Software Models

Henning Schulzrinne

Introduction

- Non-technical interlude...
- Many more type of software than before:
  - commercial projects
  - contract
  - shareware
  - open source
- Influences:
  - commercial needs
  - IP protection & licensing
  - documentation
  - support

Commercial

- Classical model, but many variations
- Internal vs. for sale
  - e.g., Columbia SSOL vs. PeopleSoft
- One time vs. retail
  - only run one (or few copies)
    - custom programming
    - missile control
  - retail aka "shrink wrap", but also download or "ship and enable"
  - games: cartridges
  - embedded – OS, peripheral, cell phones, industrial control, ...

Commercial software

- Many different licenses:
  - single computer (Microsoft XP)
  - multiple computers by same owner (e.g., laptop and home) – common for consumer software
  - IP address (servers)
  - *floating license* – "no more than 3 simultaneous users" (CAD and SE tools)
  - single network (e.g., within Columbia) – *site license*

Contract

- Consulting – individually or through consulting houses, e.g., ArthurAndersen, EDS or Wipro (India)
- "Work for hire" – output belongs to customer (GRA vs. undergraduate!)
- Usually, specific to customer
- Often, outside of core competency of client (e.g., gov't, hospital, ...)
- Billed by project or by hour

Shareware

- **not** = open source
- can freely download binaries
- usually low-cost, limited functionality items (e.g., create buttons for web pages)
- "crippleware": limited functionality (can't save work) or time limit on usage
- mostly Mac and Windows

Open source

www.opensource.org

- Free redistribution
- Source code
  - distribute either unobfuscated source code or binaries
- Derived works
  - same license as original code
- Integrity of author's source code
  - allow distribution of modified code

Open Source

- No discrimination against persons or groups
- No discrimination against fields of endeavor
  - commercial use, non-military use
- Not specific to product
  - not just if part of some distribution
- Must not restrict other software
  - can't require other software to be open-source

Open Source

- Examples of open-source licenses:
  - GNU General Public License
    - modifications are also GPL
  - GNU Library (Lesser) Public License
    - allow linking into non-GPL code
  - BSD license
    - least restrictive – allow commercial use
  - Artistic license
    - force new naming for modifications

Almost open-source

- Lots of programs that are free and/or provide source code, but:
  - don't allow modification (TeX)
  - non-commercial or educational use only
  - limited to particular hardware
  - may be patent-encumbered (e.g., audio codecs)
  - only for customer (e.g., in case supplier goes bankrupt)

Example: BSD license

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, ...

Open source arguments

- "The Cathedral and the Bazaar" (Eric Raymond)
- Social arguments
  - knowledge dissemination
  - avoid wasted effort ("doing it anyway")
  - cf. *pro bono* by lawyers
  - skill strengthening and visibility (academics)
- Technical arguments
  - quality
  - security
  - standards-compliance
  - testing on many systems

Cathedral and the Bazaar

1. Every good work of software starts by scratching a developer's personal itch.
2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
3. ``Plan to throw one away; you will, anyhow.'' (Fred Brooks, ``The Mythical Man-Month'', Chapter 11)
4. If you have the right attitude, interesting problems will find you.

Cathedral and the Bazaar

5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
7. Release early. Release often. And listen to your customers.
8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.
9. Smart data structures and dumb code works a lot better than the other way around.
10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.

Cathedral and the Bazaar

11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
12. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
13. ``Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away.''
14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.
15. When writing gateway software of any kind, take pains to disturb the data stream as little as possible.
16. When your language is nowhere near Turing-complete, syntactic sugar can be your friend.
17. A security system is only as secure as its secret. Beware of pseudo-secrets.

Open source successes

- TeX (not quite open-source)
- gcc compiler and tools
- Linux and *BSD (FreeBSD, OpenBSD, NetBSD) operating systems
- Apache web server
- sendmail mail transport agent (MTA)
- gimp bitmap graphics tool
- Tcl, Perl, Python language implementations

Open source - problems

- Ideal: small group of authors write code, thousands contribute bug fixes (not just bug reports!) and enhancements
- Need large community of technical users
  - less likely to work well for recipe management or kid's games
  - more forgiving of lack of documentation or clean install
  - can integrate different tools
  - if not, author becomes tech support

Open source - problems

- Thus, works less well for specialized programs with small user base (hundreds) – get mostly complaints
- Mostly individuals moon-lighting or paid by unrelated company (e.g., Linus Torvalds)
- Business model uncertain:
  - packaging (Red Hat)
  - tech support contract (MySQL)
  - free with hardware (IBM)
  - grants and donations

Which model is "right"?

- Commercial needs:
  - size of customer base
  - who pays?
- Intellectual property issues
  - are there patents?
  - does employer rely on IPR (trade or sell)?
- Documentation
  - how much documentation is needed to make the program useful?
- How much support is needed?
  - open-source support: mean vs. variance

"Levels" of programming

- Programmable applications
  - add macro to Excel
- Code generators
  - generate UI code from "wizard"
- Application embedding
  - new UI for IE or Mozilla
- Web scripting
  - no UI in normal sense
- "Classical" application
  - Java, C
  - use mostly OS and UI APIs

"Levels" of programming

- Device drivers and OS kernel
- embedded systems
  - specialty OS (e.g., VxWorks) or no OS\
  - often, lower-end processors
    - no virtual memory
    - no cache
    - limited memory space
    - special operations (e.g., DSP MAC)
- Specialty programming:
  - language compilers