Introduction to Software Engineering

What is a Software development methodology?

Who cares?

Why should we care?
Process of building a house

Building software is engineering
A Taste of Reality

What customers really need
What customers think they want
What programmers think
What design looks like
What gets implemented
What gets installed
What sales people say
What customers get charged

Customers Care!
A software project has failed if

• it is delivered late
• it runs over the budget
• it does not satisfy the customer’s needs
• it is of poor quality
A look at history

• Software crisis, i.e., software is delivered
  – late
  – over budget
  – with residual faults
• 1968 NATO Conference
  – endorse the term “Software Engineering”
  – aim… use the philosophies and paradigms of established engineering disciplines to solve *software crisis*

Economic Aspects

• Coding method $CM_{new}$ is 10% faster than currently used method $CM_{old}$.
  – Should it be used?
• Common sense answer
  – Of course!
• Software Engineering answer
  – Consider the cost of training
  – Consider the impact of a new technology
  – Consider the effect on maintenance
Team Programming Aspects

• Hardware is cheap
• Software is built by teams
  – Products that are too large to be written by one person in the available time
  – Interfacing problems between modules
  – Communication problems among team members
Software Life-Cycle Aspects

• Classical/Heavy Weight Software Development Life-Cycles

• Agile/Light Weight Software Development Life-Cycles
Typical Classical Life-Cycle

1. Planning
   - Review project requests
   - Prioritize project requests
   - Allocate resources
   - Identify project development team

2. Analysis
   - Conduct preliminary investigation
   - Perform detailed analysis activities:
     - Study current system
     - Determine user requirements
     - Recommend solution

3. Design
   - Acquire hardware and software, if necessary
   - Develop details of system

4. Implementation
   - Develop programs, if necessary
   - Install and test new system
   - Train users
   - Convert to new system

5. Support
   - Conduct post-implementation system review
   - Identify errors and enhancements
   - Monitor system performance

Ongoing Activities
   - Project management
   - Feasibility assessment
   - Documentation
   - Data/information gathering
Standish group data

(Data on 28,000 projects completed in 2000)

- Canceled: 23%
- Successful: 28%
- Completed late, over budget, and/or with features missing: 49%

CHAOS RESOLUTION

- Successful: 43%
- Failed: 43%
- Challenged: 1.8%

Project resolution from 2012 CHAOS research.
Cutter consortium data

- 2002 survey of IT organizations
  - 78% have been involved in disputes ending in litigation
- Among those that entered into litigation:
  - the functionality delivered did not meet up to the claims of the developers (67%)
  - the promised delivery date slipped several times (56%)
  - the defects were so severe that the information system was unusable (45%)
Causes of defects and their origin

- Data handling: 11%
- Standards: 7%
- Specifications: 25%
- User interface: 12%
- Error checking: 11%
- Hardware interface: 8%
- Software interface: 6%
- Logic: 20%

Conclusion

Classical software development methods have not solved the software crisis.
Agile Software Development Methodology

To satisfy the customer through early and continuous delivery of valuable software

Agile Software Development Methods - Review and Analysis

Manifesto for Agile Software Development

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham

Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern

Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Evolution of Software Development Methods

From nothing,

to monumental,

to agile
Last Lecture given by a CS Professor at Carnegie Mellon University about Achieving Your Childhood Dream after his doctor told him that he could only live for 6 months

3D Programming Language Alice

Virtual Reality

Randy Pausch
You’ve spoken of the importance of never quitting – of continually pushing against brick walls and other obstacles. What additional advice might you give to tomorrow’s CS student:

*Remember how quickly our field changes. That’s why you want to focus on learning things that don’t change:*

- how to work well with other people
- how to carefully assess a client’s “real” – as opposed to perceived - needs

What about advice for CS teachers and professors?

*It’s time for us to start being more honest with ourselves about what our field is and how we should approach teaching it.*

Personally, I think that if we had named the field “Information Engineering” as opposed to “Computer Science”, we would have had a better culture for the discipline.

For example, CS departments are notorious for not instilling concepts like testing and validation the way many other engineering disciplines do.
Is there anything you wish someone had told you before you began your own studies?

*Just that being technically strong is only one aspect of an education.*

### Key points to remember

- Building software is an engineering process
- Characteristics of failed software project
- Different aspects of software engineering
  - Economic
  - Team
  - Life-cycle (classical and agile)