

Open Source - State of the Art and Challenges Ahead

Darijus Strašunskas
PhD Trial lecture
2006.03.09 Trondheim



www.ntnu.no

Contents

- Introduction to terminology
- State and Characteristics of OS:
 - Licenses
 - Participants
 - Process
 - Product
- Open (re)source
- Challenges



www.ntnu.no

Free Software vs. Open Source

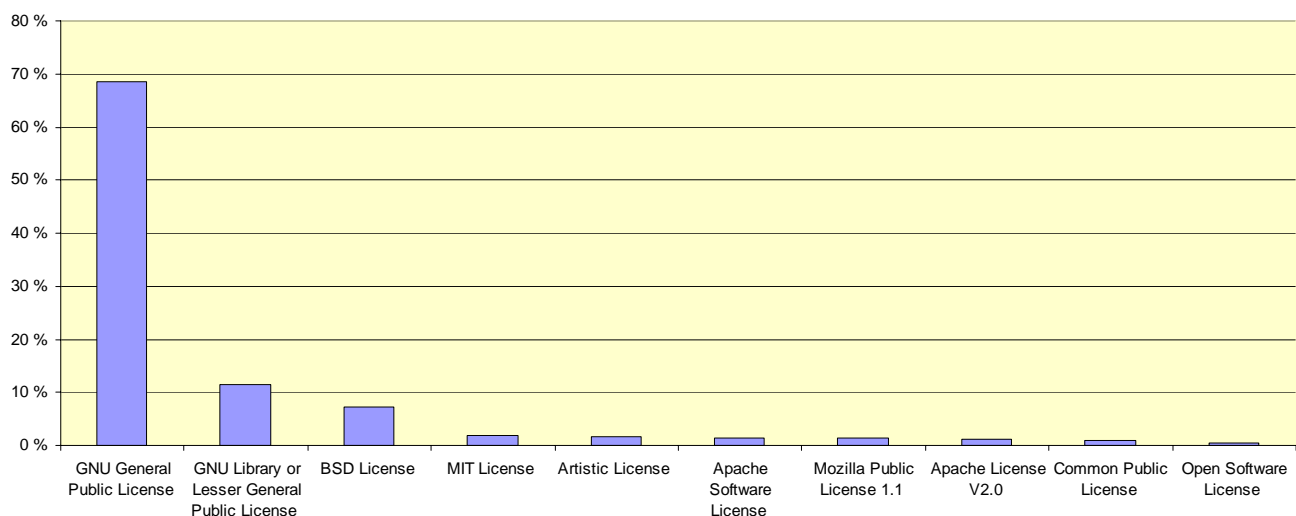
- Free Software - means free and open access to a program's source code.
- Free software / Open Source / Libre software
- FLOSS - Free/Libre Open Source Software
- Free Software as a political philosophy
- Open Source as a development method

Overview of licenses

License	Can be mixed with non-free software	Modifications can be taken private and not returned to author	Can be re-licensed by anyone	Contains special privileges for the original copyright holder over your modifications.
GPL	no	no	no	no
LGPL	yes	no	no	no
BSD	yes	yes	no	no
NPL	yes	yes	no	yes
MPL	yes	yes	no	no
Public Domain	yes	yes	yes	no

Licenses used at SourceForge.net

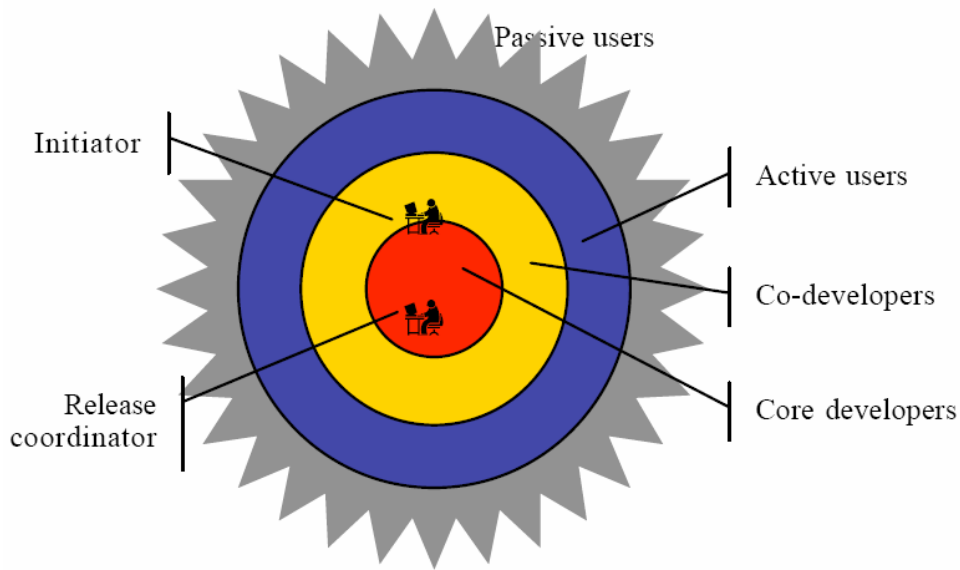
Licenses (Top 10)



