INF5750

Open Source

University of Oslo
Department of Informatics

Outline

- Commons Based Peer Production (CBPP)
- What is (F)(L)OSS?
- Historical context - free vs open
- Development process
- Business models for OSS development
- ~~Open source and Open APIs~~

Commons Based Peer Production

- A "[…] model of socioeconomic production in which large numbers of people work cooperatively (usually over the Internet)" (Wikipedia).
- Coined by Benkler (2002), in a study of open source software development
- Contrasted with the typical models of production:
    - firm production - centralised hierarchy in which tasks are defined and distributed
    - market production - tasks are tagged with a price to attract workers
- No clear-cut distinction with crowdsourcing, but usually involves a stronger sense of community

Conditions for CBPP

- Premise: existence of excess capacity, resulting from a great number of potential contributors and a set of organisational structures
- CBPP model requires that work can be modularised
- Work is divided into modules which can be:
    - independently and incrementally produced
    - sufficiently fine-grained to allow the capture of small contributions
    - quality-checked and integrated with the overall system through reasonably low cost mechanisms

Examples of CBPP

- Free and Open Source Software was the inspiration for the CBPP project
- Wikipedia - ± 30 000 active contributors with 5+ edits per month
- OpenStreetMaps - ±50 000 active contributors per month

Free and Open Source Software

[…] anyone is freely licensed to use, copy, study, and change the software in any way, and the source code is openly shared so that people are encouraged to voluntarily improve the design of the software.

– *Wikipedia*

FOSS

- Software is created by an author and is subject to *copyright*
- A *license* is needed for software to be used by others
- More on different licenses next week.
- The term *open source* coined by Open Source Initiative (OSI), established in 1999
- OSI has a list of 10 criteria for OSS to comply with

1. Free redistribution

- No restriction on redistribution (free or paid) of the software

- The software can be redistributed alone or as a component of an aggregate software distribution

- The licensee can not require a royalty or fee for redistribution

2. Source code

- The software must include the source code, or it must be easily obtainable through well-published means

- The source code should not be deliberately obfuscated, and intermediate forms (preprocessor/translator output) are not allowed.

3. Derived works

- The license must allow modification and derived works
- Derived works must be allowed to be distributed under the same terms as the original

4. Integrity of the author's source code

- The license may only restrict source-code from being distributed in modified form if it allows patch-files that can modify it at build time
- The license might require derived works to use a different name and/or version number

5. No discrimination against persons or groups

- No discrimination against persons or groups

6. No discrimination against fields of endeavour

- Restrictions in the use of the software in particular fields of endeavour is not allowed

7. Distribution of license

- License for software must also apply to those it is redistributed to

8. License must not be specific to a product

- License for software must not depend on it being part of a software distribution
- The same license must apply if the software is extracted from a distribution and distributed separately

9. License must not restrict other software

- The license must not place restrictions on other software that is distributed alongside the licensed software

10. License must be technology-neutral

- No provision of the license may be predicated on any individual technology or style of interface

Free vs Open

- Philosophical differences between *free* and *open*
- Free Software Foundation (FSF) founded in 1985 by Richard Stallman to promote *free* software
- *Free* refers to *freedom*, not zero cost

Four freedoms for software

0. The freedom to run the program, for any purpose.

1. The freedom to study how the program works, and change it to make it do what you wish.

2. The freedom to redistribute copies so you can help your neighbour.

3. The freedom to distribute copies of your modified versions to others. But doing this you can give the whole community a chance to benefit from your changes.

Source: https://www.gnu.org/philosophy/free-sw.en.html

Free vs Open

- Free software refer to freedom, not cost - "free speech", not "free beer"
- Based on promoting social solidarity and sharing
- *Free* software meet the 10 criteria for open source
- *Open source* software do not follow the four freedoms
- Practical difference: *free* licenses (e.g. GPL) require derivative work to be open source

Types of software

Software type	Free (cost)	Redistributable	Unlimited use and users	Source code available	Source code modifiable
Commercial					
Shareware	X	X			
Freeware	X	X	X		
Royalty-free libraries	X	X	X	X	
Open source	X	X	X	X	X

Historical context

- In the early days of programming, sharing of software among programmers was the norm
- Hardware vendors started to dominate software distribution in the 1980s, releasing proprietary software in binary form
- FSF established in 1985 to re-establish free software norms
- In the second half of the 1990s, the internet facilitated distributed OSS development
- OSI founded in 1998 to promote OSS as a solution for businesses

