

INFM 603: Information Technology and Organizational Context

# **Session 13: Developing and Managing Technology**



Jimmy Lin  
The iSchool  
University of Maryland

Wednesday, April 30, 2013

# The Technology Lifecycle

- How do you know what to build?
- How do you actually build it?
- How do you keep it running?

It's about asking the right questions!

# First Things First

- What's already there?
- Do users actually want it?
- Is technology even the right answer?
- What's the use context?

Technology is for solving problems!

# How do you know what to build?

- Ask your users
- Watch them



So you're going to build technology...



# Major Decisions

- Build vs. buy
- In-house vs. out-source
- Open source vs. proprietary
- Best of breed vs. integrated solution





# Architecture Choices

- Desktop software
- Web app
  - Hosted in the cloud?
  - In-house datacenter?
- Mobile app

Advantages and disadvantages to each!

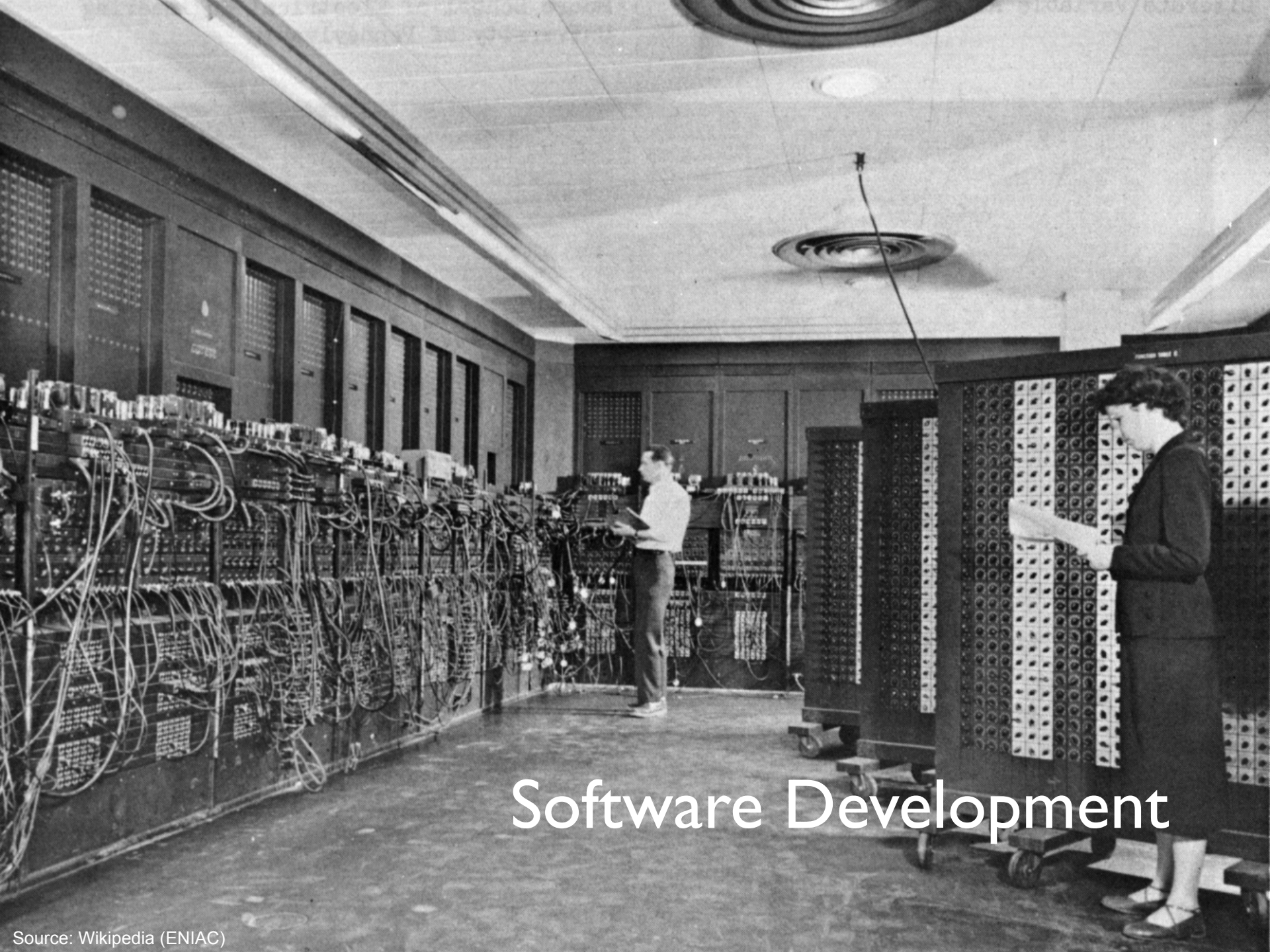


# Requirements

- Functionality
- Latency and throughput
- Capacity
- Reliability and resiliency
- Flexibility
- Development cost and time

There's no free meal!





