Lecture 2

Software Development (II)

Review

- Course introduction
- Process of small-scale software development
  - R → A → D → I → TV
- Strategies toward rapid development
  - Avoid class mistakes, Apply development fundamentals, Risk management, Schedule-orientation
- Four dimensions of software development
  - P, *P, P, T*

People dimension

- Peopleware
  - Different people have different productivity
- Staff selection
  - Top talent, job matching, career progression, team balance, misfit elimination, other.
- Team organization & motivation
  - Matching: project size, schedule goal, product attribute
  - Motivation (is the hidden driving force)

Today's Topic

- Rapid Software Development Strategy
  - *Four dimensions of development: People,* Process, Product, Technology
- Class-mistake avoidance
- Development fundamentals
  - Management fundamentals
  - Technical Fundamentals
  - Quality-Assurance Fundamentals

Process (1/3)

- Management and Technical Methodologies
- Process represents an area of high leverage in improving development speed
- Hughes aircraft, Lockheed, Motorola, NASA, Raytheon, and Xerox
  - explicitly focused on improving their development process
  - Cut time-to-market by about one-half
  - Reduced cost and defects by factors of 3 to 10

Process (2/3)

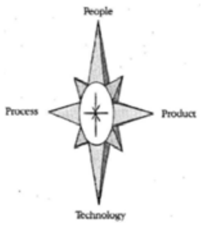
- Rework avoidance
  - Avoid requirement change at late stages
- Quality Assurance (Q/A)
  - Assure the product has an acceptable level of quality
  - Detect errors at the stage when they are least time-consuming (and least costly) to correct. → catch errors as close as possible to the time that they are introduced
- Development fundamentals
  - Analysis → design → construction → integration → testing will not product lightning-fast schedules, but they prevent disasters
  - Half of the challenge of rapid development is avoiding disaster.

Process (3/3)

- Risk management
- Resource Targeting
  - Get the most bang for your buck
- Lifecycle planning
  - Several lifecycle models to be introduced later
- Customer orientation
  - Develop software to its spec. is half job done
  - The other half is help the customer figure out what the product should be (Thus, requirement specification is very important).

Four Dimensions of Development

People
Process
Product
Technology



Product

- The most tangible dimension
  - product size
    - 80/20 rule
    - Additional features require additional specification, design, construction, testing, and integration
    - 1/2 produce size → 2/3 effort saving
  - product characteristics
    - Performance, memory footprint, robustness, reliability
    - Don't insist on too many priorities at once!

Technology

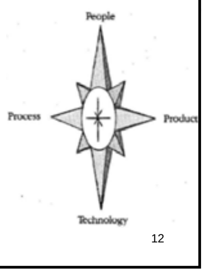
- Platform, operation system, software components, development tools, …
- From assembly → high-level languages was one of the most influential changes in software-development history
- Integrated Development Environment, IDE
  - E.g. Microsoft Visual Studio, Eclipse, Borland C++ builder, …
  - Help manages the complexity of software projects when there are many classes and methods.
- Visual Programming
  - programming can be done by drag-and-drop without writing a single line of code

Summary for 4 dimensions

- People
- Process
- Product
- Technology

Which dimension matters the most

- Different projects have different needs
  - Accept the limitations on the dimensions you cannot change
  - Emphasize the other dimensions to get the rest of the schedule benefit you need
- Examples
  - Fuel-injection system for a car
  - In-house business program
  - A feature driven shrink-wrap market



Characteristics of standard approaches to schedule-oriented development

	Effect of Development Approach On		
Development Approach	… Schedule	… Cost	… Product
Average practice	Average	Average	Average
Efficient development (balancing cost, schedule, and functionality	Better than average	Better than average	Better than average
Efficient development (tilted toward best schedule)	Much better than average	Somewhat better than average	Somewhat better than average
All-out rapid development	Fastest possible	Worse than average	Worse than average

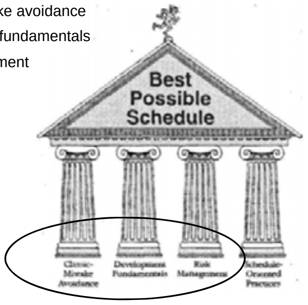
All-out rapid development: work as fast as you can, as hard as you can, and the only thing left to do at that point is to pay more, reduce the feature set, or reduce the product's polish.

compress schedule by 25% requires an 75% increase of team size → 33% cost increase

