1.264 Lecture 2

Software process

Case study 2-1: rapid development

- Volunteer to summarize the case briefly
  - Mickey’s side
  - Kim’s view
- What’s wrong with providing motivation and letting the programmers go?
  - They’re skilled professionals, well trained in software
  - Many companies use motivation as the primary management method (e.g., sales commission, bonus); why not here?
- What large company operates like this in a fundamental way?
- What factors led to the project taking so long?

Dimensions of development speed

- People
  - Matter most: ability and motivation
- Process
  - Customer focus
  - Fundamentals, QA, risk management, lifecycle planning
  - "Code like hell" and chaos are still the most common approaches
- Product
  - Size and characteristics, phasing
- Technology
  - Product or software development environment
  - Tools

Programming practices

Doing a few things right is not enough
You have to avoid doing anything wrong

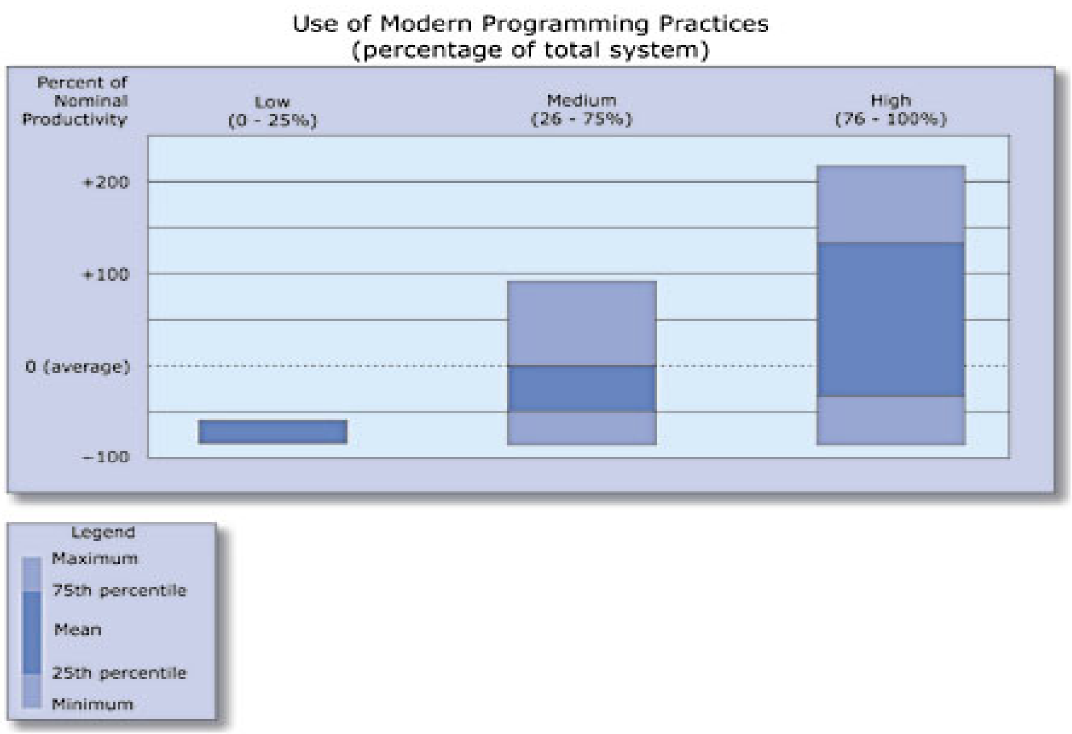

Figure by MIT OCW.

CLASSIC MISTAKES			
People-Related	Process-Related	Product-Related	Technology-Related
Heroics	Contractor failure	Feature creep	Silver-bullet syndrome
Weak personnel	Inadequate design	Developer gold-plating	Lack of automated source-code
Wishful thinking	Insufficient planning	Requirements gold-plating	Switching tools in the middle of a project
Lack of user input	Planning to catch up later	Push-me, pull-me negotiation	Overestimated savings from new tools or methods
Undermined motivation	Overly optimistic schedules	Research-oriented development	
Noisy, crowded offices	Code-like-hell programming		
Unrealistic expectations	Insufficient risk management		
Lack of stakeholder buy-in	Abandonment of planning under pressure		
Politics placed over substance	Shortchanged upstream activities		
Adding people to a late project	Insufficient management controls		
Lack of effective project sponsorship	Wasted time during the fuzzy front end		
Friction between developers and customers	Premature or overly frequent convergence		
Uncontrolled problem employees	Omitting necessary tasks from estimates		
	Shortchanged quality assurance		

Figure by MIT OCW.

Case study 2-2: rapid development

- Volunteer to summarize the case briefly
  - Sarah’s perspective
  - Eddie’s perspective
- Why did Sarah’s picking a team work? Why didn’t they bicker and delay the project?
- How did they use technology to help them?
- What time and resources were needed in this case study versus 2-1?
  - Is the difference small or large? Surprising?

Case study 3-1: Classic mistakes

- Volunteer to summarize case study
  - Bill's view
  - Mike's view
- What were the classic mistakes made?
  - When each is mentioned, please raise your hand if you've seen it or been involved in it yourself sometime
- Does this management method work in some cases?
  - What happens with outside vendors?