Microsoft Shared Code Licenses

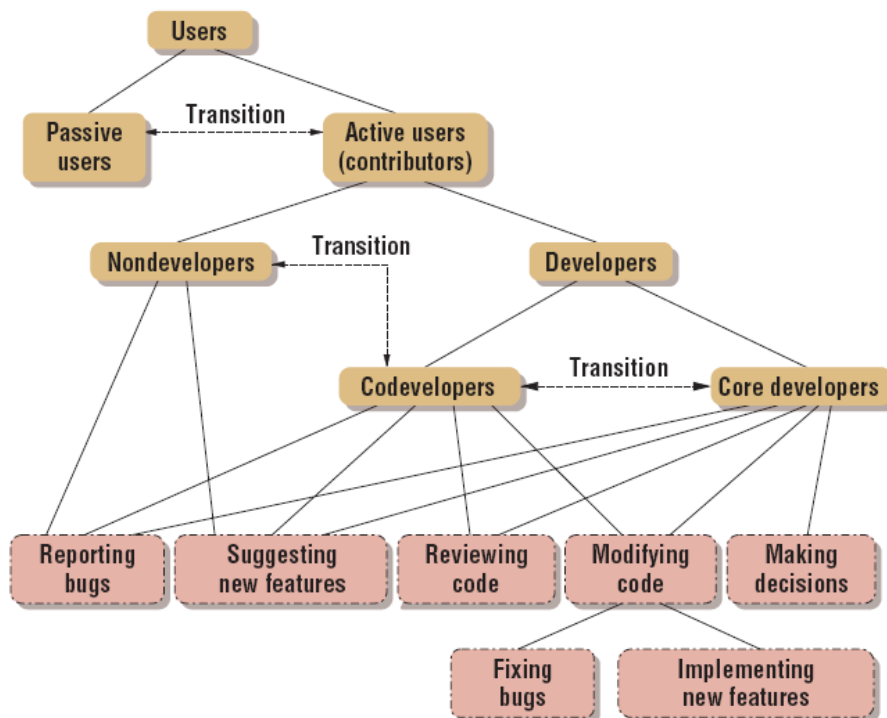
- **Microsoft Permissive License (Ms-PL)** - The Ms-PL is the least restrictive of the Microsoft source code licenses.
- **Microsoft Community License (Ms-CL)** - This type of license is commonly referred to as a reciprocal source code license and carries specific requirements if Ms-CL code is combined with own code.
- **Microsoft Reference License (Ms-RL)** - The Ms-RL is a reference-only license that allows licensees to view source code.

OS community



Source: Crowston et al., 2004

Open Source Users



Source: Gacek and Arief, 2004

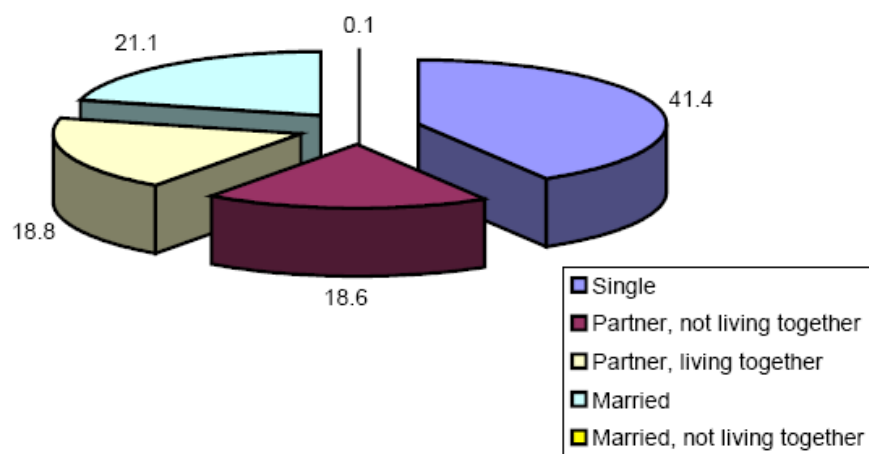
Starting age

- mean age 26-27, median of their starting ages is close to 22.



