LBSC 690: Week 13

Building and Deploying Systems

Jimmy Lin
College of Information Studies
University of Maryland

Monday, April 30, 2007

Agenda

- Building and deploying systems
- The open source model
- Issues in intellectual property management

The System Life Cycle

- Systems analysis
  - How do we know what kind of system to build?
  - How do we discern and satisfy user needs?
- Development models
  - How do we build it?
- Management
  - How do we use it?

Systems Analysis

- Understand the problem
  - What's the user task?
  - What's the environment?
- Evaluate available technology
- Only then can you design a solution

User-Centered Design

- As opposed to what?
- Understanding user needs
  - Who are the present and future users?
  - How can you understand their needs?
- Understanding the use context
  - How does the particular need relate to broader user activities?
  - How does software fit into the picture?

Information Flows

- Where does information originate?
  - Might come from multiple sources
  - Feedback loops may have no identifiable source
- Which parts should be automated?
  - Some things are easier to do without computers
- Which automated parts should be integrated?
- What other systems are involved?
  - And what information do they contain?
- What are the impacts?
  - e.g., serials use impacts cancellation policy
  - e.g., circulation policy impacts fines

Formal Analysis

system analysis
development models
management

- Process Modeling
  - Structured analysis and design
  - Entity-relationship diagrams
  - Data-flow diagrams
- Object Modeling
  - Object-oriented analysis and design
  - Unified Modeling Language (UML)

UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

Some Library Activities

system analysis
development models
management

- Acquisition
- Cataloging
- Reference
  - Online Public Access Catalog (OPAC)
- Circulation, interlibrary loan, reserves
- Recall, fines, …
- Budget, facilities schedules, payroll, ...

UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

Discussion Point

system analysis
development models
management

- Integrated Library System
  - What functions should be integrated?
  - What are the key data flows?
  - Which of those should be automated?

UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

The Waterfall Model

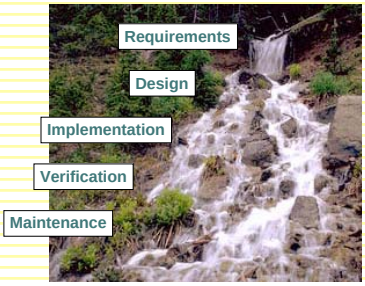
system analysis
development models
management

- Key insight: upfront investment in design
  - An hour of design can save a week of debugging!
- Five stages:
  - Requirements: figure out what the software is supposed to do
  - Design: figure out how the software will accomplish the tasks
  - Implementation: actually build the software
  - Verification: makes sure that it works
  - Maintenance: makes sure that it keeps working

UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

The Waterfall Model

system analysis
development models
management



UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

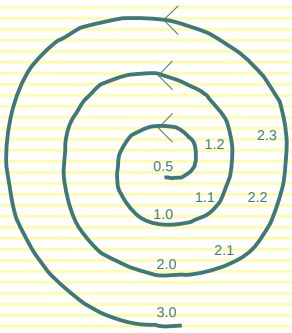
The Spiral Model

system analysis
development models
management

- Build what you think you need
  - Perhaps using the waterfall model
- Get a few users to help you debug it
  - First an “alpha” release, then a “beta” release
- Release it as a product (version 1.0)
  - Make small changes as needed (1.1, 1.2, ….)
- Save big changes for a major new release
  - Often based on a total redesign (2.0, 3.0, …)

UNIVERSITY OF MARYLAND COLLEGE of INFORMATION STUDIES

The Spiral Model



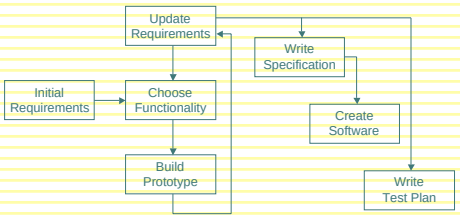
Unpleasant Realities

- The waterfall model doesn't work well
  - Requirements usually incomplete or incorrect
- The spiral model is expensive
  - Redesign leads to recoding and retesting

