


theLisboncouncil
think tank for the 21st century

Open Source, Modular Platforms, and the Challenge of Fragmentation



Christopher S. Yoo

University of Pennsylvania Law School

November 22, 2016

Organization of Talk

- Describe two concepts underlying many operating systems: open source and modular platforms
- Analyze the relationship between the concepts
 - Complementarity between the concepts
 - Tension between the concepts
- Examine past open source operating systems: Unix, Symbian, and Linux
- Explore the implications for the modern debate

Two Related, But Distinct Concepts

- Open source software
 - Permit free modification of software
 - Recruit a community of people to improve the system
 - Organized like a “bazaar,” not a “cathedral”
 - Linus’s Law: Given enough eyeballs, all bugs are shallow.
- Two approaches to managing complex systems
 - Traditional approach: maintain strict control
 - Modular approach: divide into smaller subsystems
 - Divide system into modules with carefully designed interfaces
 - Rely on the architecture to organize the system

The Complex Relationship Between Open Source and Modular Platforms

- The essential connection
 - Enables third-party provision
 - Allows independent and parallel experimentation
- The essential tension
 - Open source = total freedom
 - Modular platform = strict adherence to architecture
- Potential problems resulting from the tension
 - Noncompliant modules
 - Fragmentation (can't “write once, run anywhere”)

Solutions for Resolving the Tension

- Solutions to the tension
 - Informal governance
 - Testing (centralized or decentralized)
 - Formal governance
 - Open source projects are often called benevolent dictatorships
 - All “bazaars” have “cathedral”-like aspects

Lessons from the Past

■ Unix

- Forking forced app developers to create multiple versions
- In the absence of leadership, never found a way to unify

■ Symbian

- Supported many form factors and devices
- Killed by lack of leadership

■ Linux

- Overseen by Linux creator, Linus Torvalds
 - Contradicts myth of bottom-up ordering and meritocracy
-

Implications for Preventing Fragmentation and Incompatibility

- Alternative approaches to testing
 - No testing
 - End-user testing
 - Vertical integration
 - Centralized testing
 - Self-certification through public testing tools
- Need strong leadership/governance

Android's Approach to Preventing Fragmentation

- Compatibility definition (CDD)
 - Must include Desk Clock, Browser, Calendar, Contacts, Gallery, Global Search, Launcher, Music, and Settings
 - Can use Google's versions or substitute other versions
 - Can self-certify using free testing tools (CTS)
- Anti-Fragmentation Agreement (AFA)
 - All Android devices must comply with the CDD
 - Device manufacturer must not fragment Android (i.e., create devices that do not comply with the CDD)

Conclusions

- Some restrictions on open source components are necessary to ensure compatibility/prevent forking
- Real question is whether particular restrictions are reasonable
 - Licenses are free and nonexclusive
 - Underlying code is open source
- Restricting ability to limit fragmentation can create incentives to make platforms proprietary
- Optimal outcome may be a mix of strategies



theLisboncouncil
think tank for the 21st century