LBSC 690: Information Technology
Lecture 12
Software system development and deployment

William Webber
CIS, University of Maryland

Spring semester, 2012

Spectacular software project failures

The Advanced Automation System project was launched in the early 1980s and was originally expected to cost $2.5 billion and be completed by 1996. But by 1994, estimated project costs had soared to $7.6 billion and the project was seven years behind schedule. The FAA terminated some parts of the AAS program and restructured others, but $1.5 billion of spending ended up being completely wasted.

[1] http://www.downsizinggovernment.org/transportation/airports-atc

Spectacular software project failures

Last October [...] the giant British food retailer J Sainsbury PLC had to write off its US $526 million investment in an automated supply-chain management system. It seems that merchandise was stuck in the company's depots and warehouses and was not getting through to many of its stores. Sainsbury was forced to hire about 3000 additional clerks to stock its shelves manually.

[1] IEEE Spectrum, "Why Software Fails", Sept. 2005

Spectacular software project failures

Senior managers at RMIT University botched virtually every aspect of the implementation of a $47 million software system that collapsed last year, an Auditor-General's report has found. The system will have to be scrapped. [...]
The malfunctioning system corrupted financial records and led to delays in issuing student cards and billing of international students.
Auditor-General Wayne Cameron also said that international student enrolments at RMIT dropped by between 6 and 18 per cent last year as a result of the debacle.

Software project failure rates

- 51% of companies implementing an ERP solution felt it was unsuccessful (Robbins-Gioia, 2001)
- 40% of ERP implementations failed to achieve business case within 1 year of going live (Conference Board Survey, 2001)
- 61% of projects failed amongst survey respondents (KPMG Canada, 1997)

[1] http://www.it-cortex.com/Stat_Failure_Rate.htm

Software project failure rates

Survey by Standish Group, 2009.

- 32% of projects delivered on time, on budget, with required features
- 44% late, or over budget, or short on spec
- 24% failed and cancelled, or delivered but not used

[1] http://www.cbronline.com/news/software_project_failures_hit

Think carefully before starting

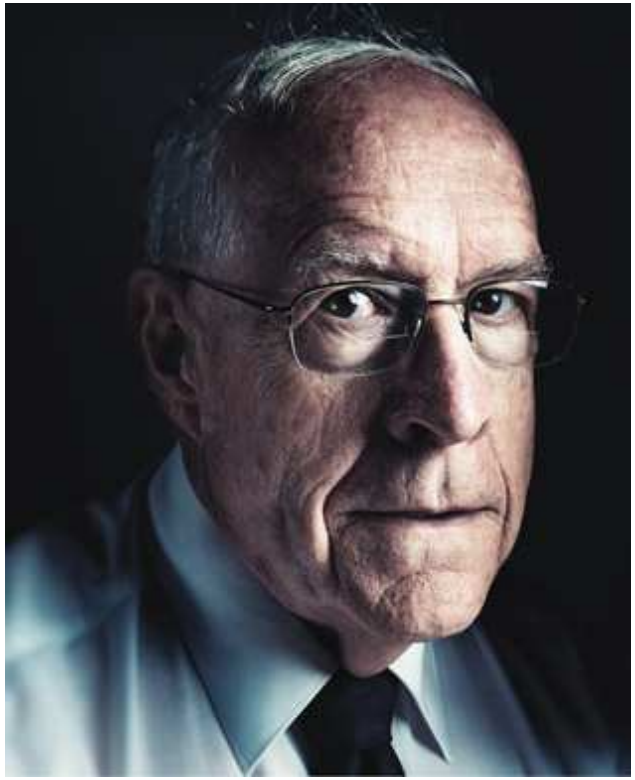
- Software development a risky business
- Do you really need to build this product?
- Is there an existing alternative?

Building software vs. building a bridge

$\neq$

- Newness, unfamiliarity of technologies used
- Projects have high degree of uniqueness (software can always be copied)
- Customer doesn't really know what they want, or how the system will in practice be used

Fred Brooks, “Mythical Man Month” (1975)



- Programmers are optimists, and always underestimate time required
- A constrained system is often a better system (second system effect)
- Programming tasks are imperfectly fungible:

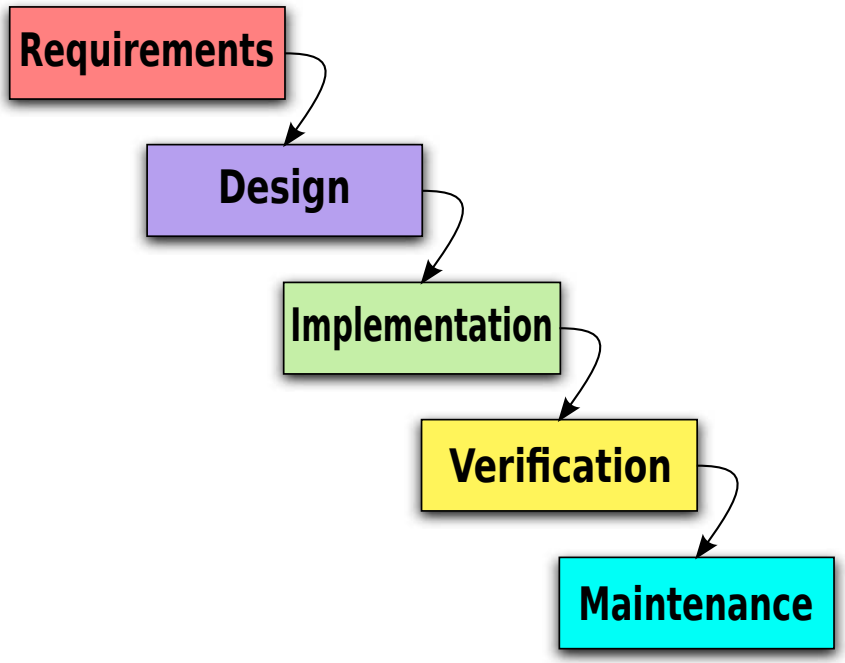
  The bearing of a child takes nine months, no matter how many women are assigned.

