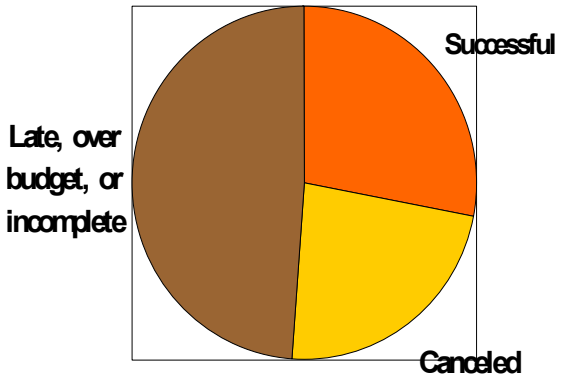
CSE 308

Software Engineering Process

Software Project Outcomes



2003 study of 280,000 SW projects

2

Software Engineering

- Analysis: Understand the nature of the problem and break the problem into pieces
- Synthesis: Put the pieces together into a large structure
- For problem solving we use
  - Techniques (methods) - formal procedures for producing results using some well-defined notation
  - Methodologies - collection of techniques applied across software development and unified by a philosophical approach
  - Tools - instrument or automated systems to accomplish a technique

3

Software Engineering: Definition

Software Engineering is a collection of techniques, methodologies and tools that help with the production of:

- a high quality software system,
- with a given budget, and
- before a given deadline,

while change occurs.

What is your budget and deadline?

How do you measure the quality of your project?

4

Factors Affecting Quality of Software

- Complexity:
  - The system is so complex that no single programmer can understand it anymore
- Change:
  - The "Entropy" of a software system increases with each change: Each implemented change erodes the structure of the system
  - As time goes on, the cost to implement a change will be too high, and the system will then be unable to support its intended task.

5

Why are Software Systems so Complex?

- The problem domain is difficult
- The development process is very difficult to manage
- Software offers extreme flexibility (limited capital investment constraints)

Usually leads to tight (impossible?) deadlines.

6

Dealing with Complexity

1. Abstraction
2. Decomposition
3. Hierarchy

7

1. Abstraction

- Inherent human limitation to deal with complexity
  - The 7 +- 2 phenomena
- Chunking: Group collection of objects
- Ignore unessential details: => Models

How might you break your project into components?

Flow charts were an important abstraction at one time - but no longer

8

Models Provide Abstractions

- System Model:
  - Object Model: What is the structure of the system? What are the objects and how are they related?
  - Functional model: What are the functions of the system? How is data flowing through the system?
  - Dynamic model: How does the system react to events?
- Task Model:
  - PERT Chart: What dependencies are between tasks?
  - Schedule: How can this be done within the time limit?
  - Org Chart: What is the organization of your team?
- Issues Model:
  - What are the open and closed issues? What constraints were posed by the client?

9

Overview

- Build-and-fix model
- Waterfall model
- Rapid prototyping model
- Incremental model
- Extreme/Agile
- Synchronize-and-stabilize model
- Spiral model

Some texts use the term "prescriptive process models"

Model to use is based on SW policies of your company

Details of a given model are less important than common components of models

10

Software Life-Cycle Models

- Life-cycle model (formerly, process model)
- The steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

These phases usually overlap

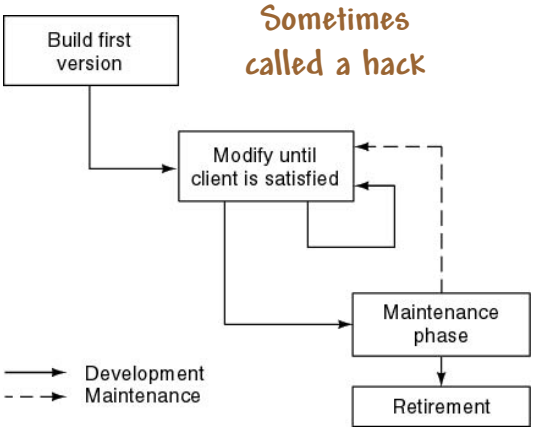
The problems are:

1. Mistakes can be made during the early phases and
2. The user can change his/her mind
3. "Feature creep"

11

Build and Fix Model

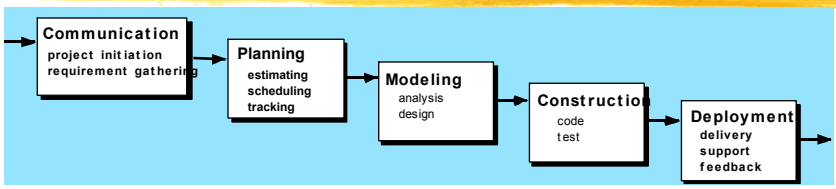
- Problems
  - No specifications
  - No design
- Totally unsatisfactory
- Need life-cycle model
  - "Game plan"
  - Phases
  - Milestones