Source: FLOSS Final Report, 2002

Civil status of OS developer

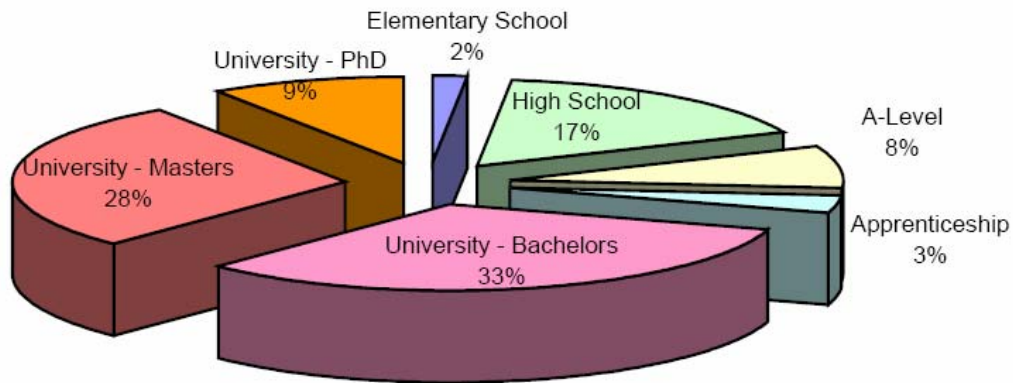


- Male-dominated (98%)

Source: FLOSS Final Report, 2002

Highest level of education

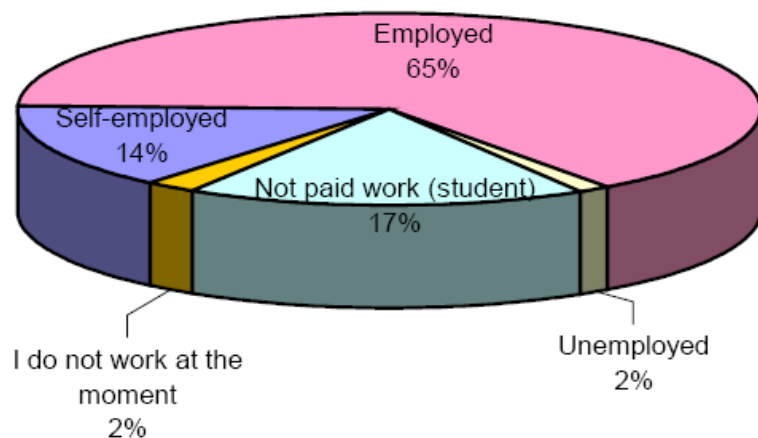
- highly educated



Source: FLOSS Final Report, 2002

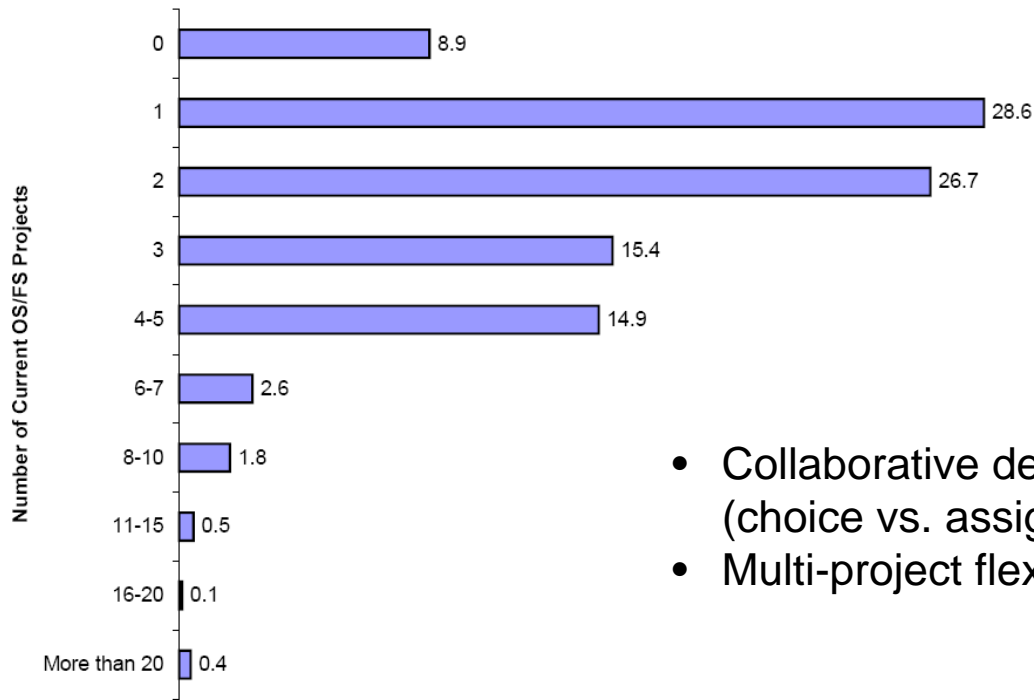
Employment status

- Part-time participation
- Mostly experienced IT professionals
- 50% report having earned money directly or indirectly through their work on FLOSS.