  Adding manpower to a late software project makes it later.

The “Ad-Hoc” (hobbyist) model

- The Heroic Age of Software Development
- Just dive in to coding . . . and struggle your way out again
- Plays to image of the “heroic programmer”, who wrests the project to completion by the raw force of his brute masculinity . . . or goes down bravely with the project if it fails

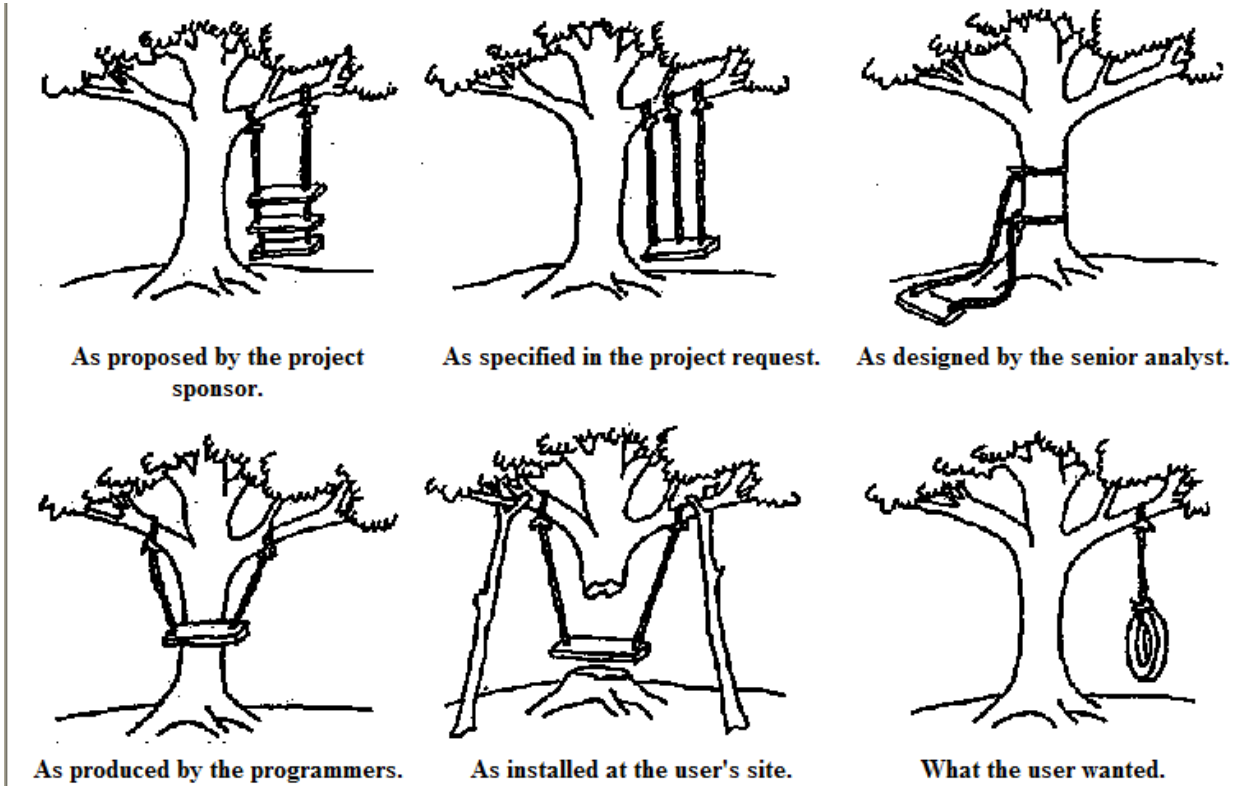
The “waterfall”



- Follow strict order of software development
- When moved to step $n + 1$, cannot go back to step n
- . . . except possibly for iterating full process
- Term usually used perjoratively
- . . . but model very widely employed