A Hybrid Model

- Goal: explore requirements
  - Without building the complete product
- Start with part of the functionality
  - That will (hopefully) yield significant insight
- Build a prototype
  - Focus on core functionality, not in efficiency
- Use the prototype to refine the requirements
- Repeat the process, expanding functionality

A Hybrid Model



Requirements

- Availability
  - Mean Time Between Failures (MTBF)
  - Mean Time To Repair (MTTR)
- Capacity
  - Number of users (typical and maximum)
  - Response time
- Flexibility
  - Upgrade path
  - Interoperability with other applications

Different Components

- Off-the-shelf applications vs. custom-developed
- "Best-of-breed" vs. integrated system

Different Architectures

- Desktop applications
  - What we normally think of as software
- Batch processing (e.g., recall notices)
  - Save it up and do it all at once
- Timesharing (e.g., OPAC)
  - Everyone uses the same machine
- Client-Server (e.g., Web)
  - Some functions done centrally, others locally
- Peer-to-Peer (e.g., Kazaa)
  - All data and computation is distributed

Management Issues

- Retrospective conversion
  - Moving from "legacy systems"
  - Even converting electronic information is expensive!
- Management information
  - Peak capacity evaluation, audit trails, etc.
  - Sometimes costs more to collect than it is worth!
  - Sometimes easy to collect, difficult to analyze
- Training
  - Staff, end users
- Privacy

Things will go wrong...

- No software is defect-free. Why?
  - Sheer size: e.g., Windows XP (in 2002) was ~40M lines of code
  - Almost impossible to predict all possible use contexts
  - Concurrency
- The importance of disaster recovery
  - Backups (periodicity, storage location)

TCO

- TCO = "Total cost of ownership"
- Buying/developing software isn't the only cost!
- Other (hidden) costs:
  - Planning, installation, integration
  - Disruption and migration
  - Ongoing support and maintenance
  - Training (of staff and end users)

What is open source?

- Proprietary vs. open source software
- Open source used to be a crackpot idea:
  - Bill Gates on Linux (3/24/1999): "I don't really think in the commercial market, we'll see it in any significant way."
  - MS 10-Q quarterly filing (1/31/2004): "The popularization of the open source movement continues to pose a significant challenge to the company's business model"
- Open source…
  - For tree hugging hippies?
  - Make love, not war?

Basic Definitions

- What is a program?

  An organized list of instructions that, when executed, causes the computer to behave in a predetermined manner. Like a recipe.

- What is source code?

  Program instructions in their original, *human-readable* form.

- What is object/executable code (binaries)?

  Program instructions in a form that can be directly executed by a computer. A *compiler* takes source code and generates executable code.

Proprietary Software

- Distribution in machine-readable binaries only
- Payment for a license
  - Grants certain usage rights
  - Restrictions on copying, further distribution, modification
- Analogy: buying a car…
  - With the hood welded shut
  - That only you can drive
  - That you can't change the rims on

Open Source Principles

- Free distribution and redistribution
  - "Free as in speech, not as in beer"

  "The license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require royalty or other fee for such sale."
- Source code availability

  "The program must include source code, and must allow distribution in source code as well as compiled form".
- Provisions for derived works

  "The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software."

Open Source vs. Proprietary

- Who gets the idea to develop the software?
- Who actually develops the software?
- How much does it cost?
- Who can make changes?