Source: FLOSS Final Report, 2002

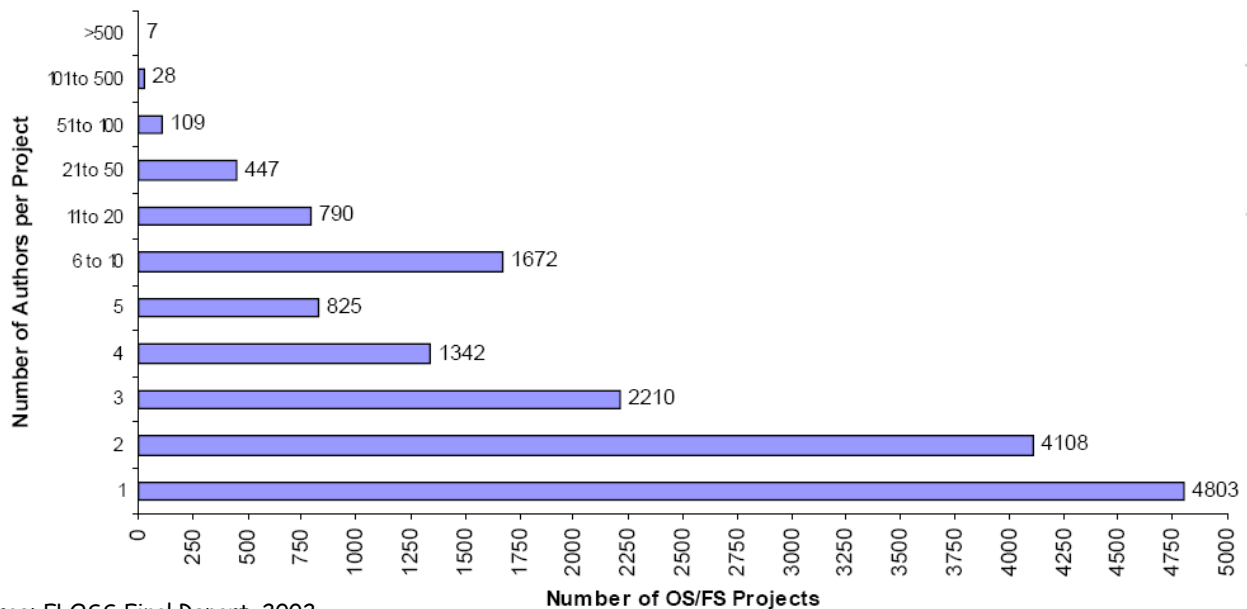
Number of projects developer is involved in



- Collaborative development (choice vs. assignment)
- Multi-project flexibility

OS projects by number of developers

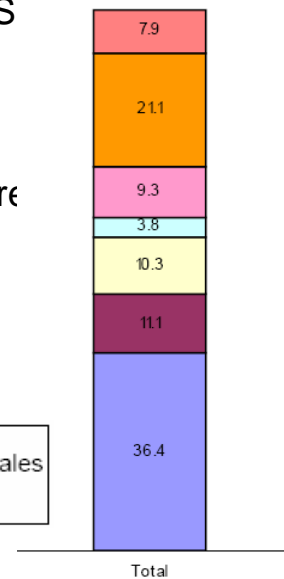
- One man effort
- (partly) distributed



OS projects by occupational background.

Usability problems

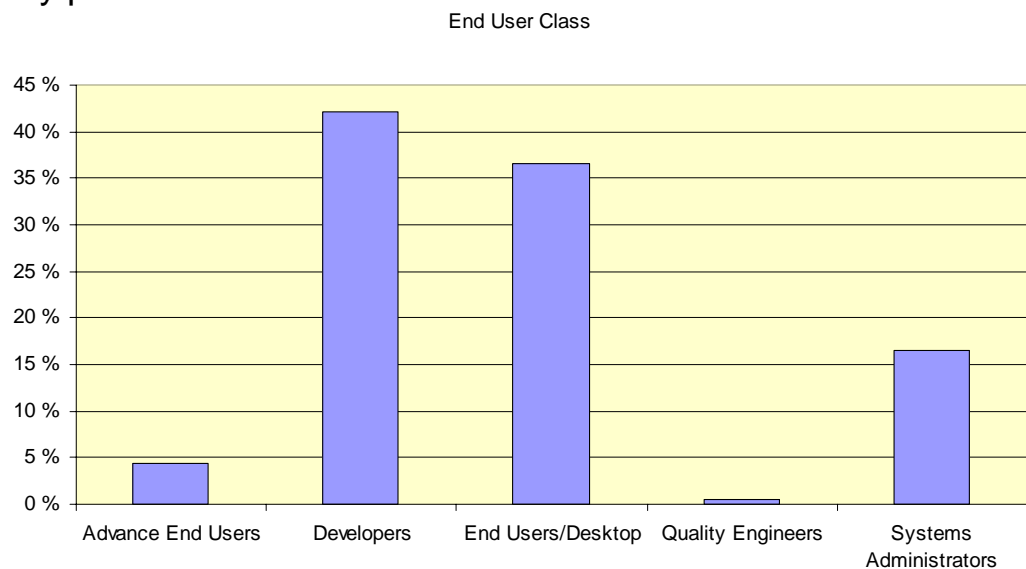
- Usability experts typically do not get involved in OS projects
- Developers are not typical end-users and they do not need a fancy front-end
- Design for usability really ought to take place before any coding, but it doesn't
- Iterative applications keep getting larger and more complicated



