How to Deliver High Quality Commercial Products with Open Source Software

Greg Olson, Sr. Director Open Source Consulting
Bill Weinberg, Sr. Director/Analyst, Open Source Strategy
Agenda

- Introductions
- Companies and Open Source Projects
- The OSS-specific elements in product development
- Summary
The Linux Foundation Consulting

- Multiple decades of open source consulting experience
- Over 300 engagements assisting companies from start-ups to the world’s largest corporations
- Deep operational experience in executive management, marketing, finance, sales, business development and software development
Companies and OSS Projects

Product Companies
Industrial Process
- Top-Down Management
- Formal Methodology
- Processes
- Metrics
- Revenue
- Contracts
- SLAs
- Customer Organizations

Conceptual Divide

Open Source Projects
Collaborative Process
- Consensus around Technical Merit
- Sponsors
- Developers
- Members
- Diffuse Processes
- Tools
- Scripts
- Repository
- Wiki
- Lists
- Users

THE LINUX FOUNDATION
Typical Commercial Product Development Cycle

Product Strategy → Product Definition → Product Plan → Product Dev → Qualify → Launch → Maintain

→ Market Diversification feature

→ Architectural feature

→ Minor feature

→ Next Release

→ Patch
Product Development with OSS

OSS Community Strategy

Strategic OSS Eval & Selection
Tactical OSS Eval & Selection

Differentiation Analysis

Product Evaluation

OSS Integration Plan
Test Plan

OSS Integration

OSS Test Program

Bug Reporting
Patches & Upstreaming

Manage Incoming Patches
Merge Releases

Product Dev

Qualified

Launch

Maintain

Patch

Next Release

Market Diversification
feature

Architectural feature

Minor feature

Product Strategy

Product Definition

Product Plan

Product Strategy
OSS-Specific Dimensions of Product Development

1. OSS Community Strategy
2. OSS Evaluation & Selection
3. Differentiation Analysis (Proprietary or OSS)
4. OSS Integration & Test Plan
5. OSS Integration
6. OSS Test Program
7. Bug Reporting, Patches and Upstreaming
8. Manage Incoming Patches and Releases
1. Community Strategy

- Community strategies typically evolve organically but benefit from conscious planning

- Identified Best Practices
  - Select strategically important OSS projects for focus
  - Seek committer / maintainer roles in identified project communities
  - Organizations adapt to OSS project culture, practices and tools to succeed with their strategically important projects
    - Each project is somewhat unique in this regard
    - Adapt on a project by project basis
2. OSS Evaluation & Selection

- There is tremendous leverage in choosing the right OSS project and community at the outset
- Most survey respondents required at least some of “due diligence”
- In most cases engineers discover and request an OSS project
  - Criteria are predominantly technical
- Licensing is also reviewed carefully in most companies
- Support and maintenance dimensions are often neglected in the evaluation processes of technology vendors.
3. Differentiation Analysis

- Deciding whether each feature is proprietary or open source is a constant activity with proprietary products built upon or coupled with OSS distributions
  - What best supports your company’s product and market strategies?
  - Even previous decisions should be re-evaluated periodically to accommodate changes in product landscape and competitive strategies.
- Note that any features or customizations not likely to be accepted by the OSS Project are inevitably proprietary
- Good decisions require a multi-dimensional evaluation (next slide)
4. OSS Integration and Test Planning

• When most of the code for a product is sourced from a single OSS project, normalizing your own engineering practices with those of that project
  • Seamless interoperability with code repo, bug tracking, release process, etc.
  • Faster on-boarding of contributors to the relevant OSS project

• Primarily test and QA OSS code during/post integration together with dependencies and value-added product software and hardware
  • Utilize OSS project test code when available
  • Develop some in-house tests directed specifically at OSS where customer or market requirements dictate
5. OSS Integration

- Research results indicate
  - Respondents integrate high percentages of OSS code into products
  - Large and small organizations integrate directly from OSS trees
    - Product teams given large degree of freedom to choose appropriate versions
  - Strictly minimize customization of OSS to keep patch loads manageable
  - Modularize changes, extensions to the OSS wherever possible
  - Leverage automated continuous integration to
    - Minimize pain from update and merge
    - Track OSS project trees most closely
6. OSS Test Program

- Need to test OSS standalone and as integrated code
  - OSS module unit testing
  - OSS project / sub-system and/or platform testing
  - Final product testing with integrated open source code

- Successful organizations integrating open source
  - Aggressively contribute test code to projects so that releases arrive pre-tested
  - Develop relationships with OSS project leaders to facilitate upstreaming
7. Bug Reports, Patches and Upstreaming

- Research reveals common core practices for upstreaming
  - Most successful organizations invest in upstreaming early
    - Build community / maintainer relationships
    - Retain minimal forked code as “value-added”
  - Large Orgs (Samsung, Red Hat et al.)
    - Company ID does not guarantee upstream patch acceptance
    - Able to dedicate more resources on upstream interface
  - Small Orgs (smaller OEMs and integrators)
    - Patches reviewed on merit, as with large contributors
    - Even more important to consider project style, roadmaps, etc.
8. Manage Incoming Patches/Releases

- Simultaneous development by OSS projects and by product development and support teams must be reliably and efficiently merged and tested.
- Complexity of the problem often leads to slow and expensive processes.
- Best practices and research findings dictate to
  - Strictly minimize customizations in order to keep the patch load manageable.
  - Keep retained changes small and modular to streamline merging.
  - Cultivate OSS project relationships to enhance communication and minimize skew.
  - Invest in project test code to minimize quality issues in OSS updates.
  - Use available tools merging capabilities (patch, git/github, etc.).
Summary

- Our research suggests a number of ways that companies can structure their development processes to improve
  - Quality of its OSS-based product releases
  - Quality of support and mean time to fix critical issues
  - Predictability of their OSS development resource requirements
  - Efficiency of their OSS development and management