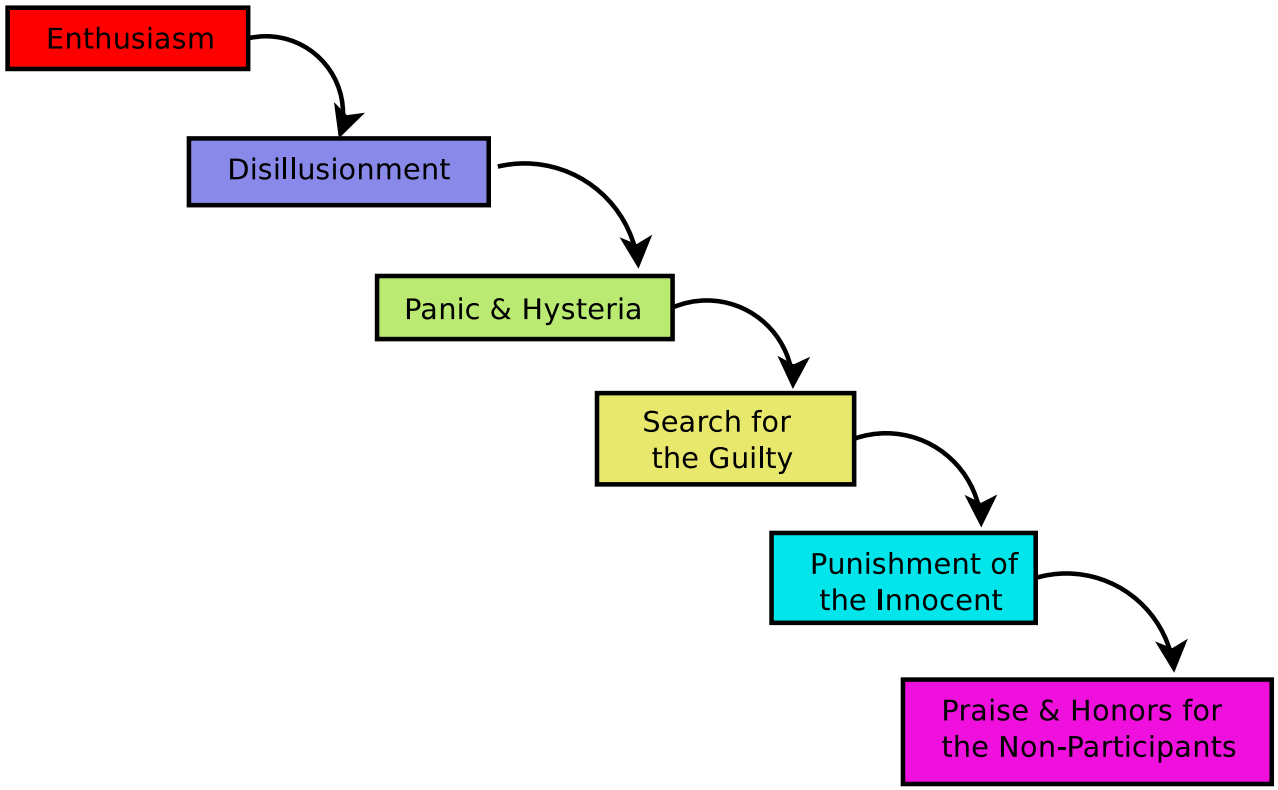
[1] Wikipedia, “The Waterfall Model”

What the customer wanted...



- User may not know, be able to specify what they want in requirements
- No mechanism for course-correction as project proceeds

The true waterfall



- The waterfall model doesn't have a great reputation with programmers
- Why?

[1] Attributed by Steve McConnell to Chris Kemerer

Agile development

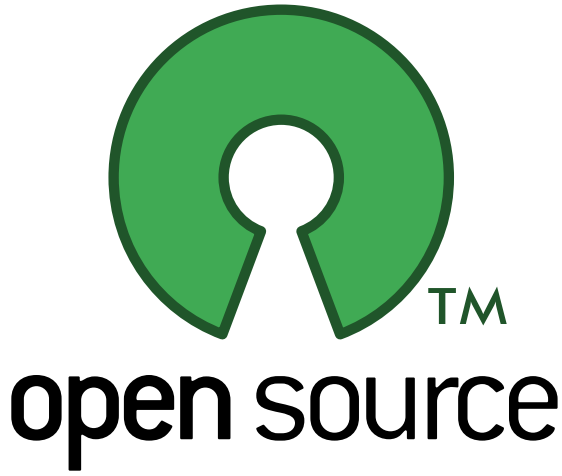
- Iterative and incremental software development
- Each iteration moves through full (waterfall) cycle
- Continuously develop working software for demonstration to client
- A family (or philosophy) of methods, with many formal implementations

[1] Wikipedia, “Agile software development”

Agile pros and cons

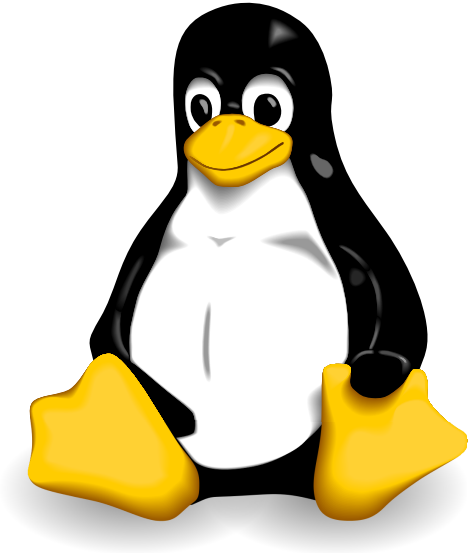
- Experience suggests Agile approach works well for
  - Low-critical systems (tolerant to failure)
  - Talented developers
  - Small teams
  - Frequently changing requirements
- But can also be euphemism for ad-hoc method
- Biggest problem (it seems to me): what goes in the contract?