Source: FLOSS Final Report, 2002

Intended audience and usability

- Incentives in OS work better for improvement of functionality than usability
- Usability problems are harder to specify and distribute than functionality problems



Motivation

For person

For company

Technological

- Meet personal needs
- Work with leading edge of technology
- Have intellectual stimulation

- Reliability: security, stability and privacy
- Innovation

Socio-political

- Satisfy intrinsic motivation to do work
- Take on a threat, rival, monopoly
- Reputation enhancement
- Ideological urge

- Helping domestic industries and other nationalistic motives
- Ideological

Economic

- Improve programming skills
- Take advantage of low cost and opportunity to participate with nothing to lose.

- Competition
- Independence
- Cost saving

Growth of Complexity

- Games (Adventure – 1977; Rogue – 1980)
- Compilers (gcc – 1987)
- Operating systems (Linux – 1991; 386BSD – 1992)
- Enterprise applications:
 - CMS: eZ publish CMS
 - CRM: SugarCRM
 - ERP: Compiere, ERP5
 - PBX: Asterisk
 - EHR: Vista

OS Development Life-Cycle

- Code, review, pre-commit test, development release, parallel debugging, production release;
- Most time spent in coding, debugging, testing;
- Short, fast, iterative development cycles;
- Coordination:
 - Computer mediated communication (asynchronous communication)
 - Only “beautiful” solutions accepted
 - Email voting to make significant decision
 - Delegating responsibility of the module to specific individuals
- The more mature project – the more formalized the way of working

Traditional	Agile	Open source	
Documentation is emphasized as a means of quality control and as a management tool.	Documentation is deemphasized.	All development artifacts are globally available, including code and information documentation.	Documentation
Business analysts translate users' needs into software requirements.	Users are part of the team.	The developers typically are the users.	Requirements
Developers are assigned to a single project.	Developers are assigned to a single project.	Developers typically work on multiple projects at different levels of involvement.	Staffing model
Peer review is widely accepted but rarely practiced.	Pair programming institutionalizes some peer review.	Peer review is a necessity and is practiced almost universally.	Peer review
Large number of requirements bundled into fewer, infrequent releases.	Release early, release often.	Hierarchy of release types: “nightly,” “development,” and “stable.”	Release schedules
Teams are managed from above.	Teams are self-organized.	Individual contributors set their own paths.	Management
Testing is handled by QA staff, following development activities.	Testing is part of development.	Testing and QA can be performed by all developers.	Testing
Different parts of the codebase are assigned to different people.	Anyone can modify any part of the codebase.	Anyone can modify any part of the codebase, but only committers can make changes official.	Distribution of work

Requirements in OS

- OS requirements come from:
 - Directly from the developers - “scratches an itch”
 - Users of OS software (fetchmail)
 - Implementation of explicit standards (Apache, gcc)
 - Emulation of implicit standards (look-alike)
 - Need to build learning prototypes
- Since end-user is a developer as well
 - Requirements and design take less time to articulate and negotiate (compared with SE projects)

Requirements Engineering in OS

Traditional SE	OS
Requirements elicitation	Assertion of open software requirements
Requirements analysis	Requirements reading, sense-making, and accountability
Requirements specification and modeling	Continually emerging narrative and informal descriptions
Requirements validation	Condensing discourse that hardens and concentrates system functionality and community development
Communicating requirements	Global access to open software webs

Innovations

- 500 most active project from sourceforge.net
ca 3% of total projects; 6106 developers
- All started in 1999-2001, none of the later started
project gained popularity to join most active group.

