Cornell University
Computing and Information Science

CS 5150 Software Engineering

Software Development in Practice

William Y. Arms

Overall Aim of the Course

We assume that you are technically proficient. You know a good deal about computing, can program reasonably, can learn more on the job.

But success or failure in software development depend on many factors beyond writing good code.

When you leave Cornell, you are going to work on production projects where success or failure may cost millions of dollars.

Soon you may be in charge. It may be your money.

We want you to make your mistakes now and learn from your mistakes.

Previous Experience (Mine)

Much of my career, I was in charge of computing at universities such as Dartmouth and Carnegie Mellon, with some time in industry.

Projects where I was in charge

- Operating system, compilers, etc.
- Campus networks, routers, protocols, etc.
- Distributed computing environment, file systems, etc.
- Administrative data processing, general ledger, etc.
- Digital libraries

Theme has been first production system where the methods have previously been used only in research.

During this course I will describe some of these projects (both successes and failures) from the viewpoint of a senior manager.

Course Theme: Variety

Software products are very varied

Data processing:	telephone billing, pensions
Real time:	air traffic control
Embedded software:	device drivers, controllers
Mobile devices:	digital camera, GPS, smart phone
Information systems:	web sites, digital libraries
Sensors:	weather data
System software:	operating systems, compilers
Communications:	routers, telephone switches
Offices:	word processing, video conferences
Scientific:	simulations, weather forecasting
Graphical:	film making, design

etc., etc., etc., ...

Variety

The craft of software development

Software products are very varied

- Client requirements are very different
- There is no standard process for software engineering
- There is no best language, operating system, platform, database system, development environment, etc.

A skilled software developer knows about a wide variety of approaches, methods, tools. The **craft** of software development is to select appropriate methods for each project and apply them effectively.

What is Good Software?

What is good software?

General characteristics

- Functionality
- Usability
- Maintainability
- Dependability
- Efficiency

Good software products require good programming,

but ...

Programming quality is the means to the end, **not the end** itself.

Course Theme: Software is Expensive

Software is expensive

The major costs are:

- salaries (your salaries)
- organizational change

Software is Expensive

Who is paying the money?

What does that person or organization want?

- What is success?
- What is failure?

Technical people may have very different criteria of success from the people in charge of the organization.

Example

- Early Unix computers, Sun and IBM

The Three-way Trade-off

Competing goals

Every software project has a trade-off between **functionality**, **cost**, and **time**.

Extra functionality adds extra costs for development, testing, maintenance, etc.

What is important to the person who is paying?

Examples:

- Ship date for Dartmouth financial system
- Console monitor
- Start-up companies: What features will generate real customers who will pay for the product?
- Car anti-lock brakes (no bugs allowed)
- Web browser in cell phone (no delays in release allowed)

Course Theme: Clients, Customers, and Users

Client

The client is the person for whom the software development team creates the software.

- The client provides resources and expects some product in return.
- The client is often a member of the organization that is providing the money.
- The client's job success may depend on the success of the software project.

Client satisfaction is a primary measurement of success in a software project.

Who is the client for Microsoft Excel?

Who is the Client?

Categories of client and software product:

- Bespoke (customized) (e.g., IRS internal system)
- Customized versions of generic packages (e.g., Cornell's payroll system)
- General purpose package (e.g., Microsoft Excel, Mathematica)
- Embedded (e.g., cell phone)
- Research (e.g., Web Lab)

For each category of product:

Who is the client?
Who is providing the money?
What do they want?

Clients, Customers, and Users

Customer

The customer is the person who buys the software or selects it for use by an organization.

User

A user is a person who actually uses the software.

- With personal software, the user may be the same person as the customer.
- In organizations, the customers and the users are usually different.

The Cornell Finance Office uses Microsoft Excel. Who is the customer? Who are the users?

Course Theme: Risk

Many (probably most) software development projects run into difficulties

Problems:

- Does not work as expected (FUNCTION)
- Over budget (COST)
- Late delivery (TIME)

Much of software is wasted (perhaps 50% is never used)

Never used because:

- Does the wrong thing
- Users dislike to use it
- There are no customers
  - *etc.*

Risk

Many software projects fail because the software developers build the wrong software.

The software development team must:

- Understand what the client expects of the software
- Understand what the client's organization expects of the client

The software development team will often:

- Add technical insights and suggestions, but remember:

Client satisfaction is a primary measurement of success in a software project.

Risk

Failures of software development projects can bankrupt companies.

What is the penalty **to the client** if software is:

late?
over budget?
does not work or full of bugs?

Failures of a software development project will often cost senior executives their jobs.

Example: Apple's mapping app.

Minimizing Risk: Communication with the Client

- **Feasibility studies** (whether to begin a project).
- **Separation** of **requirements** (what the client wants) from **design** (how the developers meet the requirements).
- **Milestones** (how the developers report or demonstrate progress to the clients) and **releases**.
- **Acceptance** (how the client tests that the software meets the requirements) and **user testing**.
- **Handover** (ensuring that the client receives a package that can be operated and maintained over a long time period).

Course Themes: Visibility

Visibility

The people who take the responsibility must know what is happening

The problem (as seen by a manager)

Must rely on others for reports of progress or difficulties

... *but s*oftware developers

- Have difficulty evaluating progress
- Are usually optimistic about progress
- Consider reporting a waste time

etc.

Working software provides excellent visibility.

You will make regular progress reports on your projects

Minimizing Risk: Short Development Cycles

Short development cycles with frequent releases provide an important methodology that is used by a number of successful companies.

Risk is minimized by frequent delivery of working software (weeks rather than months).

- Client, customers, and users can evaluate the developers' work.
- Opportunities to adapt to changing circumstances.

This is one of the basic principles of Agile Software Development.

Course Theme: Teams

Most software development is by **teams**

- Effectiveness of team determines success

Most large software projects are **built on older ones**

- It is rare to start a new system from scratch
- Building on the work of others is a fundamental skill of software development
- Much software is built in increments, with different teams responsible for the increments

Course Themes: Scale

Large and very large systems have different needs

- Software design
  - Software architecture
  - Object-oriented design
- Dependable systems
  - Reliability
  - Verification
- Legacy systems

Scale

CS 5150 Projects

- A CS 5150 project is about 0.3 person/years.

This is about the size of a single increment (sprint) in a production Agile process.

A big project may be 100 to 10,000+ person years

- Every large system is developed by many people, who are constantly changing.
- Before a big project is completed the requirements have changed many times.
- No large system is ever complete.
- Nobody comprehends more than a fraction of the project.

Course Themes: Process

Process in large software projects

- Software as a product

  Quality, performance, usability
  Maintenance, evolution

- Separation of requirements and design
- Project management

  Personnel management
  Economic, legal, and social factors

- Development processes

  Sequential
  Iterative refinement
  Incremental (Agile)

Course Theme: 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 (e.g., trade secrets, personal data).
- Legal environment: Software exists in a complex legal environment (e.g., intellectual property, obscenity).
- Acceptable use and misuse: Computer abuse can paralyze an organization (e.g., the Internet worm).

Academic Integrity & Professional Practice

Software Engineering is a collaborative activity. You are encouraged to work together, but ...

- Some tasks may require individual work.
- Always give credit to your sources and collaborators.

Good professional practice: To make use of the expertise of others and to build on previous work, with proper attribution.

Unethical and academic plagiarism: To use the efforts of others without attribution.

See: **Academic Integrity** on the course web site, which points to the Cornell code.