off-by Process**
  - Using git
- Users
- Downloads
- Participate

Home ›

Signed-off-by Process

The OpenMAMA project uses the signed-off-by language and process, used by the Linux kernel, to give us a clear chain of trust for every patch received.

Linux Kernel Certificate of Origin v 1.1

"By making a contribution to this project, I certify that:

a. The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

b. The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

c. The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

d. I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved."

Using this Process

We have the same requirements for using the signed-off-by process as the Linux kernel.

In short, you need to include a signed-off-by tag in every patch:

> Signed-off-by: this is a developer's certification that he or she has the right to submit the patch for inclusion into the project. It is an agreement to the Developer's Certificate of Origin (above). Code without a proper signoff cannot be merged into the mainline.

If you are unfamiliar with this process, you should read the official policy at kernel.org and you might find this article about participating in the Linux community on the Linux Foundation website to be a helpful resource.

Home
Home
About
Developers
Users
Downloads
Participate

About
FAQs
Participating Companies
Privacy Policy
Terms of Service
Trademark

Training
Linux Performance Tuning
Linux Developer Training
Open Source Compliance

THE LINUX FOUNDATION

Final Reviews Before Release

リリース前の最終レビュー

Final Reviews (i.e. Final Final)

- **All requirements identified by the business, legal, technical, and marketing reviews are completely met**
- **Examples:**
  - License, attribution, and copyright texts are all complete and in place
  - Source code scanner reports clean bill of materials
  - Every line of code is licensed appropriately for release
  - Comments are sanitized of casual or unrelated language
  - Source code does not inadvertently reveal internal projects

Operation “Launch”

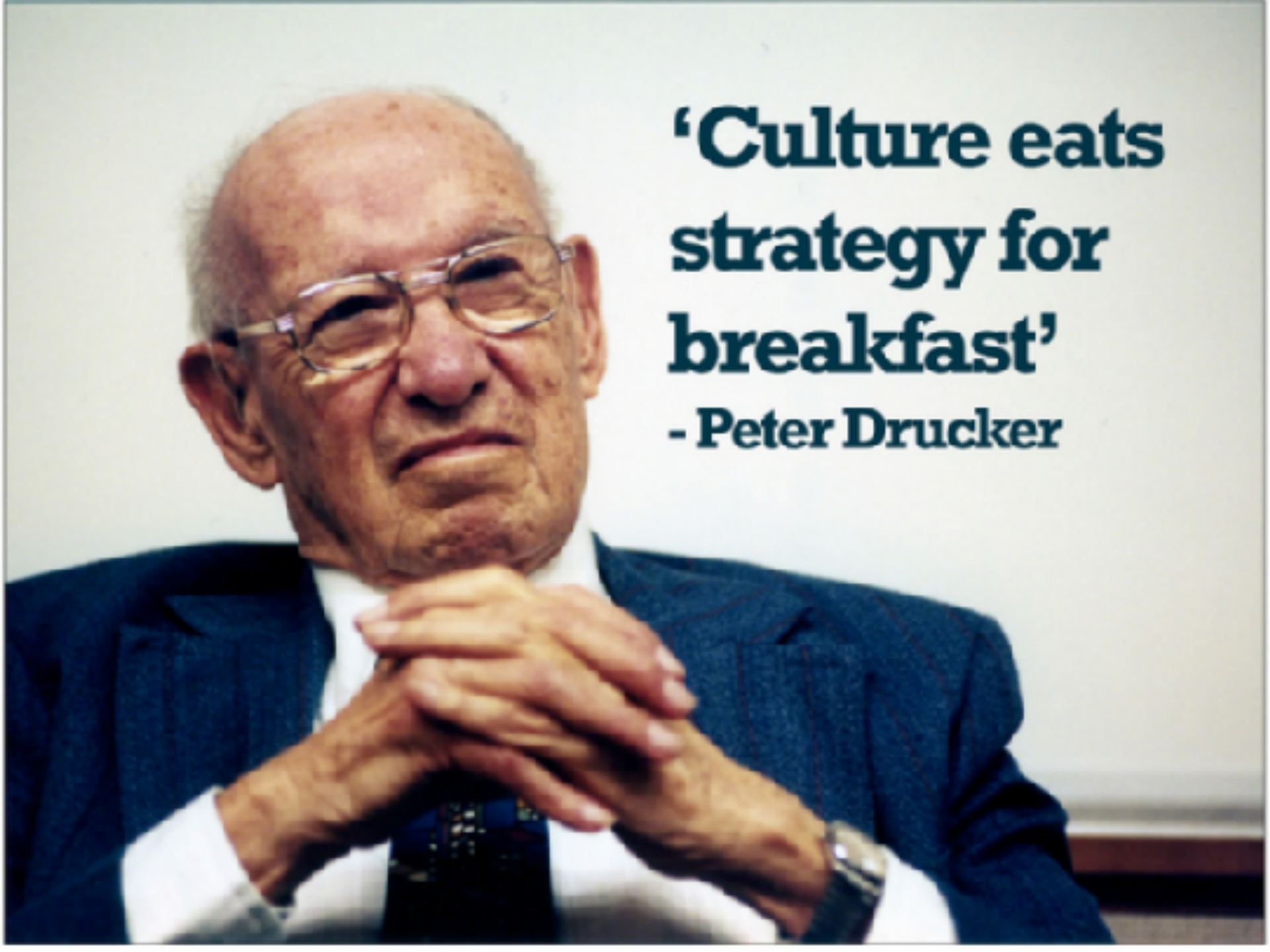
オペレーション"打ち上げ"

Prior To Launch

- **Build critical mass before launching**
  - Provide preview to customers and partners so they can begin to work with the code
  - Lobby for launch-day participants among your existing business partners
- **Ensure that all project infrastructure is running, secure, and scalable**
- **Upload the code**
- **Ensure internal developers join and continually monitor IRC channel, mailing lists, etc.**

Is this your first time doing this?

- **Train your employees and managers**
  - Open source development methods and processes
  - Working with the open source community
  - Your company's open source policies and compliance rules
  - Integrating open source software within your software development model
- **Follow open source practices internally**


'Culture eats strategy for breakfast'
- Peter Drucker

After the launch

- **Work on building a developer community**
  - Is it easy to find and join as an outsider?
  - Does the community have the documentation they need, and a means to update it?
  - Is the process for accepting community code working?
- **Follow open source development model & practices**
- **Remain visible**

Be a Good Open Source Citizen

- **Have conversations and make decisions in the open**
  - Builds goodwill, but also reduces overhead in documenting decisions
  - Streamlines onboarding process for new participants
  - Archives ensure continuity if participants change
- **Listen to the community**
  - They know what they are doing, particularly on integration and testing
  - Encourage generalized implementations that extend what you need, particularly if someone else volunteers

Closing

結論


It takes whole community
to raise an open source
project

Leadership != Control

リーダーシップ != コントロール

Thank you goes to the thousands of open source
developers. Not just for the source , code but most importantly for innovating a better way to create software.