Cornell University
Computing and Information Science

CS 5150 Software Engineering
25. Professionalism

William Y. Arms

Personal Thoughts about Software Engineering

The software industry needs to take more responsibility for its products.

- Products are sold or licensed with no guarantees about whether they work as advertised.
- Time-to-market dominates much of software development.
- We know how to develop much more reliable and secure software, but reliability and security often have low priority in business plans.

Compare: The car industry (safety and reliability)

Personal Thoughts about Software Engineering

Too much software development is wasted.

- Projects are often begun with poorly thought out scope.
- Senior management often pays little attention to a strategic area of their organization.
- Technical teams are often poorly led.
- We do not have a good methodology for education in software development.

Example: Thick sandwich

Why Software Projects Fail

There is an article on this topic in IEEE Spectrum:

http://spectrum.ieee.org/computing/software/why-software-fails

“We waste billions of dollars each year on entirely preventable mistakes.”

Failures and Risks

Software development projects can fail in many ways:

The software engineering triangle

- 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 between the client and the development team.

Failures and Risks: Some Examples

Here are some examples of software projects that have encountered problems. In some of them the problems were successfully overcome.

Most of the examples are based on my own experience. Some are recent; others happened many years ago.

I have hidden the names of the organizations and the dates, but these are all true examples. Because my experience has been mainly with universities, not-for-profits, and government agencies, most of the examples are from those areas.

Example: Too Difficult

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?

Example: Failure to Cancel a Project

University F developed a novel programming language for educational software.

- For several years, this language showed considerable promise. It was used successfully for a number of educational projects.
- After a few years, it was clear that the language was not gaining acceptance beyond University F and that it was being left behind by alternatives.
- But development continued for many more years (about $500K).

Not cancelled because ...

Example: Too Big to Cancel

Organization A had antiquated administrative systems. The senior management decided to replace them all with commercial packages from X. The timetable and budget were hopelessly optimistic.

- Staff became dispirited. Many of the best people left.
- The Chief Information Officer found another job.
- A new Chief Information Officer was appointed.

What should she do?

Example: Doing it the Wrong Way

University B had a (big) joint project with Company Y to develop new system software.

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

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

Doing it the Wrong Way (Second Example)!

Company X had a (very big) project to develop a new computer operating system.

After several years work by thousands of people, the head of the company reviewed the project and decided that the technical approach was wrong.

- What should the company do?

Example: A Sense of Urgency

A not-for-profit corporation developed a system for a government organization.

- After three years all the major components had been completed and the system demonstrated successfully with real users.
- Nine years later the system was 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

Example: Canceling a Project

A group at University P developed innovative user interface software.

- It was technically superior to any commercial alternatives.
- A group at University Q was doing work in the same area.
- The leader of the work at University Q considered the work at University P to be more advanced.

but...

- University Q had an industrial partner who planned to turn their work into a product with massive support, while nobody was committed to supporting the work of University P.

What should University P do?

Example: Overtaken by Events

University C had a project to develop a large application suite for libraries, with funds from Company Z , private foundations, and the government.

- After several years development, an extensive system was running at the university and Z was marketing the technology to its customers.
- But technical management at both C and Z became convinced that better protocols and formats than those chosen were now available and the system was not going to succeed.

What should the company do?

What should the university do?

Example: Changes of Leadership

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.
- Many years later and several presidents later, Company W has built its future around a modern version of the same operating system.

A graduate student from University D became Senior Vice President of Company W.

Example: Accept the Obvious

Six organizations were funded by the National Science Foundation, for one year, to develop demonstration projects.

The National Science Foundation hoped that the six organizations would then submit a multi-million dollar, five year proposal to develop the production system together.

- But ... there were differences (technical and personal) between the organizations.
- Three weeks before the proposal was due, the principal investigator at University M decided that the plan was doomed to failure.

What should he do?

Learning from the Examples

What can we learn from these examples?

In every case, the difficulties concern the **relationship** between senior management, technical management, development teams, and individual developers.

- When this was a good relationship, tough decisions were made and the problems were resolved or at least minimized.
- When this was a bad relationship, nobody faced up to the problems and the waste of resources continued.