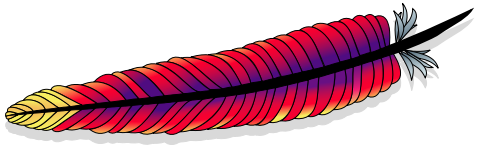
Free software: revenge of the hobbyist



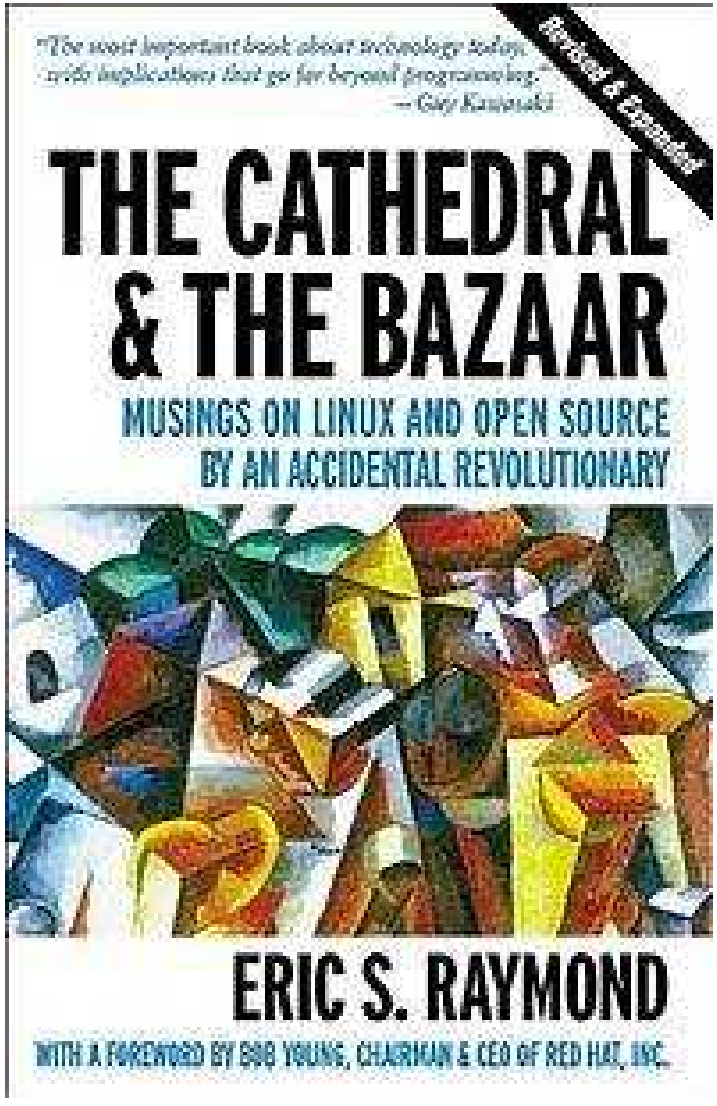
- Hobbyist (amateur?) mindset: work on problems that interest you (rather than from financial motivations)
- Open source: release source code of your projects, let others freely download, correct, modify, and extend it
  - Grew out of earlier, more dogmatic free software movement (software *should* be free)
- Open-source model now employed by large companies
- Large inspiration behind wiki approach, too

Open-source projects





The Cathedral and the Bazaar



Open source as a different model for software development

- Fruitful chaos of (trading in a) marketplace, rather than hierarchical control of (building a) cathedral
- Allow anyone to contribute to a project

Convinced Netscape to open-source Mozilla (now Firefox)

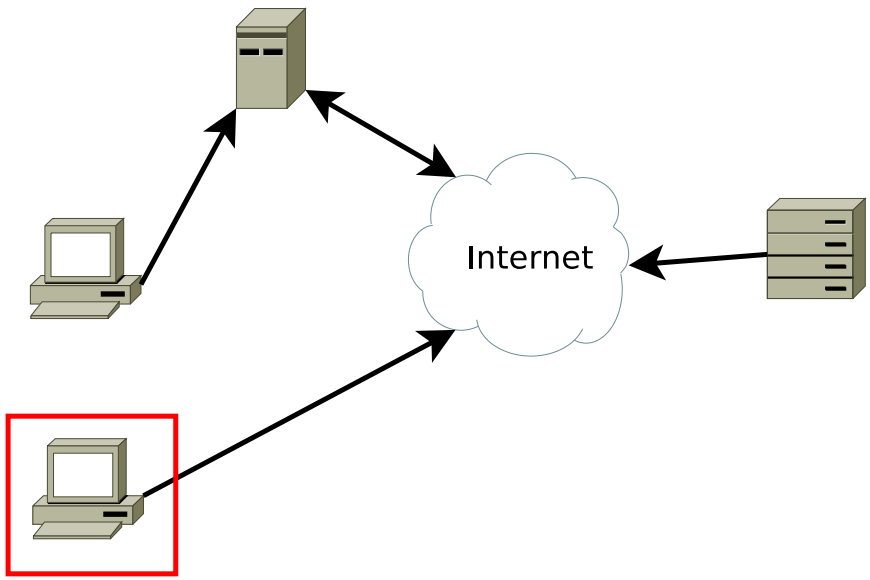
Considering open-source

- In practice, seems (well-targeted) bug reports from the open-source crowd more common than substantive code
- . . . though approach allows self-selection of dedicated project contributors
- What is the business model? (though little money in selling generic software these days)
- What if no-one is interested in the area?
- What does it mean for software-as-a-service?