	New technology	New for a platform	Existing technology
New market	Radical invention (breakthrough) 5 (1.0%)		Marketing innovation 3 (0.6%)
Existing market	Technology modification 4 (0.8%)	Platform modification 52 (10.4%)	No innovation 436 (87.2%)

Source: Klineciewicz, 2005

Innovations. Statistics for different types of projects

Project type	No. of projects	Average no. of developers	Average no. of new feature requests	Average no. of support requests	Average no. of forum messages
All	500	12.21	79.47	207.84	902.13
All (excluding SourceForge.net)	499	12.19	74.08	32.56	895.56
Non-innovative projects	431	12.30	77.98	33.42	921.86
Radical inventions	5	21.40	555.80	17542.20	1317.00
Radical inventions (excluding SourceForge.net)	4	85.00	10.00	39.00	2403.00
Technology modifications	4	38.50	236.25	0.00	0.00
Platform modifications	52	8.79	35.83	31.65	795.23
Marketing innovations	3	7.67	28.67	1.67	228.00

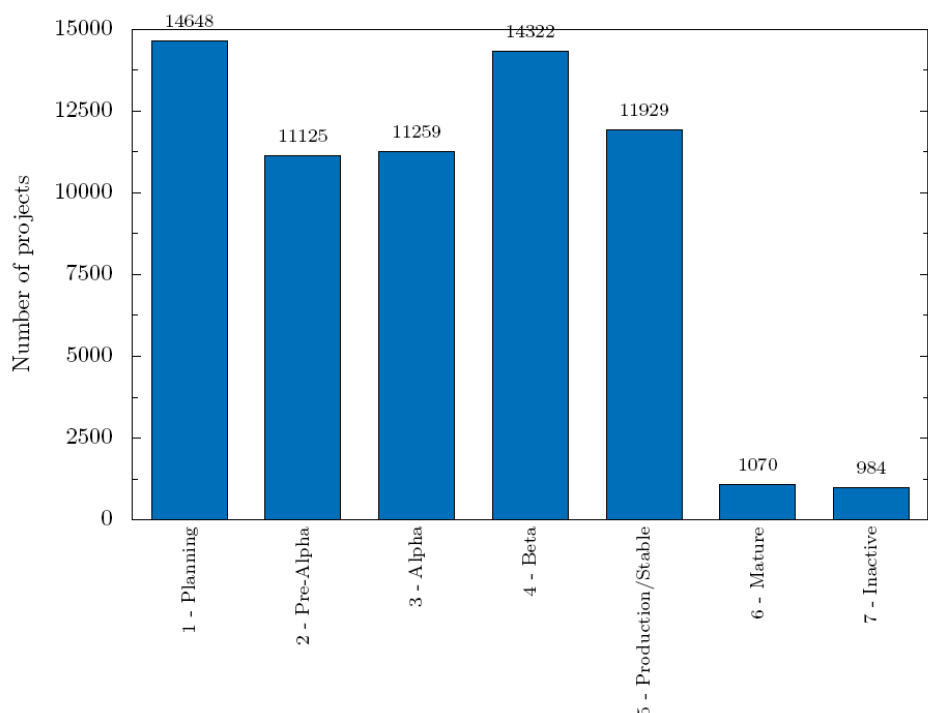
Source: Klineciewicz, 2005

Innovations (cont'd)

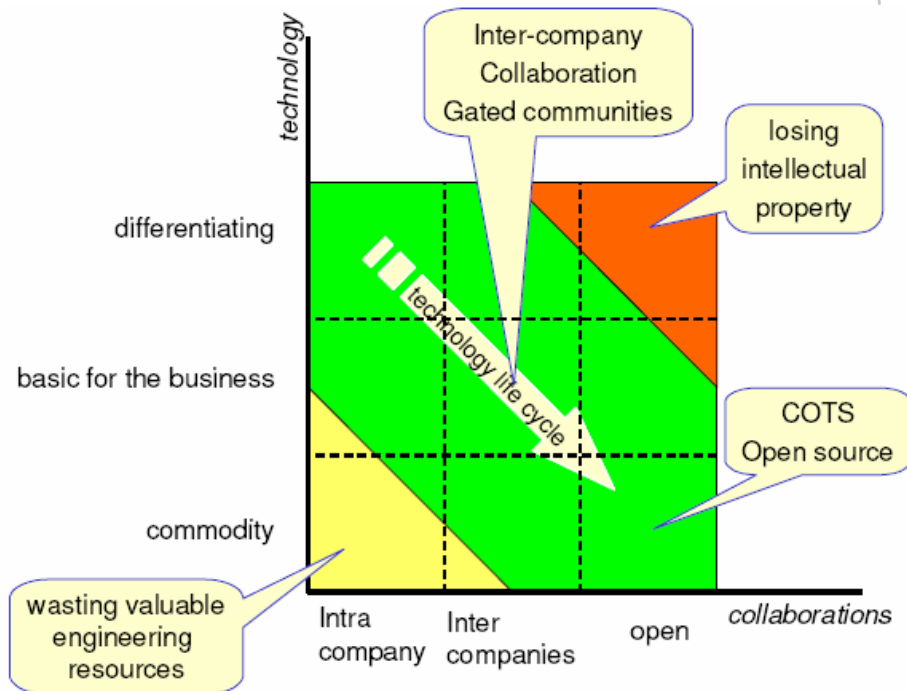
- Multiplicity of projects results from *code base forking*
- Join competing projects is a rare phenomena
- Pioneering new concepts is more difficult than implementing and commoditizing proven designs
- Many novel ideas by university researchers, but narrow research field results in no activity (e.g. Music Miner)
- Innovation requires that the community is open for change, but that is not always the case.
 - 85% of changes performed by a core developers – no space for left for innovation
 - Innovation vs. imitation (Linux, Mono)