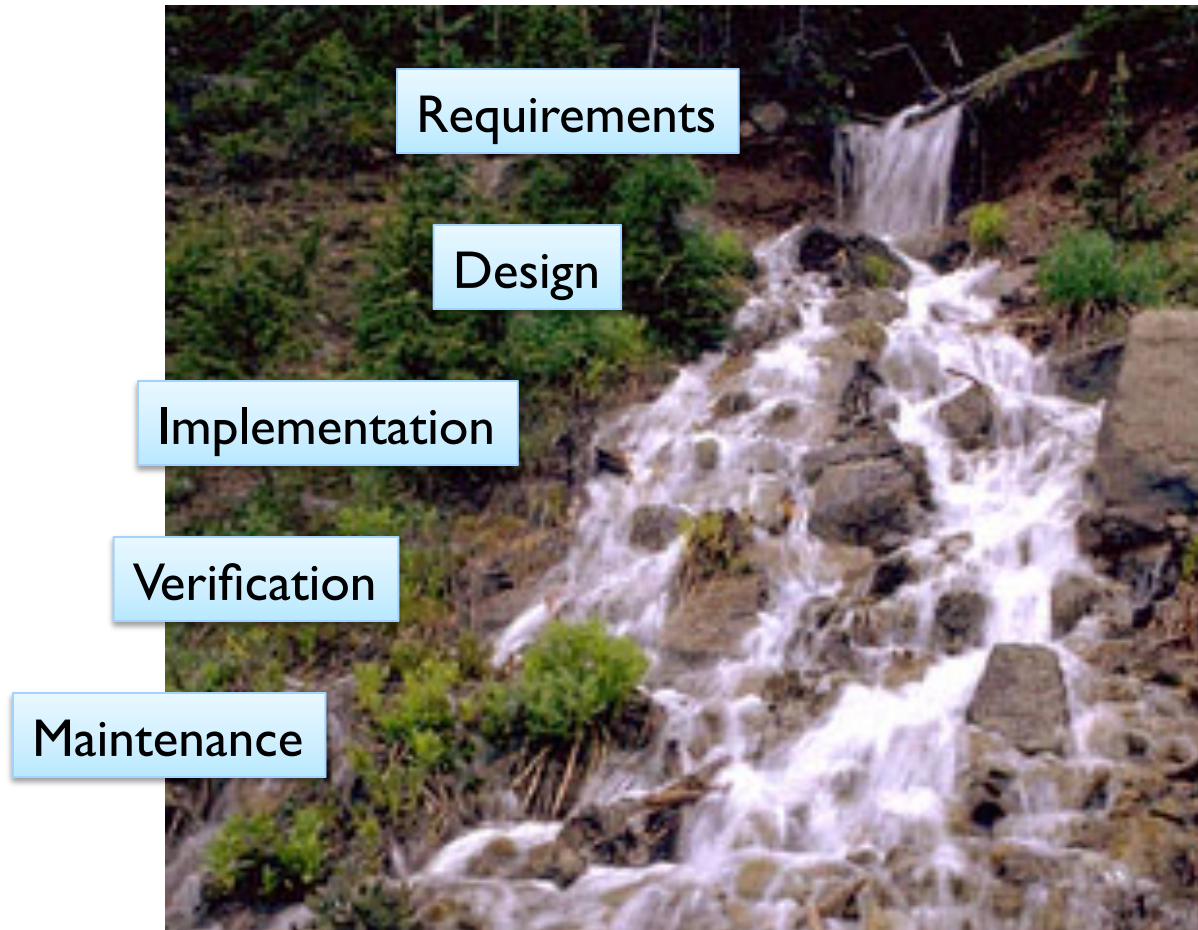
# Software Development

# The Waterfall Model

- Key idea: 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



# The Waterfall Model



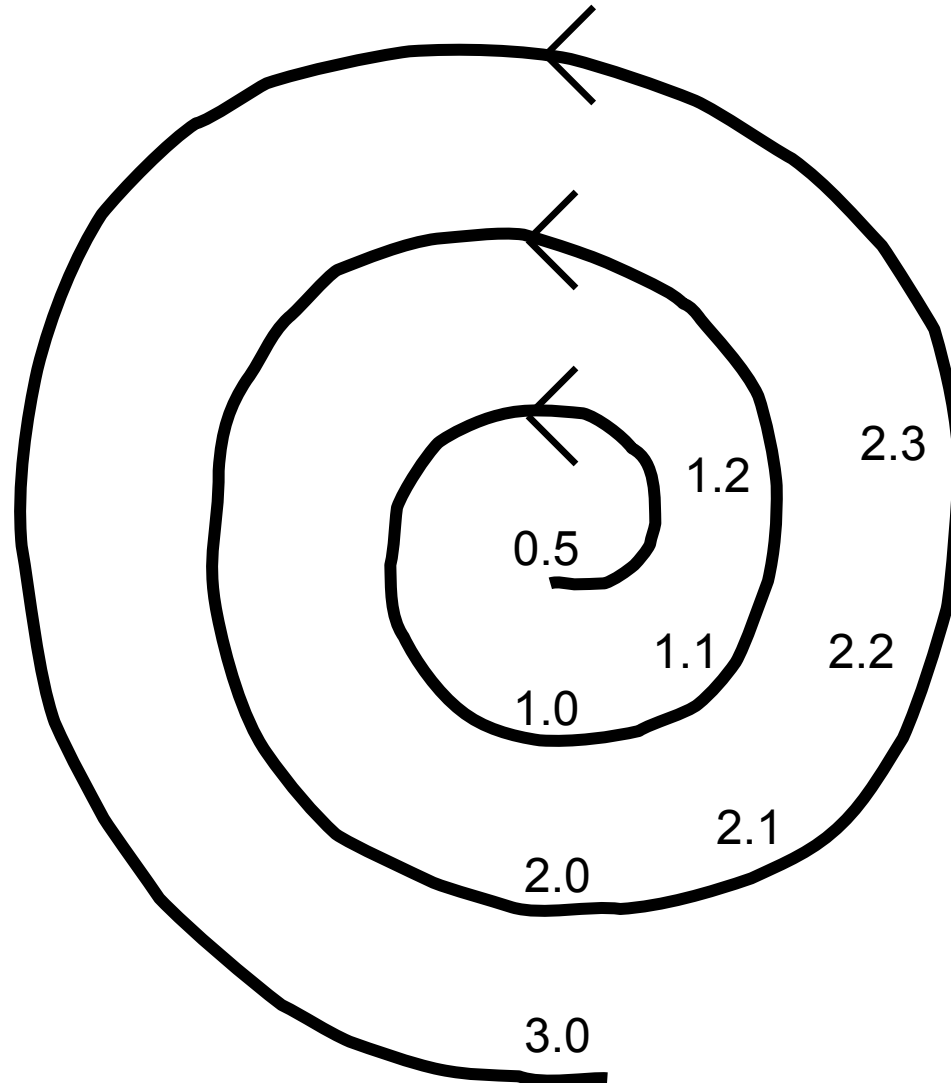


# The Spiral Model

- 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, ...)



# 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 wasted effort

# Prototyping

- What's the purpose of the prototype?
  - Meant to explore requirements, then be thrown away
  - An initial version of the software to be subsequently refined
- Both are fine, as long as the goal is clear
- Be aware, interfaces can be deceiving

# SCRUM

- An agile development methodology: the “fashion” today
- As with any methodology...
  - Don't blindly follow processes
  - Understand the rationale behind them
  - Adapt them to your context

# Key Concepts

## ○ Roles:

- Product owner: voice of the customer
- Development team: small team software engineers
- Scrum master: primary role as facilitator

## ○ User stories: short *non-technical* description of desired user functionality

- “As a user, I want to be able to search for customers by their first and last names”
- “As a site administrator, I should be able to subscribe multiple people to the mailing list at once”