Application deployment options

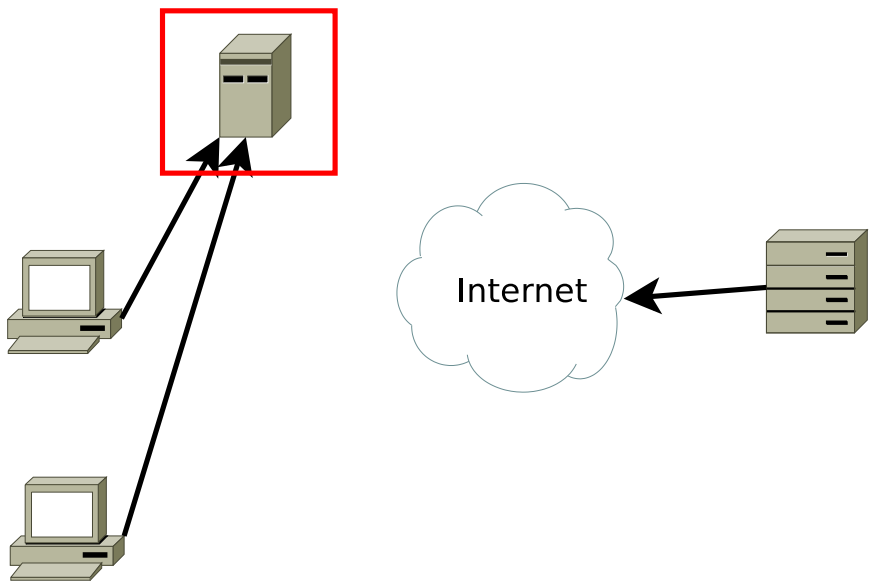
Where to run the software you have commissioned (purchased, downloaded)?

- On client



Application deployment options

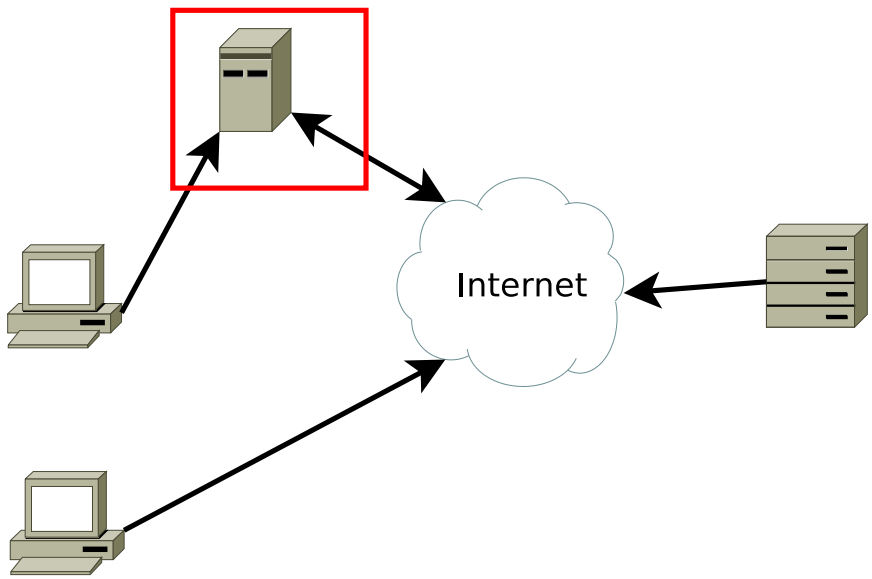
Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet

Application deployment options

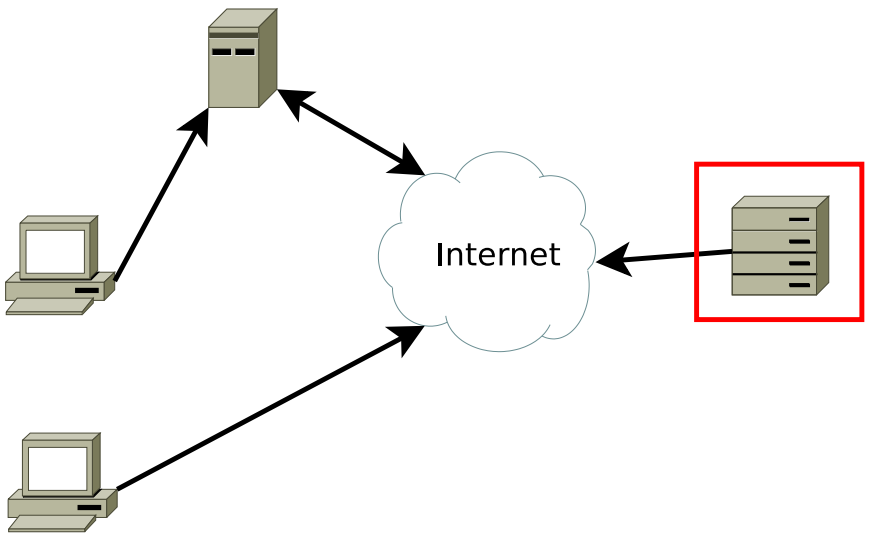
Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet
- On local server with internet connection