Open Source is already here…

- Apache web server has ~70% market share of the public Internet (Nov., 2005)
- Sendmail mail server has ~50% market share
- Linux is a very popular OS for servers
  - Sales figures unreliable
- Lots more…

Examples

	Proprietary	Open Source
Operating system	Windows XP	Linux
Office suite	Microsoft Office	OpenOffice
Image editor	Photoshop	GIMP
Web browser	Internet Explorer	Mozilla
Web server	IIS	Apache
Database	Oracle	MySQL

Server vs. Desktop

- Open source has made significant inroads in the server market
- The next big challenge: the desktop market

Open Source: Pros

- Peer-reviewed code
- Dynamic community
- Iterative releases, rapid bug fixes
- Released by engineers, not marketing people
- High quality
- No vendor lock-in
- Simplified licensed management

Pros in Detail

- Peer-reviewed code
  - Everyone gets to inspect the code
  - More eyes, fewer bugs
- Dynamic community
  - Community consists of coders, testers, debuggers, users, etc.
  - Any person can have multiple roles
  - Both volunteers and paid by companies
  - Volunteers are highly-motivated to work on something that interests them

Pros in Detail

- Iterative releases, rapid bug fixes
  - Anyone can fix bugs
  - Bugs rapidly fixed when found
  - Distribution of “patches”
- Released by engineers, not marketing people
  - Stable versions ready only when they really are ready
  - Not dictated by marketing deadlines
- High quality

Pros in Detail

- No vendor lock-in
  - Lock in: dependence on a specific program from a specific vendor
  - Putting content in MS Word ties you to Microsoft forever
  - Open formats: can use a variety of systems
- Simplified licensed management
  - Can install any number of copies
  - No risk of illegal copies or license audits
  - No anti-piracy measures (e.g. CD keys, product activation)
  - No need to pay for perpetual upgrades
  - Doesn't eliminate software management, of course

Cons of Open Source

- Dead-end software
- Fragmentation
- Developed by engineers, often for engineers
- Community development model
- Inability to point fingers

Cons in Detail

- Dead-end software
  - Development depends on community dynamics: What happens when the community loses interest?
  - How is this different from the vendor dropping support for a product? At least the source code is available
- Fragmentation
  - Code might “fork” into multiple versions: incompatibilities develop
  - In practice, rarely happens

Cons in Detail

- Developed by engineers, often for engineers
  - My favorite "pet feature"
  - Engineers are not your typical users!
- Community development model
  - Cannot simply dictate the development process
  - Must build consensus and support within the community
- Inability to point fingers
  - Who do you call up and yell at when things go wrong?
  - Buy a support contract from a vendor!

Open Source Business Models

- Support Sellers ("Give Away the Recipe, Open A Restaurant")

  Give away the software, but sell distribution, branding, and after-sale service.
- Loss Leader

  Give away the software as a loss-leader and market positioner for closed software.
- Widget Frosting

  If you're in the hardware business, giving away software doesn't hurt you and has it's advantages. What are they?
- Accessorizing

  Sell accessories: books, compatible hardware, complete systems with open-source software pre-installed. (open-source T-shirts, coffee mugs, Linux penguin dolls, etc.)

Mature enough? Yes

- Some open source software have been around for 15+ years
- Lots of servers already running open source software

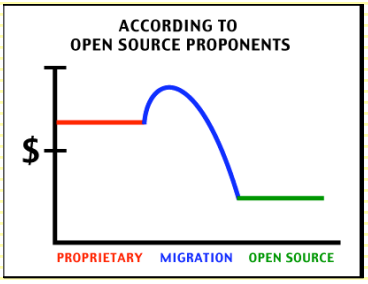
Sustainable? Yes

- Businesses and governments are choosing open source
- Software companies are creating are supporting open source (e.g. IBM, Sun, HP)
- Many schools are considering or adopting open source software

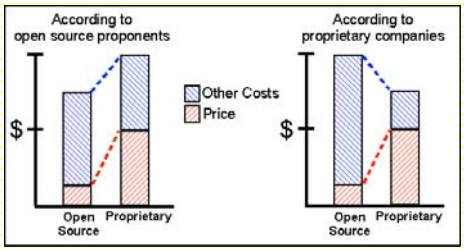
Open Source in Government

- Freedom of Information Act – free, open access to public records
  - What are the implications of using a proprietary format?
- Proposal in Massachusetts – all government documents must be created and saved in open source programs.
- U.S. Office of Management and Budget – On July 1, 2004, officially recognized Open Source software as a viable option for civilian agencies of the federal government
- Open source gaining traction internationally

It comes down to cost...