Status of OS projects

- Fragile and fallible project
- Non-contractual & unremunerated mode of activity
- 31% of projects registered 1999-2003 are in stages 1 to 3.



Efficacious development



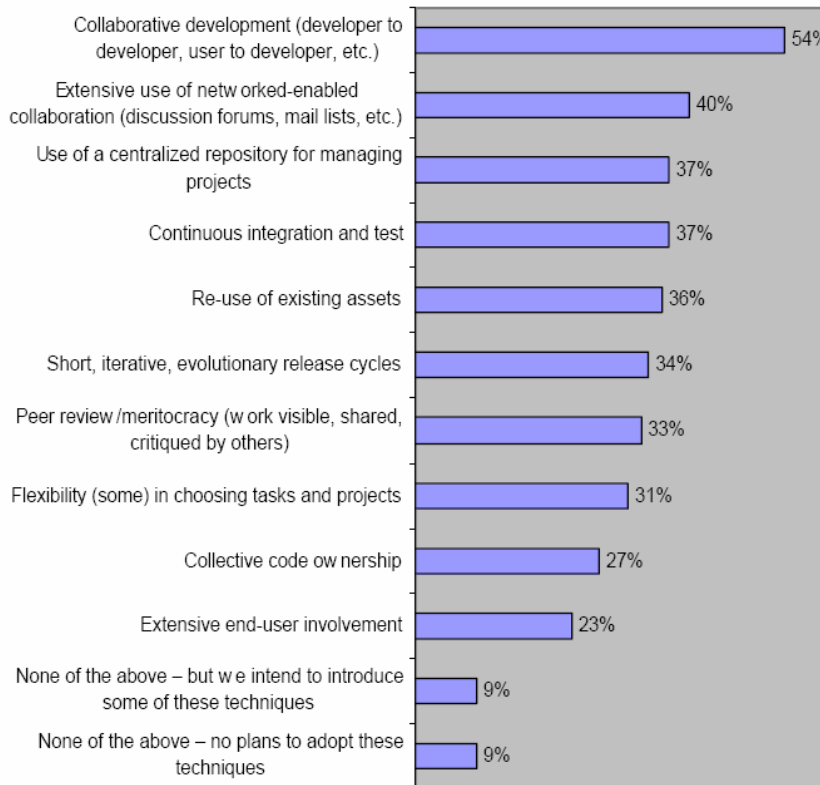
Source: IKT-Norge, 2005

OS adoption as a component

- OS typically is used with very few modification
- Formal selection procedures are seldom used to select components
- System maintainability requires more local resource (debatable in long-term)
- Higher technical skills needed from system administrators

Source: Carbone et al., 2005; Li et al., 2005

OS adoption as a process



Source: VA Software, 2005

Open Source - State of the Art. Summary

- Open Source – the Art to State instructions that both computers (compilers) and humans (software engineers) understand.
- But still it is not a silver bullet.

Towards Information Society

- Open Content

- Del.icio.us (Folksonomies and Webpage annotation)
- OpenDocument Format (ODF)
- Canto Livre (Free/Open Song)
- WikiMedia
- MIT OpenCourseWare
- Content access, copyright term
 - Springer Open Choice License
 - Less strict standard copyright agreements
 - Creative Commons License (Attribution, Commercial, Derivative works, Share alike)
- ClickWorkers - a small NASA experimental project that used public volunteers (clickworkers) for scientific tasks that require human perception and common sense, but not a lot of scientific training.