# Basic SCRUM Cycle

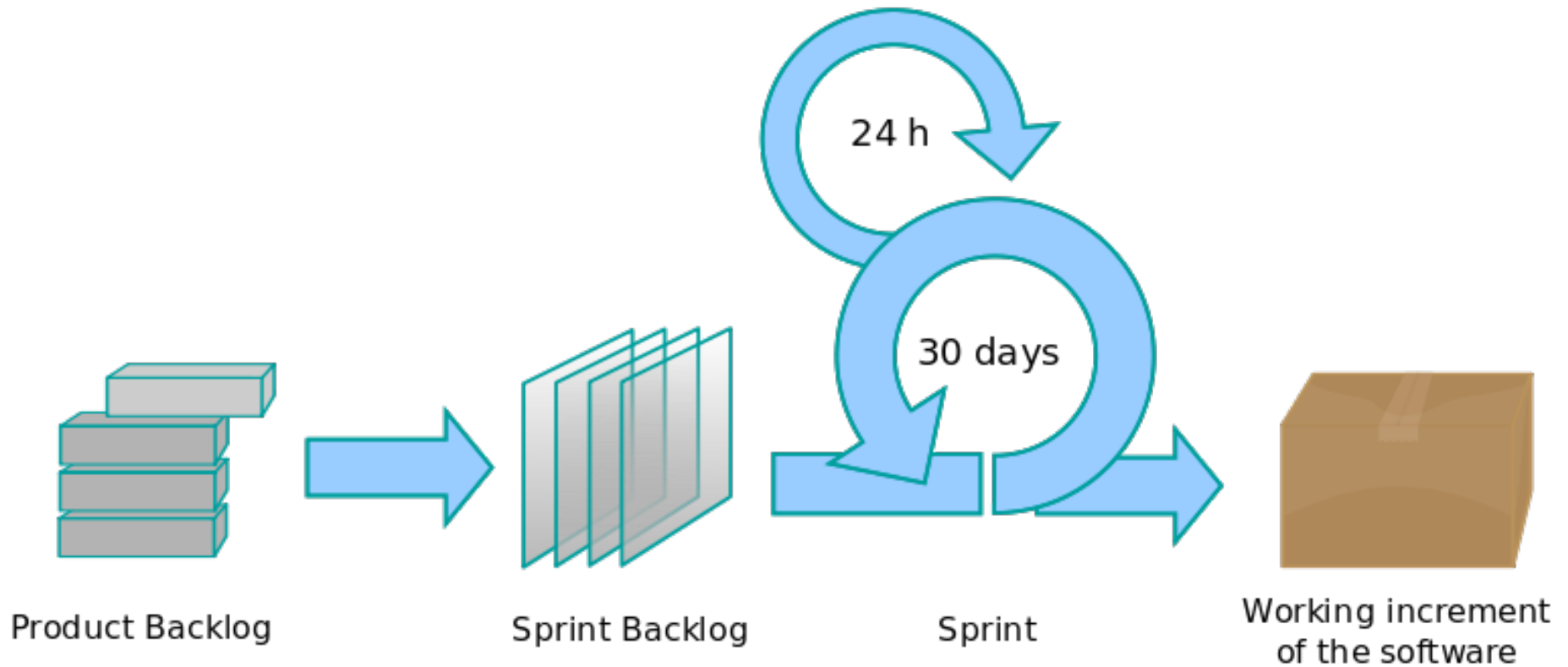
- The sprint:
  - Basic unit of development
  - Fixed duration (typically one month)
  - End target is a working system (*not* a prototype)
- Sprint planning meeting
  - Discussion between product owner and development team on what can be accomplished in the sprint
  - Sprint goals are owned by the development team

# Standup Meetings

- Short, periodic status meetings (often daily)
- Three questions:
  - What have you been working on (since the last standup)?
  - What are you planning to work on next?
  - Any blockers?



# SCRUM



# Advantages of SCRUM

- Fundamentally iterative, recognizes that requirements change
- Development team in charge of the sprint backlog
  - Favors self-organization rather than top-down control
  - Reprioritize in response to changing requirements and progress
- Time-limited sprints ensure periodic delivery of new product increments
  - Allows opportunities to receive user feedback, change directions, etc.
- Buzzword = *velocity*

# Disadvantages

- Can be chaotic
- Dependent on a good SCRUM master to reconcile priorities
- Requires dedication of team members
- Slicing by “user stories” isn’t always feasible



Keeping Technology Running



# Management Issues

- Operating costs
  - Staff time
  - Physical resources (space, cooling, power) or the cloud
  - Periodic maintenance
  - Equipment replacement
- Retrospective conversion
  - Moving from “legacy systems”
  - Even converting electronic information is expensive!
- Incremental improvements
  - Upgrade path?