Free vs Open (2)

- Still debate among proponents of "free" and "open source" proponents
- Main contention
- "Free speech"

Models for production of software

1. Managerial command systems - firms and organisations with "lines of command"

2. Markets - transaction costs define the production

3. Commons Based Peer Production

- OSS can follow any of the models, but peer production is perhaps the "typical" example

The open source approach

- Feller and Fitzgerald (2002) analyses the OSS development approach along 5 dimensions:

  What, Why, When and Where, How, Who

- Based on Zachman's framework of IS architecture and Checkland's CATWOE technique

- Fitzgerald (2006): open source is transforming from its "free software" origins to a more mainstream and commercially viable approach

What

- OSS is defined by adherence to the OSI definition
- Dominated by operating and networking system software, development tools and infrastructural component
- Examples:
  - Linux operating system
  - Apache web server
  - Perl, Python programming languages
  - V8 javascript engine
  - React, Angular, Vue, Ember++ javascript frameworks

Why

- Three levels of motivations for open source software:
  - Technical
  - Economic
  - Socio-political

Why - technical and economical motivation

- OSS seen as having potential to address "Software crisis" - software taking too long to develop, not working well when delivered, and costing too much
    - Speed - OSS characterised by short development cycles. "Adding manpower to a late software project makes it later" vs "given enough eye-balls, every bug looks shallow".
    - Quality - peer review of source code. Some argue OSS devs are among the most talented and motivated.
    - Cost - shared costs and shared risks of development.

Why - socio-technical motivation

- Motivation of individual developers often socio-technical
- Studies point to "rush" of being able to produce something that get feedback and is used by others
- Meritocracy, where quality of code speaks for itself
- Arena for demonstrating skills for potential employers

- Different in OSS projects where developers are paid

When and Where

- Decentralised geographically - distribution of work
- Rapid evolution with frequent, incremental releases

How

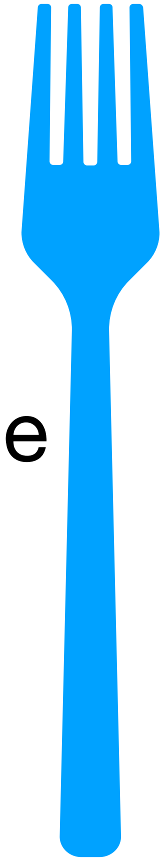
- Classic (early) example:
  - One single or a small group of developers establishes a project and its direction
  - Other developers submit patches to fix bugs or add functionality
  - Examples: apache web server, fetchmail, emacs

How

- Increasingly (OSS 2.0):
  - Companies establish OSS projects as part of a purposeful strategy
  - Developers are paid to contribute
  - Examples:
    - React and Angular largely developed by Facebook and Google
    - Linux kernel top 10 contributors include Intel, Red Hat, Samsung, IBM

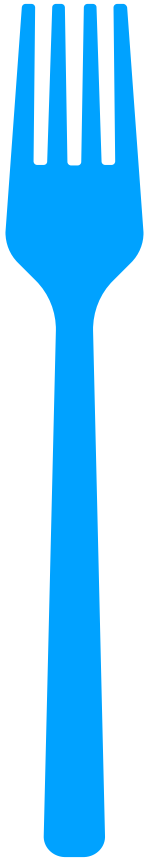
How - forks

- Often no written rules within open source projects - customs and taboos must be learned by experience
- The right to **fork** is central to OSS - making a copy of the source code which is then developed separately
- However, forking is often seen as bad practice



How - forks

- Examples of well-known forks:
  - OpenOffice => LibreOffice
  - KHTML => WebKit => Blink
  - Mambo => Joomla
  - Debian => Ubunutu
  - Visualisation of linux forks



How - infrastructure

- Internet has become the key infrastructure that supports OSS development

- Real-world meetings are seldom, coordination happens in various online tools

- Version control and source code repositories critical (GitHub, sourceforge etc)

Who

- Three key stakeholders on OSS development:
    - Individual developers - often perceived as "hobbyists", but in reality often full-time developers
    - Companies supporting development and distribution
    - Users - experts and early adopters, often the same people who contribute to open source projects

Business models

- Business model: how an organisation creates value.