Challenges

- Manage diversity
- Identification of relevant OSS.
- Quality assessment of relevant OSS.
- Identification of potential / actual mismatches between components.
- Common foundation for development of OS requirements
- New requirements engineering approaches, modeling languages for OS community.
- Open specification as part of open source

- Advancement of programming tools

Challenges (cont'd)

- Open Source policy for companies
- OS not a business model, but a component of a business model.
- Adoption of OS not as software, but as an efficient means to develop and distribute software.
- Digital Rights Management
- Techniques for promote new ideas, gain support
- Effective integration of new members, i.e. boost social learning model

Bibliography

- Ahokas, M. and Laurila, P. Benefiting from Communities in Open Source Business
- Barnett, L. Applying Open Source Processes in Corporate Development Organizations. Forrester Research, Inc. 2004
- Carbone, E., Charpentier, R. and Demers D. Free and Open Source Software (FOSS) Usage in Military Computing. 10th Intl. Command and Control Research and Technology Symposium, The Future of C2. June 2005, Virginia, USA.
- Clickworkers, NASA. <http://clickworkers.arc.nasa.gov/top>
- Crowston, K., Annabi, H., Howison, J. and Masango, C. Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development. Proceedings of ACM WISER, ACM Press. 18-26
- Duijnhouwer, F.-W. and Widdows, C. Open Source Maturity Model. August 2003.
- Dahlander, L. and Magnusson, M. G. Relationships between open source software companies and communities: Observations from Nordic firms. Research Policy 34(4), Elsevier, 2005, 481-493.
- Divitini, M., Jaccheri, L.M., Monteiro, E. and Traetteberg, H. Open source processes: no place for politics? Proceedings of Workshop on Open Source co-located with ICSE 2003.
- Ducheneaut, N. Socialization in an Open Source Software Community: A Socio-Technical Analysis. Computer Supported Cooperative Work (CSCW) 14(4), Springer, 2005, 323-368
- Gacek, C. and Arief, B. The many meanings of open source. IEEE Software 21(1), pp.34-40, 2004.
- Giacomo, P. Di. COTS and Open Source Software Components: Are they really different on the battlefield? Proceedings of ICCBSS 2005, LNCS 3412, Springer-Verlag, 301-310, 2005.
- Gregorio Robles, Juan Jose Amor, Jesus M. Gonzalez-Barahona, Israel Herraiz, "Evolution and Growth in Large Libre Software Projects," iwps, pp. 165-174, Eighth International Workshop on Principles of Software Evolution (IWPSE'05), 2005.
- IKT-Norge. Norsk COSI - Project description. 2005, 15p.

Bibliography (cont'd)

- Jensen., B.B., Lyngshede, S., and Søndergaard, D. A Quality Definition for Open Source Software. In Proceedings of 2nd Workshop on Software Quality (co-located with ICSE 2004).
- Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Christian Bunse, Umair Khan, Marco Torchiano and Maurizio Morisio: "Validation of New Theses on Off-The-Shelf Component Based Development." Proceeding of the 11th IEEE International Metrics Symposium (Metrics'05).
- Jorgensen, N. Putting it all in a trunk: incremental development of FreeBSD open source project. Information Systems Journal 11, 2001, 321-336
- Klinecicz, K. Innovativeness of open source software projects. 2005. URL: <http://opensource.mit.edu/papers/klinecicz.pdf>
- Levesque, M. Fundamental issues with open source software development. First Monday, 9(4), 2004
- Nichols, D.M. and Twidale, M.B. The usability of open source software. First Monday, 8(1), 2003
- Micorsoft. Shared Source Licenses. <http://www.microsoft.com/resources/sharedsource/licensingbasics/sharedsourcelicenses.mspx> , 2005.
- Scacchi, W. Understanding the Requirements for Developing Open Source Software Systems. IEE Proceedings--Software, 149(1), 24-39, February 2002.
- VA Software. Application development and open source process trends: Survey analysis and finding. White paper, VA Software, 2005.
- Weiss, D. A large crawl and quantitative analysis of open source projects hosted on sourceforge. Institute of Computing Science, Poznan University of Technology, Poland," Research Report RA-001/05, 2005
- Wynants, M. and Cornelis, J. (Eds.) How Open is the Future? Economic, Social & Cultural Scenarios inspired by Free & Open-Source Software. VUB Brussels University Press. 2005.

Open Source - State of the Art and Challenges Ahead

Darijus Strašunskas
PhD Trial lecture
2006.03.09 Trondheim

Copyright and patent

- Software innovations occur rapidly and can be made without a substantial capital investment
- Tend to be creative combination of previously-known techniques.
- Copyright law was originally designed “for the encouragement of learning” and “to promote the progress of science and useful arts”. Authors received a protection against unauthorised copying and plagiarism for a limited period, before their works were passed into the public domain.
- In recent decades there has been a shift from the right to protect a work against unauthorised “use” towards a right to prevent others from unauthorised access to it.