The TCO Debate



Is open source right for you?

- Do you have access to the necessary expertise?
- Do you have buy-in from the stakeholders?
- Are you willing to retool your processes?
- Are you willing to retrain staff and users?
- Are you prepared for a period of disruption?
- Do you have a well-thought out plan for rolling out open source software?

DRM and DMCA

- DRM = Digital Rights Management
  - Access control
  - Copy control
- DMCA = Digital Millennium Copyright Act
  - A prohibition on circumventing access controls
  - An access control circumvention device ban (sometimes called the "trafficking" ban)
  - A copyright protection circumvention device ban
  - A prohibition on the removal of copyright management information

"Chilling Effect"

- Rights negotiation replaced with terms dictated unilaterally
- Exceptions are very narrow
  - Libraries, archives, and educational institutions can circumvent access controls to make a determination of whether or not to acquire the work
- Impinging on fair use rights?
- First sale doctrine?

Case Study: DeCSS

- Content-Scrambling System (CSS) = encryption used for DVD
- DeCSS = program capable of decrypting CSS
  - Released in 1999 by Norwegian teenager Jon Johansen
- What are legitimate uses of DeCSS?
- Illegal to distribute under DMCA
- Creative ways of distribution:
  - T-shirts
  - Dramatic readings
  - Haiku poems
  - "Illegal prime number"

Case Study: Music Sharing

- In the beginning…
- Along came the Internet
- Rise and fall of Napster
- P2P out of the ashes
- The media companies strike back
- Sony rootkit: the latest saga

In the works...

- Broadcast flags
- "Trusted computing"

Why is this important?

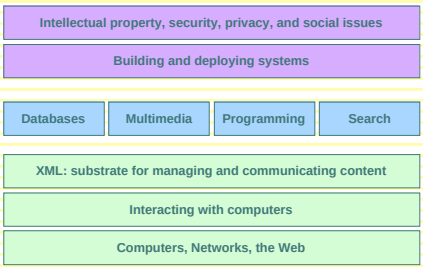
- ALA has traditionally been a defender of civil liberties
- Social responsibility of information professionals

Agenda

- Building and deploying systems
- The open source model
- Issues in intellectual property management

Questions?

The Grand Plan



Computer and Networks

- Computers and networks as devices for transferring, storing, and manipulating data
- Concepts of speed, time, and size
- Computer hardware
  - Managing time: storage hierarchy and caching
- Computer networks
  - Packet-switched networks: routers, routing tables, etc.

The Web

- HTML, HTTP, URLs
- How does a Web page get from the server to your Web browser?

Interacting with Computers

- User Interfaces: arguably the most important component of software!
- Interface design principles
  - Consistency, alignment, flow, etc.
- User's mental model of software

XML

- Substrate for storing, managing, and communicating content
- What XML is and what XML isn't
  - Cutting through the hype
- Future promise and potential of XML
  - Semantic Web
  - Interoperability

Relational Databases

- Tables and relations
  - Primary keys, foreign keys
  - Normalization
- Database operations
  - Join (select x, y)
  - Project (select…)
  - Restrict (where...)
- ACID Properties
- Web sites that are really databases
  - Deep vs. surface Web

Multimedia

- Tricking the human senses
  - Pixels to images
  - Images to movies
  - Sampling for audio
- Compressions, compression, compression
- Streaming delivery

Programming

- Like cooking!
- Built from individual instructions grouped into three types of control structures:
  - Sequential
  - Conditional
  - Iteration
- Functions "bundle together" common operations
- Javascript

Search and IR

- Why is IR hard?
- The interaction cycle
- Boolean queries and index structure
- Evaluating systems: precision vs. recall

System Life Cycle

- Building and deploying systems
  - System analysis
  - Software development models
  - Managing complex systems
- Open Source
- The “chilling effects” of DRM, DMCA, etc.

The Grand Plan