- Major organisations base their business on OSS - Red Hat, SUSE, Canonical, Apache Foundation, Mozilla, eZ System

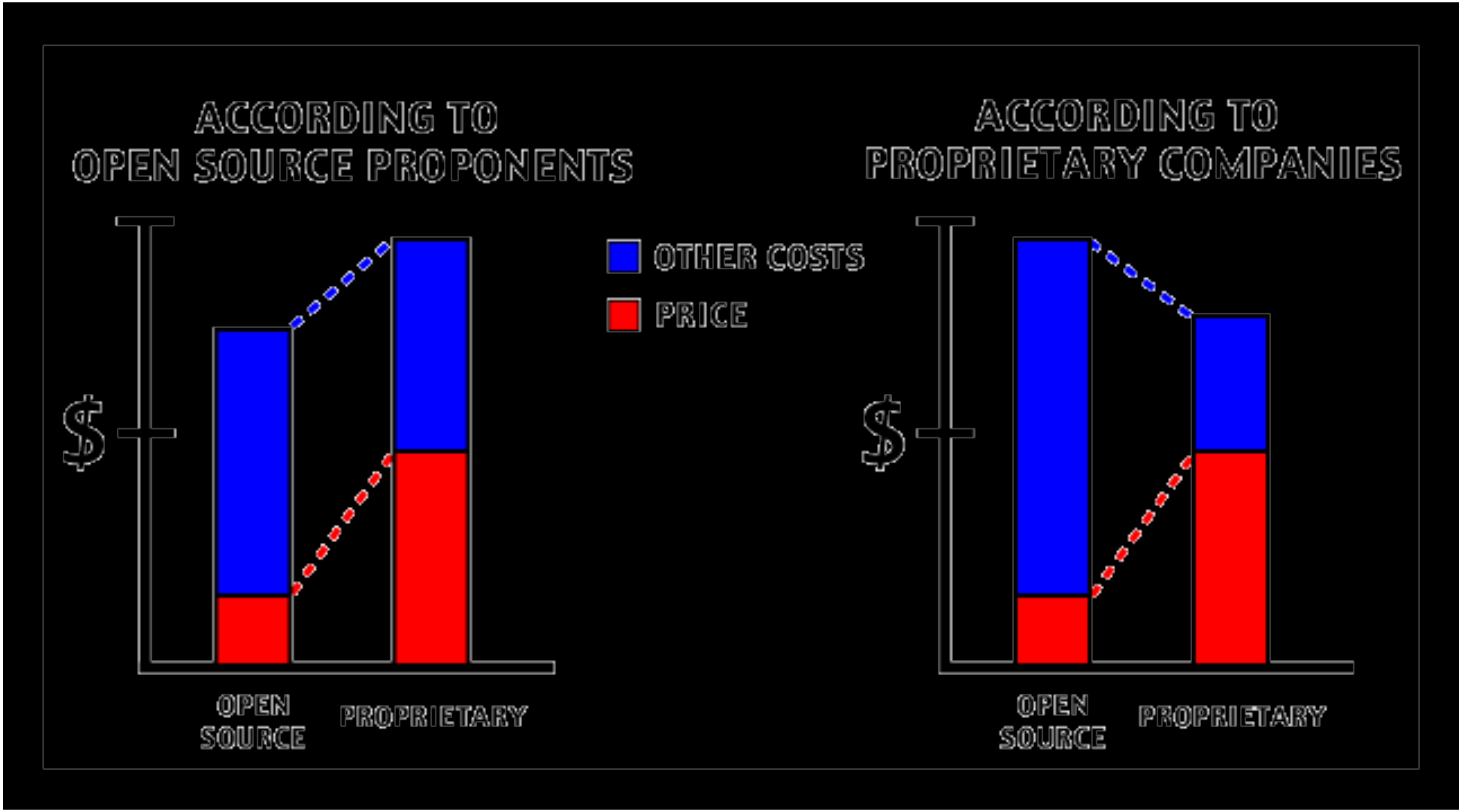
- Other organisations use OSS without having it as a main business - IBM, Google, Apple, Oracle

- Different business models are used

Cost Reduction

- OSS can help reduce cost
- Depends on TCO of OSS vs the alternatives
- Applicable when software sale is not the main revenue
- Example: Sun Microsystems buying the company behind what would become OpenOffice, to reduce licensing cost (and market share) of MS Office. LibreOffice was forked from OpenOffice.org in 2010

Cost Reduction



Source: Northwest Regional Educational Laboratory, Portland, Oregon

Services

- Offering services based on OSS: web hosting, file hosting, infrastructure/platform/software as a service (IaaS/PaaS/SaaS)
- Often combined with the freemium model

> *It's obvious, if we don't support Linux, we'll be Windows only and that's not practical.*
> Mark Russinovich, CTO of Microsoft Azure

- Example: Linode (or other VSP) who provide hosting of servers running different versions of linux

Support and consulting

- Charging for consulting, support and maintenance of OSS
- Configuration of complicated software, providing training etc
- Example: Canonical, who develops the Ubuntu linux distribution and makes money from support and consulting related to it.