Application deployment options

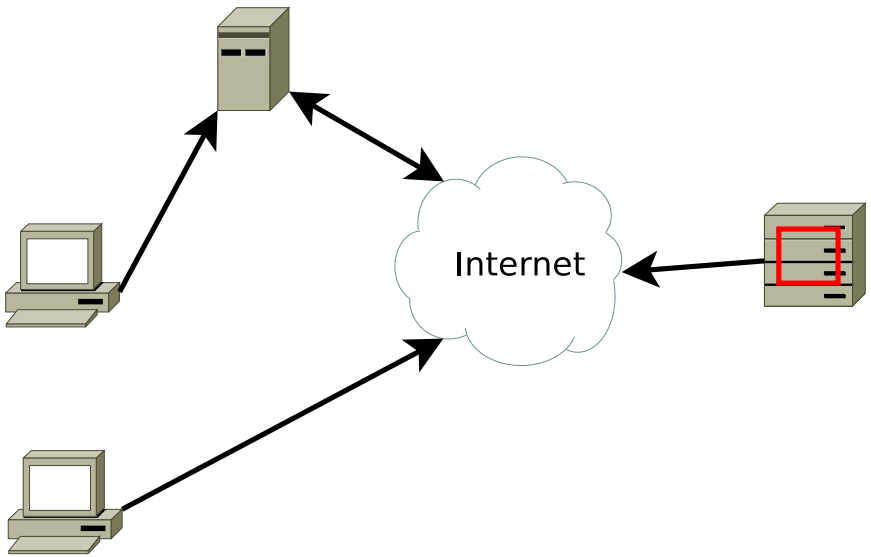
Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet
- On local server with internet connection
- On dedicated hosting machine in data center

Application deployment options

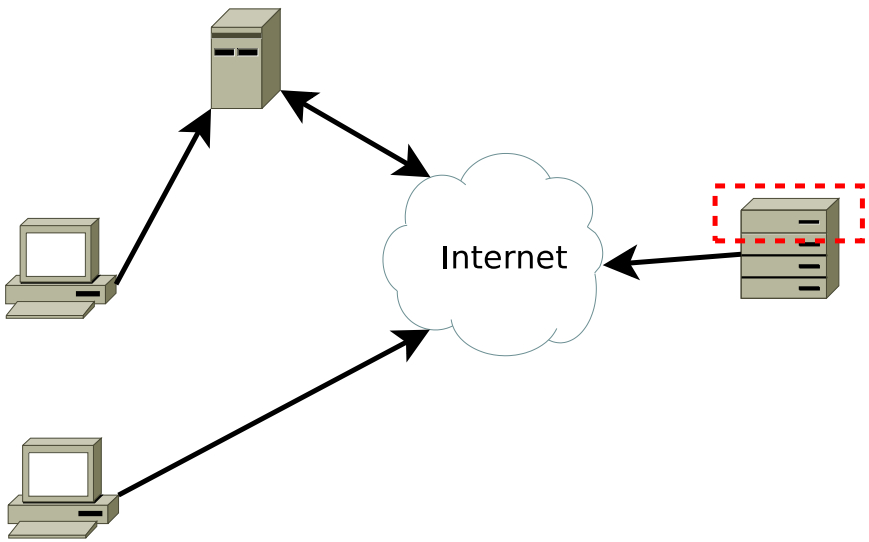
Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet
- On local server with internet connection
- On dedicated hosting machine in data center
- On shared host in data center

Application deployment options

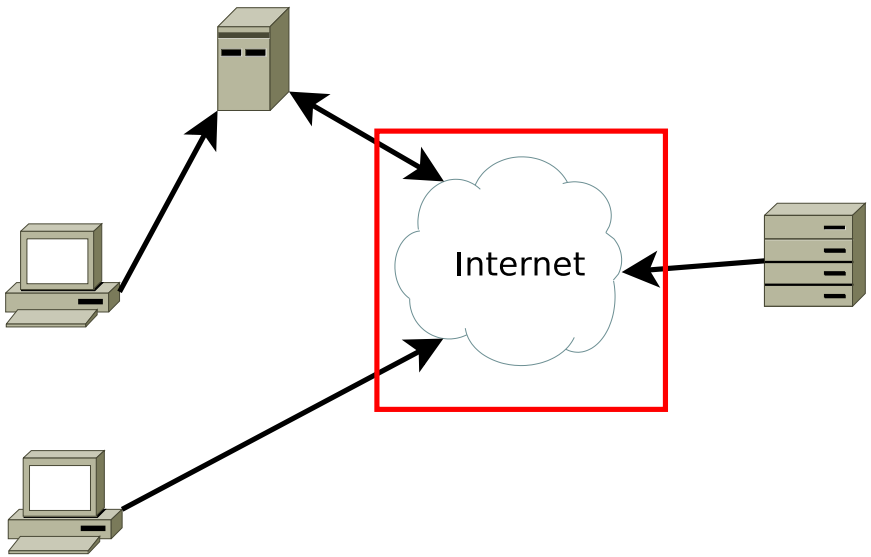
Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet
- On local server with internet connection
- On dedicated hosting machine in data center
- On shared host in data center
- On virtual private server