# Management Issues

- Usage information
  - Usage logs, audit trails, etc.
  - Collection, storage, as well as analysis
- Training
  - Staff
  - Users
- Privacy, security, access control
- Backup and disaster recovery
  - Periodicity, storage location
  - The cloud doesn't necessarily solve all the issues

**Remember Murphy's Law!**

# TCO

- “Total cost of ownership”
- Hardware and software aren’t the only costs!
- Other (hidden) costs: all the issues discussed above



# 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?

# Examples of Open Source Software

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

# Open Source: Pros

- Peer-reviewed code
- Dynamic community
- Iterative releases, rapid bug fixes
- Released by engineers, not marketing people
- 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

# 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

# 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 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

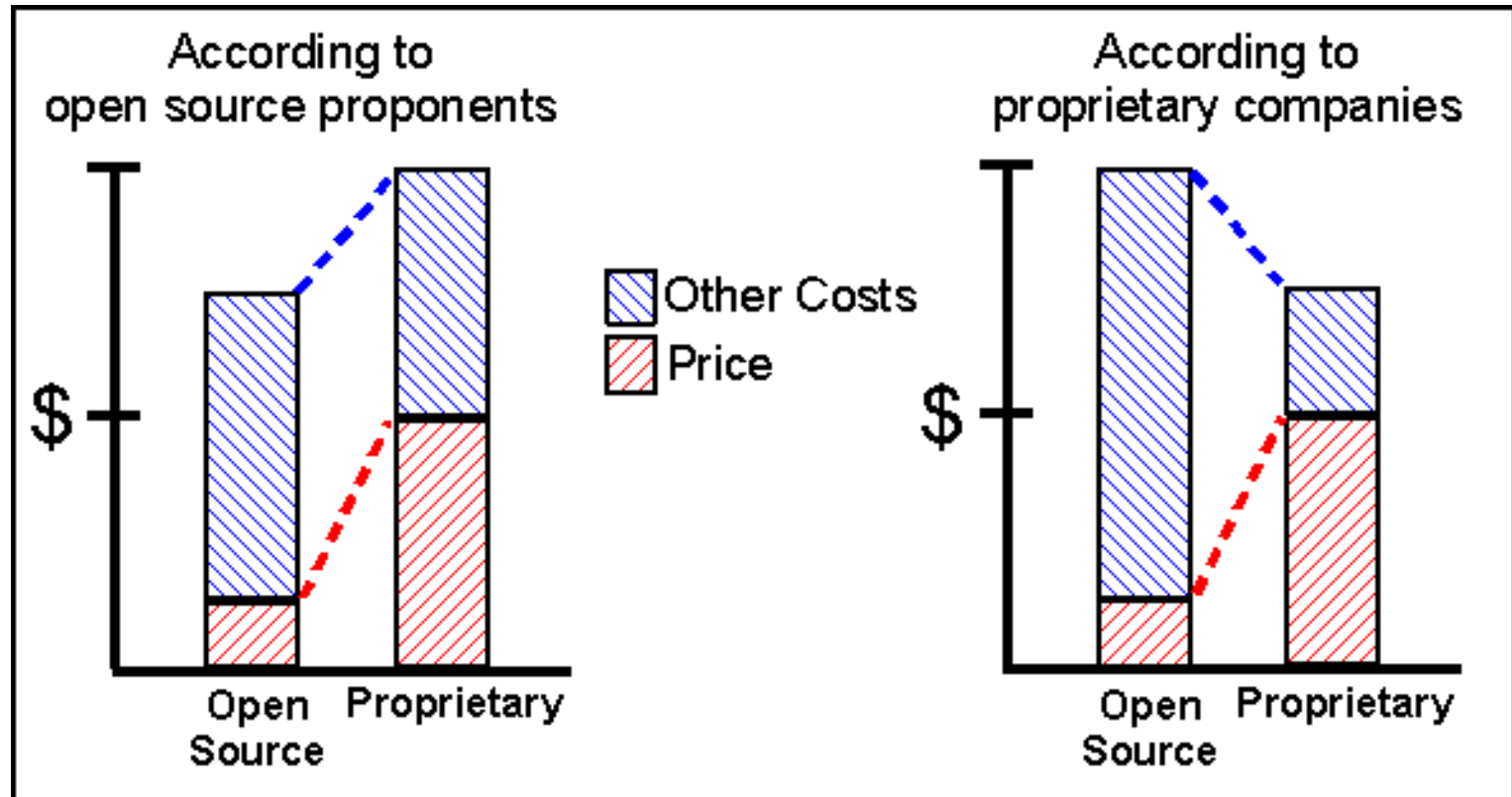
If you're in the hardware business, giving away software doesn't hurt you and has its advantages. What are they?

- Accessorizing

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



# The TCO Debate



# Is open source right for you?

- Depends

# Questions?