Loss-leader

- Providing a product for free or low cost to increase the market and/or attract sales of related products
- Often combined with dual-licensed software
- Examples:
  - IBM open sourcing Eclipse IDE, in order to increase market for related products.
  - MySQL providing open source community edition to drive sales of commercial edition

Other

- Freemium - providing a free basic tier to attract users
- Open core - open sourcing the core product, but with certain parts under a proprietary license
- Hardware - using OSS in hardware products with as routers, TVs etc
- Accessorising - selling accessories related to OSS
- Advertising and search - ads and search engines
- Donations

Process	FOSS	OSS 2.0
Development Life Cycle	• Planning—"an itch worth scratching" • Analysis—part of conventional agreed-upon knowledge in software development • Design—firmly based on principles of modularity to accomplish separation of concerns • Implementation ○ Code ○ Review ○ Pre-commit test ○ Development release ○ Parallel Debugging ○ Production Release (often the planning, analysis, and design phases are done by one person/core group who serve as "a tail-light to follow" in the bazaar)	• Planning—purposive strategies by major players trying to gain competitive advantage • Analysis and design—more complex in spread to vertical domains where business requirements not universally understood • Implementation subphases as with FOSS, but the overall development process becomes *less* bazaar-like • Increasingly, developers being paid to work on open source
Product Domains	• Horizontal infrastructure (operating systems, utilities, compilers, DBMS, web and print servers)	• More visible IS applications in vertical domains
Primary Business Strategies	• Value-added service-enabling • Loss-leader/market-creating	• Value-added service enabling ○ Bootstrapping • Market-creating ○ Loss-leader ○ Dual product/licensing ○ Cost reduction ○ Accessorizing • Leveraging community development • Leveraging the open source brand
Product Support	• Fairly haphazard—much reliance on e-mail lists/bulletin boards, or on support provided by specialized software firms	• Customers willing to pay for a professional, whole-product approach

Example - OSS projects

- https://github.com/vuejs/vue
- https://github.com/nfarina/homebridge
- https://github.com/dhis2
- https://www.openhub.net/p/dhis2
- https://github.com/topics/javascript

Example - OSS in the smartphone market

- Role of OSS in modern smartphones
- Categorise the smartphone value chain as:
  hardware, operating system, system apps, GUI, app store, third party apps, content++
- Comparing Android, iOS, Replicant, Fire OS

Source: Leister and Christophersen (eds), 2015

Android

- Linux core - open source (free)
- System software is partly open source, partly proprietary
- Trademark and Open Handset Alliance used to prevent open source parts of android to make competing products

Replicant and Fire OS

- Replicant developed by FSF. Based on Android, but with only free software

- Fire OS developed by Amazon. Fork of Android, and thus produced by manufacturers outside the OHA