Case study 4-1: Fundamentals

- Volunteer to summarize case study
  – Bill's perspective
- What would you do to avoid these errors again?
  – What is the most important single thing?
  – Next most important thing?
- Do you think the next project could be successful at this company? Why or why not?

Technical fundamentals

Spiral model as basis for development

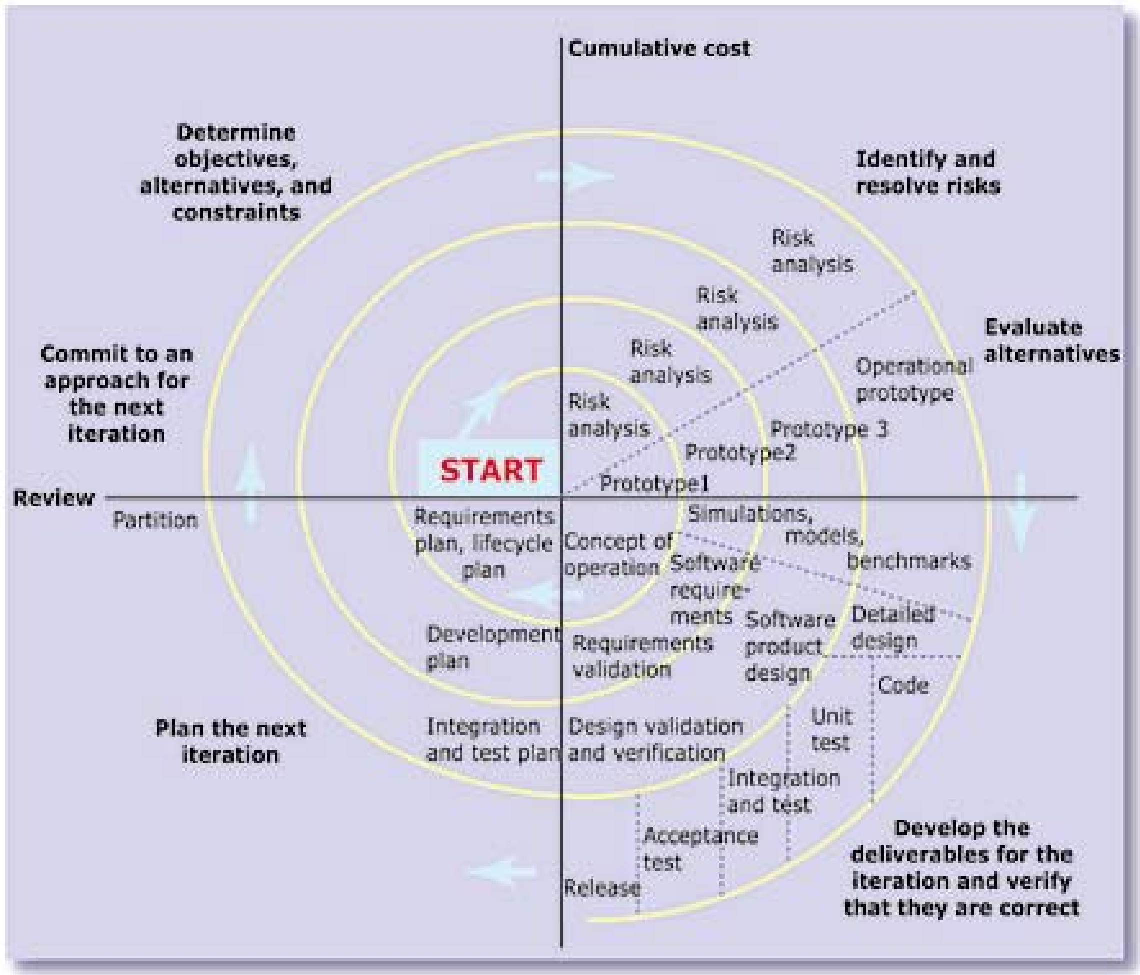

Figure by MIT OCW.

Requirements fundamentals

- Requirements: what does the system do?
  - What we're doing in homeworks this semester is essentially an extended requirements analysis (and first spiral)
    - Homework 2 is a very preliminary stage in requirements analysis
  - Analysis using Unified Modeling Language (UML)
    - Data modeling (entity-relationship diagrams, covered next week) (not a part of UML, strictly speaking)
      - Static but complete picture of all data (or objects) in the system
    - Class diagrams
      - Static but complete picture of relationships among all objects
    - Sequence and collaboration models (dataflow diagrams)
      - Dynamic view of multiple flows of data and control in the system
    - State models (state transition diagrams)
      - Dynamic and complete view of the data values and logic
    - Data dictionary

Design fundamentals

- Design: how does the system work?
  - Object and database design (next week)
  - System architecture (late in term)
    - Subsystems, layering/libraries, communications
  - System design principles
    - Internationalization, portability, performance, memory, availability, reliability, … (also called non-functional requirements)
    - Tools selection: database, middleware, application, monitoring, …