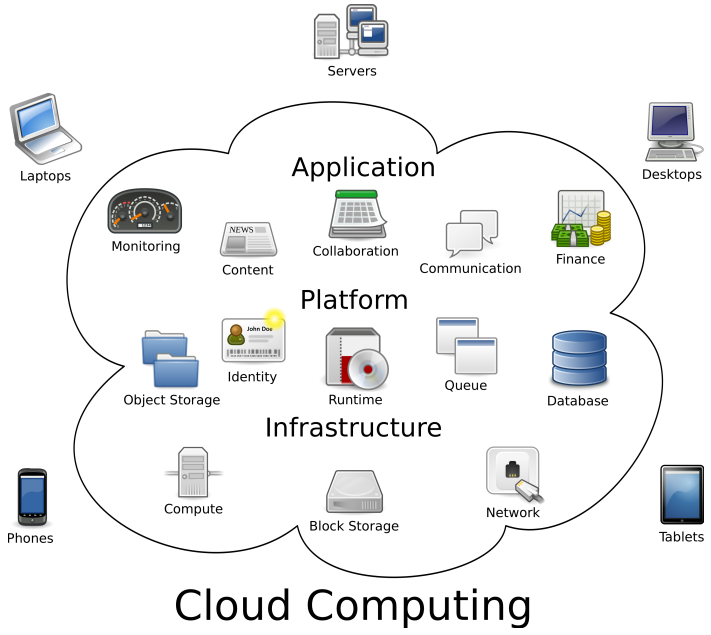
Application deployment options

Where to run the software you have commissioned (purchased, downloaded)?



- On client
- On local intranet
- On local server with internet connection
- On dedicated hosting machine in data center
- On shared host in data center
- On virtual private server
- On “the cloud”

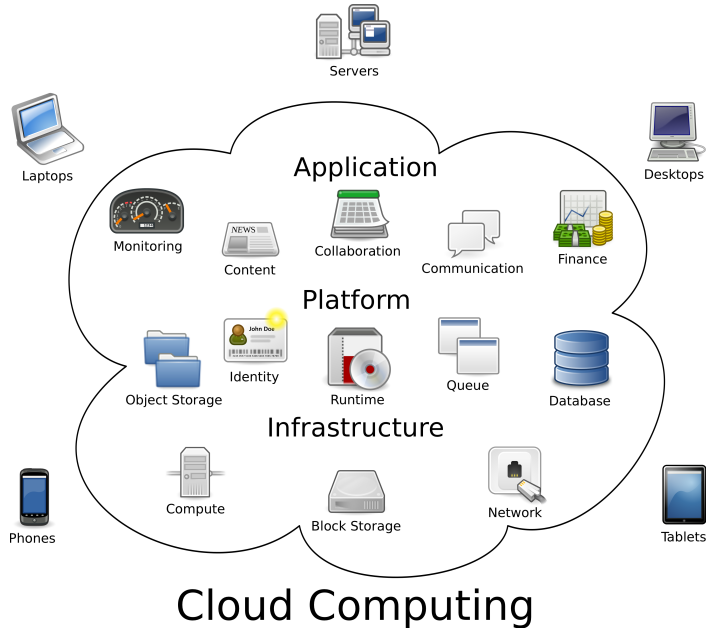
Cloud computing



What constitutes cloud computing?

- Buy computing as a utility, not as infrastructure
- Pool of shared resources, allocated on demand
- Provided at some “anonymous” remote data center or centers

Three levels of cloud computing

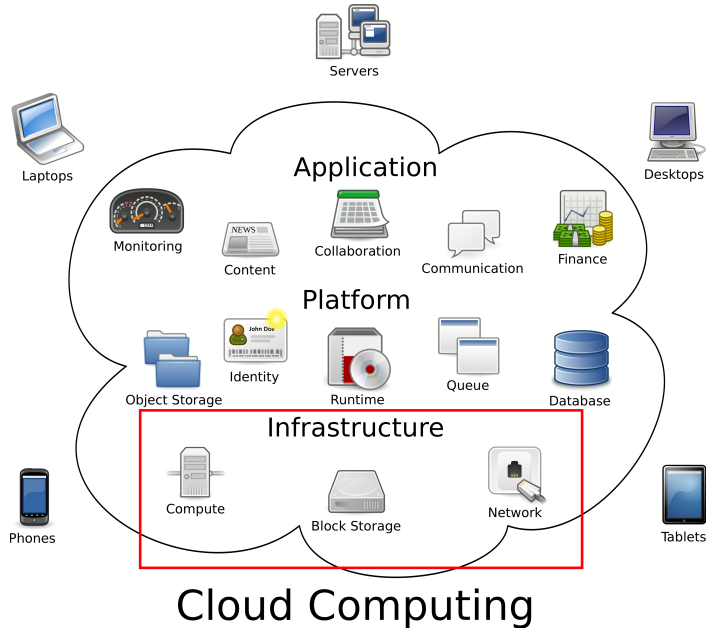


Three varieties of cloud computing:

