

CS 501: Software Engineering

Fall 2000



Lecture 26

Risk in Software Engineering

Administration: Laptop Return Times



Date: Wed Dec 6th

Time: 9:30 - 11:00 AM

Room: 5130 Upson

Date: Mon. Dec 11th

Time: 1:30 - 3:00 PM

Room: 5130 Upson

Date: Wed. Dec 13th

Time: 9:30 - 11:00 AM

Room: 5130 Upson

Administration



- Examination

1.5 hours

5 questions

based on lectures

Objective is to reward those people who were regular in attending class.

Failures and Risks



Software development projects can fail in many ways:

Bad software engineering

- Late, over budget
- Lack of function, full of bugs, bad performance

Changing circumstances

- Changing markets
- Better alternatives
- Changes of management

The biggest single source of problems is poor understanding of requirements

Managing Risk



Manage projects to avoid risk:

- Open and visible software process
 - => Avoid surprises
- Continual review of requirements
- Willingness to modify or cancel projects

Canceling a Project



Example: Andrew Window Manager (wm)

- Technically superior to X (MIT's Athena project) in 1986
but ... Digital Equipment Corporation turning X into a product with massive support
nobody ready to support wm
- Therefore wm cancelled in 1986, Andrew user interface and applications ported to X

Failure to Cancel a Project



Example: University F developed a novel programming language.

- From 1985 to 1989, this was a promising language for simple programming of window-based applications
- By 1990, clearly not gaining acceptance beyond University F
- But development continued for many more years (about \$500K)

Not cancelled because ...

Too Big to Cancel!



Example: University A has antiquated administrative systems. Senior management decides to replace them all with commercial packages from X. The timetable and budget are hopelessly optimistic.

- Staff get dispirited.
- The Chief Information Officer finds another job.
- A new Chief Information Officer is appointed.

What should she do?

We are doing it the Wrong Way!



Example: University B has a (big) joint project with Company Y to develop a new computer operating system.

After two years work, a junior software developer persuades the university leader that the technical approach is wrong.

- What should the university do?
- What should the company do?

How to Stop Gracefully



- It is harder to cancel a project than to start it.
- It is harder to withdraw a service than introduce it.

Considerations

- The proponents of the system must now reverse their public stance.
 - => Management of expectations
- Users of the service need a migration strategy.
- Technical staff must have a graceful path forward.

Time to Complete a Software Project



Large software projects typically take at least two years from start to finish

- Formative phase -- changes of plan are easy to accommodate
- Implementation phase -- fundamental changes are almost impossible

Yet many things can change in two years.

A Sense of Urgency



Example: A not-for-profit corporation is developing a system for a government organization.

- By 1996 all research had been completed and the system demonstrated successfully with real users.
- In 2000, the system is still not in full production

Reasons:

- => Incremental improvements to the software
- => Repeated requests for more functionality
- => Reluctance to reorganize clerical staff

Nobody had a sense of urgency

Overtaken by Events



Example: University C has a project to develop a digital library system, with funds from Company Z , private foundations and the government.

- By 1993 an extensive system is running at the university and Z is marketing the technology to its customers.
- By 1994 it is clear that web browsers and web formats (though technically weak) are becoming widely adopted.

=> What should the university do?

=> What should the company do?

Changing Requirements and Design



Example: The CNRI Handle System -- a high performance, distributed system to map names to resources (1994-99).

- In 1994 only web browser was Mosaic
- In 1994 Java did not exist
- In 1994 mirroring and caching utilities were not available
- In 1994 commercial interest was limited

Design decisions made in 1994 had to be changed. Software was rewritten and greatly improved in 1998/9.

If a job's worth doing, it's worth doing twice!

Changes of Leadership



Many projects are wasted because of management changes

Example: In 1988, Company W gave University D \$1,000,000 to port a new operating system to its personal computers. The work was well done, on time.

- Company W changed its president and senior technical staff during the project. The work was wasted.
- A decade later and several presidents later, Company W is releasing a modern version of the same operating system.

A graduate student from University D is now Senior Vice President of Company W!

Client Oversight



When work is out-sourced, the client must be vigilant.

Example: Company G was the world's leader in software for optimization (e.g., linear and integer programming). G had implemented several packages for various manufacturers.

- An operating system Company H contracted with G to develop an optimization package for its new operating system.
- The package was late, performed badly and disliked by customers.

What went wrong? What can we learn?

Too Difficult!



Example: A development team at University E was given government funds to build a high-performance gateway from protocol x to protocol y .

- A promising young developer was hired and assigned to this task
- The project was too difficult for him, but he hid his problems for many months.
- The project produced nothing of value.

What can we learn from this experience?

Engineering and Marketing



Corporate engineering & marketing divisions at cross purposes:

Examples:

- Xerox's Palo Alto Research Center pioneered window managers, Ethernet, graphical user interfaces, font managers, etc,
=> Apple, Adobe, Digital, etc. brought them to the market
- IBM would not bring its first Unix workstation to the market until the software had been largely rewritten
=> Sun's early workstations were unreliable but sold well

Senior Management Dynamics



- Directors and shareholders appoint the President
=> The President does not want to admit failures
- The President appoints the Chief Information Officer
=> The CIO does not want to admit failures
- The CIO appoints the computing managers
=> The computing managers do not want to admit failures
- The computing managers appoint the developers
=> The developers do not want to admit failure

Everybody pretends that things are going well

Senior Management Dynamics



At last the troubles can not be hidden ...

- Directors and shareholders try to blame the President
- The President tries to blame the Chief Information Officer
- The CIO tries to blame the computing managers
(and grumbles about the President)
- The computing managers try to blame the developers
(and grumble about the CIO)
- The developers grumble about their managers

What can we do better?

Sobering Thoughts



- Major computing projects are very complex. Inevitably there are delays and failures.
- Few organizations know how to manage risk & uncertainty.
- The best CIO's
 - => Manage to minimize risk
 - => Have the confidence of their staff who keep them truthfully informed
 - => Have the self-confidence to keep their seniors truthfully informed