Development fundamentals

- Requirements and design dictate development success
  - 60% of defects exist at design time
- Coding practices
- Unit testing and debugging practices
- QA practices: requirements, design, code reviews
- Integration strategies: incremental
- Code tuning and performance
- CASE (computer aided software engineering) tools
  - Software configuration management (Visual Source Safe, others)
  - UML, data models

Quality assurance fundamentals

- Testing
  - Unit tests find 10-50% of defects
  - System tests find 20-60% of defects
  - Reviews and inspections find 60-90%: more critical than testing
- Error prone modules: identify and rewrite
  - 57% of errors in 7% of modules (IBM)
  - If more than 10 defects per 1000 lines of code, rewrite
- QA starts at project initiation
  - Requirements scrubbing and reviews
  - Design reviews
  - Code inspections: inspector, author, scribe

Defect prevention and removal

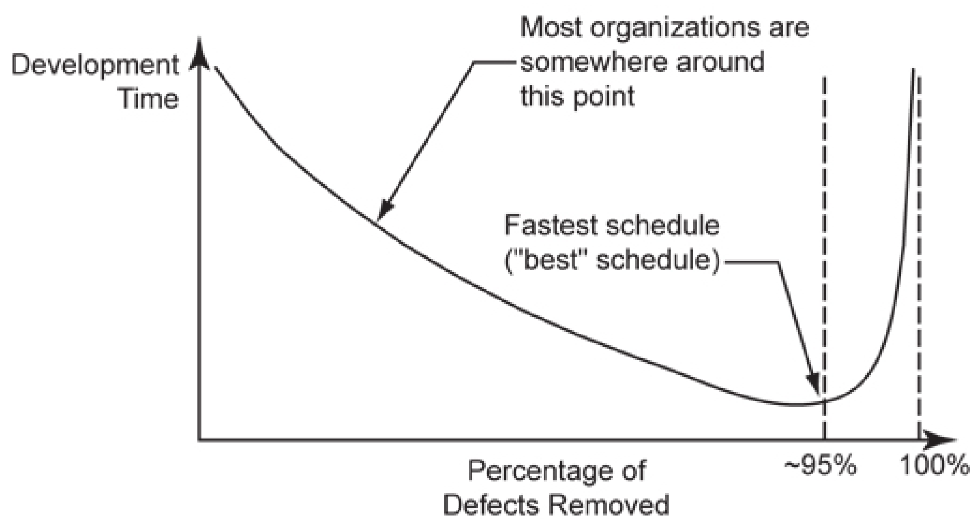

Figure by MIT OCW.

If you find more than 5% of bugs after release, you are in trouble

Superpages.com

- SuperPages (Web yellow pages)
  - Prototype Fall 1994 not production, many shortcuts
  - Decision to launch December 1994
  - Marketing set date of March 1995 for launch. I became manager in January 1995
  - Server team:
    - Prototypers, very creative, very opinionated
    - Object database, text retrieval engine, Netscape Commerce Server, C+
    - Production design too sophisticated (advanced C++ features), only 2 people could code it (the 2 designers)
    - Long hours, schedule pressure, many complaints from 2 developers
    - No time for written down design or reviews, per the developers
  - Back office team
    - At least telcos know that back office is half or more of picture
    - Relational database, C code, very conservative
    - Team could scale up with junior developers; substantial mentoring and review by default

Superpages.com

- Shaky launch in April 1995
  - Based on prototype code: frequent crashes, poor performance (but ok given low initial usage)
  - Hardware not installed on time
  - Massive data problems: paper yellow pages data not ideal
- Project restructuring
  - Moved best back office developer to lead server team; made most experienced developer the technical lead
  - Moved project into one dept
  - Two server developers transferred to other projects ('wage slave', Bible quotes)
  - User interface designers very temperamental (psychologists), but really matured
- Unable to change overall architecture until July 1996
  - One year of quarterly releases of added functions based on prototype code
  - Put requirements process in place, required quarterly plan, quick design, code estimates
  - No QA group; assigned junior developers to QA in server and back office groups
- Competition: BigYellow (NYNEX: 20 PCs), BigBook (no business model, nice UI), other RBOCs, directory companies

Case study 5-1: risk management

- Volunteer to summarize case study
  - Kim’s view
- Risk management is managing items not on the classic mistakes list.
  - Is this case study risk management, or classic mistakes?
  - Look at the classic mistakes list on the previous slide

Case study 5-2: risk management

- Volunteer to summarize case study
  - Eddie’s view
- How much difference did it make that the previous project’s staff was still in place?
- What requirements, design, implementation and test process was used?
  - Why do you think it was sufficient?

Case study 7-1: software process selection

- Problems:
  - Prototyping regarded as ‘fastest’ approach, without considering context
  - Idiot manager, idiot consultant (common combination)
  - Out of control requirements process
    - Field agents calling tech lead directly with ideas

Case study 7-2: software process selection

- Good practices:
  - Careful project size, resource estimation to select proper scope and focus
  - Spiral development model chosen to manage risk
  - Risk management in requirements, staffing, choice of approach (safe approach, but design ahead for risky features)
  - Simple implementation of most features