- IAAS Infrastructure-as-a-service
- PAAS Platform-as-a-service
- SAAS Software-as-a-service

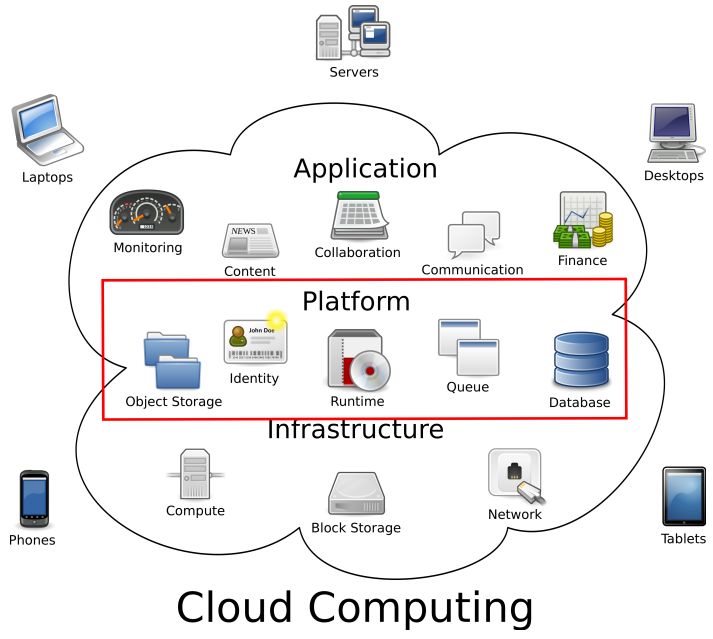
[1] Wikipedia, “Cloud computing”

Infrastructure as a service



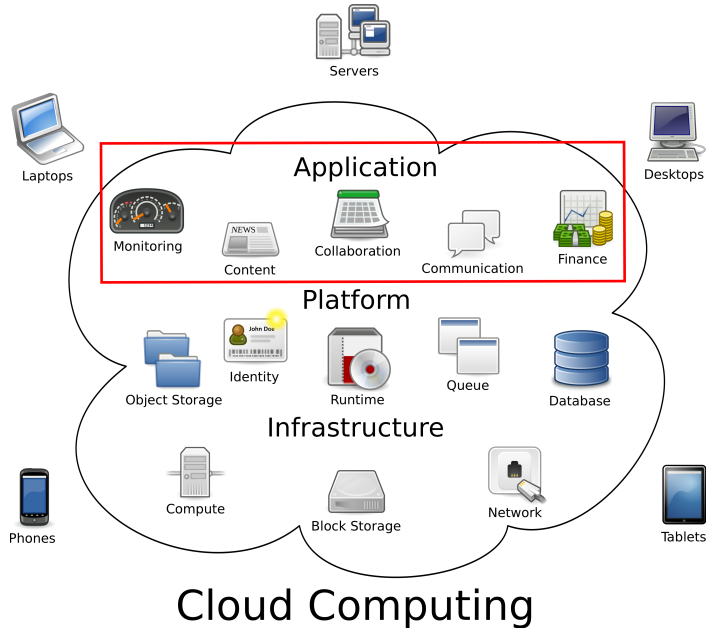
- Provide raw (virtual) machines; user installs OS, software
- User charged on usage basis for:
  - CPU minutes
  - Disk space used
  - Amount of I/O
  - Network utilization
- Alternative to dedicated hosting
- For example, Amazon EC2, `http://aws.amazon.com/ec2/pricing/`

Platform as a service



- Provide, charge for platform on (scalable) cloud infrastructure:
  - Web server
  - Web programming runtime
  - Database
- Alternative to shared hosting

Software as a service



- Provide software services (through browser) running remote from client (often on cloud substratum)
- Charge by usage or by subscription
- Can be used to replace client applications
  - E.g. gmail
- Can be used to replace locally installed, maintained enterprise software
  - E.g. salesfore.com customer relationship management (CRM) software

Why cloud computing?

- Pay as (and what) you use rather than infrastructure payment
- Lower costs through commoditization of resources, services
- Supports variable workloads
- Outsources maintenance tasks

Why not cloud computing?

- Premium pricing
  - Amazon EC2 storage charges 10c / GB / month, plus 10c / million read/writes
  - Purchase redundant disks for 25c / GB, transfers free
- If usage level stable, purchasing infrastructure may be cheaper
- Concerns about loss of control of corporate, esp. government, data

Summary

- Software development is risky
- Ad-hoc, waterfall, and agile development models
- Open source software as an alternative to proprietary software
- Multiple options for application deployment
- Cloud computing: computing utility, not infrastructure