Open Source Development

Timo Savola

<timo.savola@evtek.fi>

EVTEK Institute of Technology

March 24, 2006

Mid Sweden University, Sundsvall

- Process is open
- Software is free (libre)

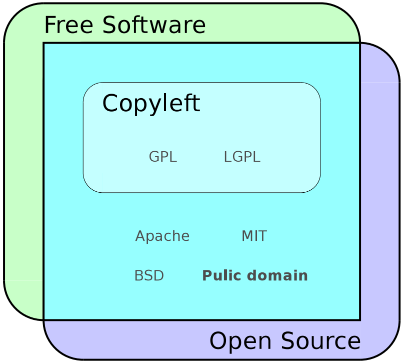
Benefits

- Efficiency
  - Development does not depend on one or a few parties
  - Pace and direction
  - Continuity
- Quality
  - Transparent process
  - Transparent results
  - Everyone can fix problems
- Knowledge
  - More experts
  - New application areas
- Standardization

Problems

- Controversial
  - Contradicts with traditional business practices
  - Access to information is not exclusive
- Legal issues
  - Who is responsible for code written outside of the company?
  - Does the code violate patents?
  - Has the company licensed parts of the code from other companies?
  - What is expected of a company that modifies software with certain license terms?

License types



- Free Software Foundation (`www.fsf.org`)
- Open Source Initiative (`www.opensource.org`)

Business models

- Dual licensing
  - The program is published under a closed and a **copyleft** license
  - A single company holds the copyright for the whole code base
- Products
  - Closed programs built on open programs
  - Embedded systems (standard software, special hardware)
- Services
  - Sell programming instead of programs
  - Various kinds of support

Project categorization

1. Dead
   a. Unfinished (out of motivation, time or funding)
   b. Finished (implements a standard, bugs not found often)
2. Personal
   - At an early stage or uninteresting topic
   - Small project; no need for many developers
3. Company
   - A company opens up their product
   - The company wants to continue overseeing the development
4. Community
   - Most active/interesting projects
   - Development is overseen by the founder, a group of developers or a foundation

Evolutionary development

- Free software tends to evolve
  1. Personal projects
  2. Community projects
- Form of iterative development
  - Software is built one function(ality) at a time
  - Proceeds in the order which pleases the developer
- Practical results
  - Software is in real use between iterations

Exploiting free software in products

- Find suitable projects
  - Not in early stages of development
  - Active developer community
  - The license allows for possible closed extensions
- Evaluate/create prototypes
  - Free software is not always well understood
  - You might not know what you get before you apply it to your problem
  - Never used in your application area?
  - No objective information about performance?
- Reuse as much as possible without modification
- Do the rest yourself
  - Generic changes to existing projects benefit the community
  - Special features of the product might be kept closed

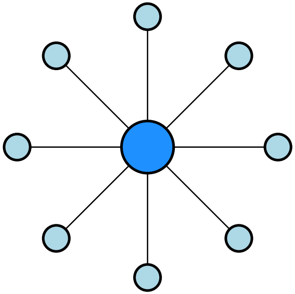
Developing with the community

- Communicate with existing projects
  1. Make changes to the upstream version
  2. Adapt your changes to the wishes of the community
  3. Continue working with the updated upstream version
- Try to create a community around your projects
  - Spin-off projects from your product development program
- ROI
  - If community accepts your work, they will maintain it for you

Quality assurance

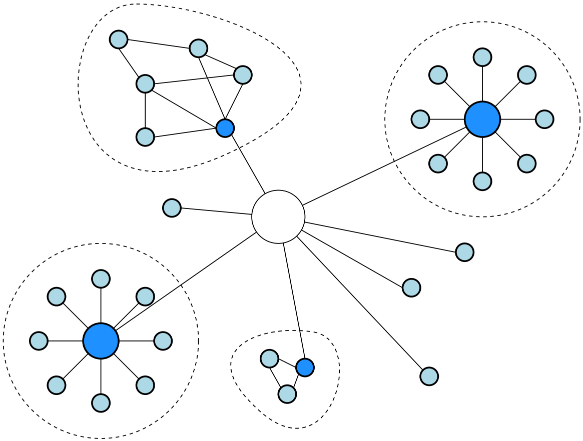
- Developer and user community
  - Lots of potential testers and reviewers
  - Disorganized
- Only you are responsible for testing
  - Input from the community
  - Output to your customers
- “More eyeballs find more bugs”
  - Might or might not be true for your project

Traditional project organization



- Centralized project management
  - Communication
  - Code
- Tools support this way of working

Open project organization



- Separate teams develop different features
- Companies want to maintain their custom versions

Challenges

- Branching development
  - How are the branches related?
  - How do you merge the branches again?
- External developers
  - How and where do you submit improvements?
  - Are there first- and second-class developers?
- Quality assurance
  - Are fixes distributed to all branches?
  - Can you easily release a new version of a production branch?
- Tools should not dictate the work flow
  - Revision control system

Decentralized version control

- Branches can be located at different physical sites
  - Private branches are compatible with the mainline
- Separate development teams have own branches
  - Used like a centralized revision control system
  - Finished work can either be merged to mainline or maintained separately
- Each developer has her own branch
  - A work-in-progress version of a particular feature
  - The finished feature is merged to the team's branch
  - Fine-grained revision history
  - Automatic backups
- Contribution is easy
  - Everyone makes changes using the same tools
  - Project leader approves or refuses contributed changes

...

?