Sometimes called a hack

12

Waterfall Model



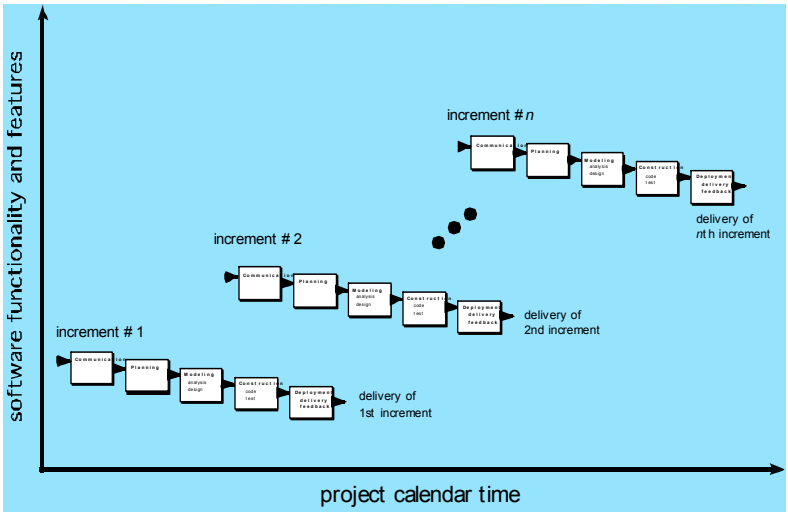
- Characterized by
  - Feedback loops
  - Documentation-driven
- Advantages
  - Documentation
  - Maintenance easier
- Disadvantages - specs

Feedback loops are rarely seen when this process is used

Imagine buying a car by only reading the specs

13

The Incremental Model



14

Incremental Model

- Divide project into *builds*
- System is built incrementally
- Testing after each build
- Customer review (and use) is possible for incremental builds
- Customer evaluation of a build may result in a change in requirements (for a subsequent build)
- Each build usually adds implementations of additional use cases

Incremental model provides financial flexibility

You should be almost ready to determine your project increments

15

Synchronize-and Stabilize Model ...

- Microsoft's life-cycle model
- Requirements analysis—interview potential customers
- Draw up specifications
- Divide project into multiple builds
- Each build is carried out by small teams working in parallel

16

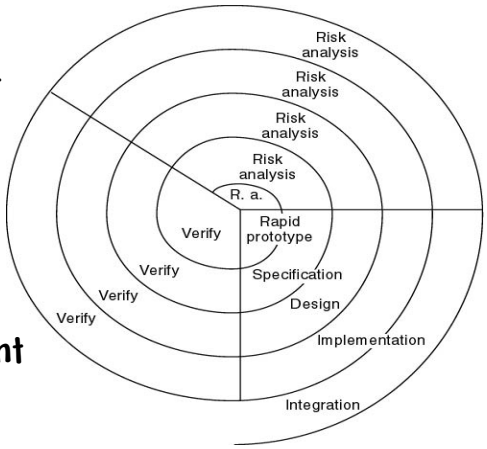
... Synchronize-and Stabilize Model

- At the end of the day—synchronize (test and debug)
- At the end of the build—stabilize (freeze build)
- Components always work together
  - Get early insights into operation of product

17

Simplified Spiral Model

- Use of engineering prototypes (different from requirements prototypes)
- Risk management requires accurate team self-assessment



18

Extreme/Agile Development

- More recent (and popular) approach
- Stories (features client wants)
- Estimate duration and cost of each story
- Select stories for next build
- Each build is divided into tasks
- Test cases for task are drawn up first
- Pair programming
- Continuous integration of tasks

19

Unusual Features of Agile/XP

- Client representative is present
- Early delivery of partial system
- Series of builds
- Refactoring

Not much quantitative data, but popular for smaller projects

20

Conclusions

- Different life-cycle models
- Each with own strengths
- Each with own weaknesses
- Criteria for deciding on a model include
  - The organization
  - Its management
  - Skills of the employees
  - The nature of the product (size, scope, complexity)
- Best suggestion
  - "Mix-and-match" life-cycle model

21