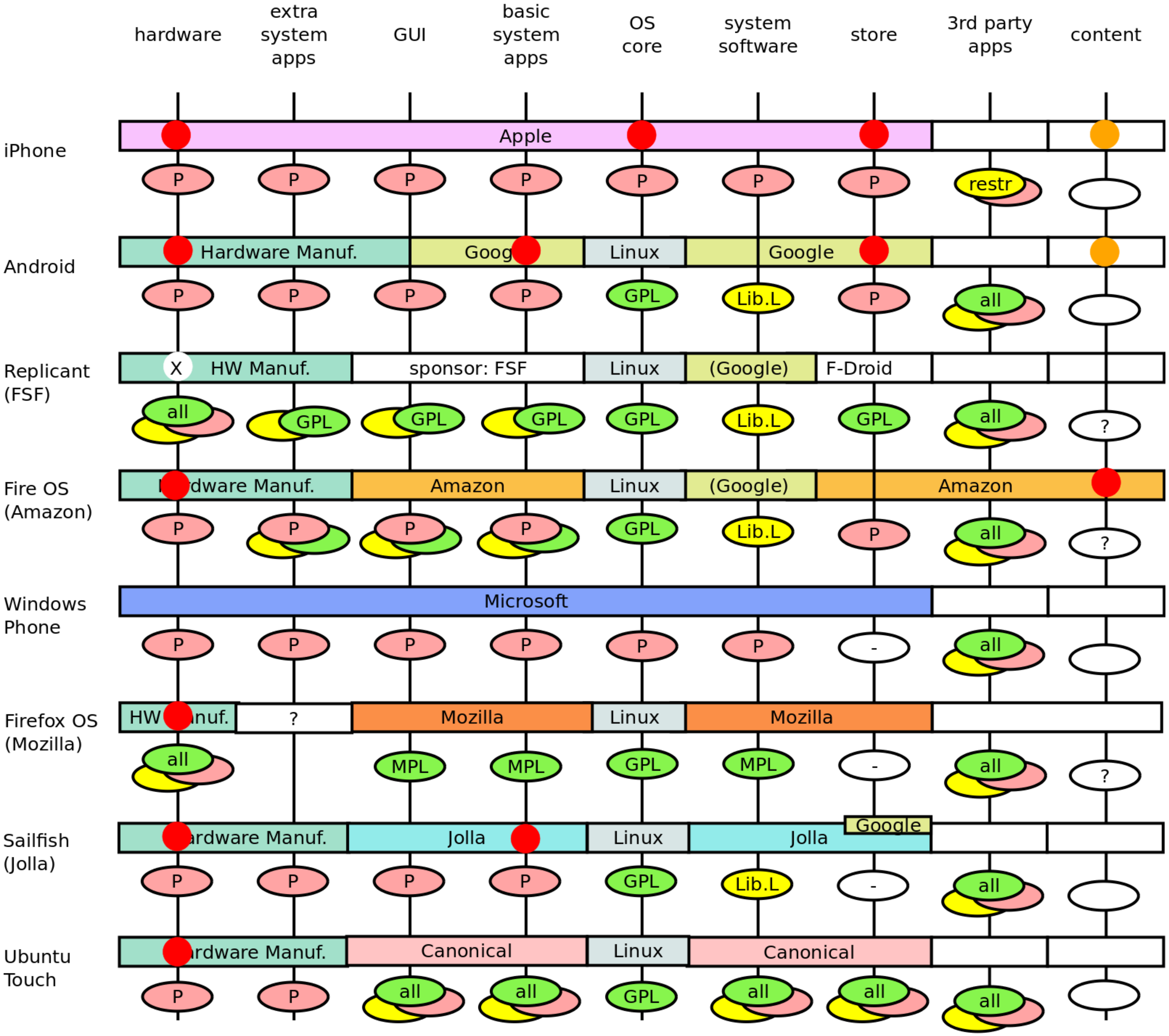
- Contains proprietary code replacing non-open parts of android

- Replicant and Fire OS do not support Google Play store

iOS

- Based initially on BSD-licensed core (UNIX)
- Modifications are proprietary
- *Free* software not allowed in app store (more next week)

Overview



	hardware	extra system apps	GUI	basic system apps	OS core	system software	store	3rd party apps	content
iPhone	●			Apple	●		●		●
	P	P	P	P	P	P	P	restr	
Android	● Hardware Manuf.		Goog	●	Linux	Google	●		●
	P	P	P	P	GPL	Lib.L	P	all	
Replicant (FSF)	X HW Manuf.		sponsor: FSF		Linux	(Google)	F-Droid		
	all	GPL	GPL	GPL	GPL	Lib.L	GPL	all	?
Fire OS (Amazon)	●dware Manuf.		Amazon		Linux	(Google)		Amazon	●
	P	P	P	P	GPL	Lib.L	P	all	?
Windows Phone				Microsoft					
	P	P	P	P	P	P	-	all	
Firefox OS (Mozilla)	HW●nuf.	?	Mozilla		Linux	Mozilla			
	all		MPL	MPL	GPL	MPL	-	all	?
Sailfish (Jolla)	●rdware Manuf.		Jolla	●	Linux	Jolla	Google		
	P	P	P	P	GPL	Lib.L	-	all	
Ubuntu Touch	●rdware Manuf.		Canonical		Linux	Canonical			
	P	P	all	all	GPL	all	all	all	

Sources

- Feller and Fitzgerald, 2000.
- Fitzgerald, 2006.
- Stallman, 2009
- Curriculum articles on Open APIs and Open Source
- Leister and Christophersen (eds), 2015. *Open Source, Open Collaboration and Innovation.* Mainly chapter 4.
- Wikipedia (FOSS)
- Wikipedia (LibreOffice)