Senior Management Dynamics

Only too often...

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 mangers 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 and uncertainty.

The best Chief Information Officers:

- 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.

Managing Risk

Manage projects to avoid risk

- Open and visible software process, avoiding surprises
- Continual review of requirements
- Willingness to modify or cancel projects
- Try for short phases each with deliverables

Managing Risk: Visibility

Example:

An ambitious leader forms a consortium, whose success depends on a large innovative computer system. The system is continually failing, runs very slowly, loses important data, etc.

- The board hire a consultant to advise them whether to abandon the entire consortium.
- The consultant talks to the technical staff.
- The staff have a technical plan to solve the problems, but need time.
- The ambitious leader has continually promised unrealistic delivery dates.
- The consultant asks the technical team for a cautious timetable and explains it to the board.
- The board approve the extended timetable, which turns out to be quite accurate.
- The consortium is successful.

What can we learn from this example?

Managing Risk: Changing Requirements and Design

Example

In 1994 a not-for-profit organization developed a high performance, distributed system to map names to resources.

Decisions made in 1994 had to be changed.:

- In 1994 the only Web browser was Mosaic
- In 1994 Java did not exists
- In 1994 mirroring and caching utilities were not available
- In 1994 commercial interest was limited

Software was rewritten and greatly improved in 1998/9. Today, it is widely used.

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

Managing Risk: Time to Complete a Software Project

Large software projects typically take at least two years from start to finish, often much more.

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

Yet many things can change in two years.

Incremental software methodologies, such as agile, aim to apply the advantages of small teams and short development cycles to large projects.

Managing Risk: How to Stop Gracefully

Canceling a project

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

If a project has to be cancelled, do it as early as possible.

Software Development as Engineering

Software development demands a high degree of professionalism.

Question: Is software development a branch of engineering?

Answer: It depends on how you define *engineering*.

Engineers, the State, and the ACM

Proposal in a state legislature to license professional engineers in Software Engineering.

What role should the ACM play?

What is Engineering?

A definition of engineering

The profession of:

... creating cost-effective solutions ...

... to practical problems ...

... by applying scientific knowledge ...

... and established practices ...

... building things ...

and taking responsibility for them!

With this definition, software development is clearly engineering.

What is Engineering?

A second definition of engineering

A professional who

… is licensed by a professional society …

… based on a set educational program with a standard body of knowledge and specified experience …

… who is the only person permitted to oversee certain tasks …

If this is your definition of engineering it is hard to see it applied to software development.

Professional Responsibility: Safety Critical Software

A software system fails and several lives are lost. An inquiry discovers that the test plan did not consider the case that caused the failure. Who is responsible?

(a) The testers for not noticing the missing cases?

(b) The test planners for not writing the complete test plan?

(c) The managers for not having checked the test plan?

(d) The client for not having done a thorough acceptance test?

This example comes from Pleeger, "Software Engineering theory and Practice".

Professional Responsibility: Software Developers and Testers

- Carrying out **assigned tasks** thoroughly and in a professional manner
- Being committed to the **entire project** -- not just tasks that have been assigned
- Resisting pressures to **cut corners** on vital tasks
- **Alerting** colleagues and management to potential problems

Professional Responsibility: Computing Management

- Organization **culture** that expects quality
- Appointment of suitably **qualified people** to vital tasks (e.g., testing safety-critical software)
- Establishing and overseeing the software development **process**
- Providing **time** and **incentives** that encourage quality work
- Working closely with the **client**

Accepting responsibility for work of team

Client Responsibility

- Organization **culture** that expects quality
- Appointment of suitably **qualified people** to vital tasks (e.g., technical team that will build a critical system)
- Reviewing **requirements** and **design** carefully
- Establishing and overseeing the **acceptance** process
- Providing **time** and **incentives** that encourage quality work
- Working closely with the **software team**

Accepting responsibility for the resulting product

Conclusion: Professional Responsibility

Organizations put trust in software developers

Competence:

Software that does not work effectively can destroy an organization.

Confidentiality:

Software developers and systems administrators may have access to highly confidential information.

Legal environment:

Software exists in a complex legal environment.

Acceptable use and misuse:

Computer abuse can paralyze an organization.