Source: RD

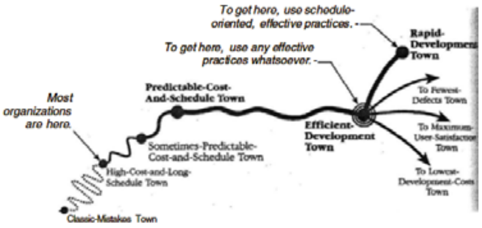
Efficient Development

1. Classic-mistake avoidance
2. Development fundamentals
3. Risk management



Source: RD

The Road to Rapid Development


To get here, use schedule-oriented, effective practices.
Rapid-Development Town
To get here, use any effective practices whatsoever.
Most organizations are here.
Predictable-Cost-And-Schedule Town
Efficient-Development Town
To Fewest-Defects Town
To Maximum-User-Satisfaction Town
To Lowest-Development-Cost Town
Sometimes-Predictable-Cost-and-Schedule Town
High-Cost-and-Long-Schedule Town
Classic-Mistakes Town

Source: RD

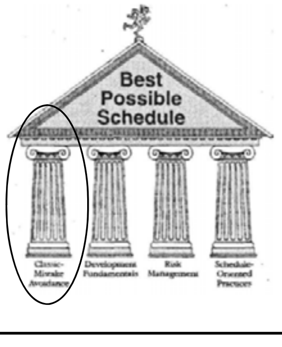
An Alternative Approach (1/2)

- How
  - Hiring the best people
  - Total commitment to the project
  - Full autonomy
  - Motivating them to an extreme degree → work 60, 80, or 100 hours a week until the completion of the project

An Alternative Approach (2/2)

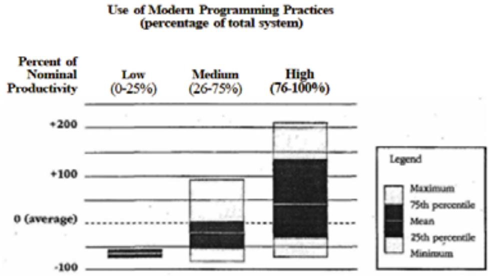
- Usually code-like-hell approach
- Hit-or-miss
- Causes long-term motivation problems
- Unrepeatable
- Hard on non-software organizations
- Wastes human resources extravagantly

Classic Mistakes



RD, Chapter 3

Do a few things right isn't enough



Data based on 44 projects from 9 countries

Need to avoid making any big mistakes !

People

1. Undermined motivation
2. Weak personnel
3. Uncontrolled problem employees
4. Heroic
5. Adding people to a late project
6. Noisy, crowded offices
7. Friction between developers and customers
8. Unrealistic expectations
9. Lack of effective project sponsorship
10. Lack of stakeholder buy-in
11. Lack of user input
12. Politics placed over substance
13. Wishful thinking

Process

14. Overly optimistic schedules
15. Insufficient risk management
16. Contractor failure
17. Insufficient planning
18. Abandonment of planning under pressure
19. Wasted time during the fuzzy front end
20. Shortchanged upstream activities
21. Inadequate design
22. Shortchanged quality assurance
23. Insufficient management controls
24. Premature or overly frequent convergence
25. Omitting necessary tasks from estimates
26. Planning to catch up later
27. Code-like-hell programming

Product

28. Requirement gold-plating
29. Feature creep
30. Developer gold-plating
31. Push-me, pull-me negotiation
32. Research-oriented development

Technology

33. Silver-bullet syndrome
34. Overestimated savings from new tools or methods
35. Switching tools in the middle of a project
36. Lack of automated source-code control

Development Fundamentals

- Development Fundamentals
  - Management fundamentals
  - Technical Fundamentals
  - Quality-Assurance Fundamentals

Development Fundamentals

Management Fundamentals

Estimation and Scheduling
Planning
Tracking
Measurement

Management Fundamentals (1/4)

Estimation and Scheduling

- Estimate the size of the project
- Estimate the effort needed
- Estimate the schedule

Management Fundamentals (2/4)

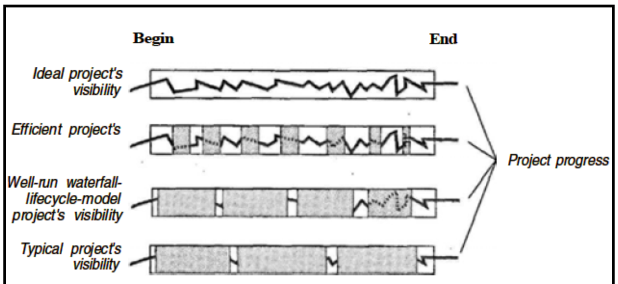
Planning

- Estimation and scheduling
- Determine how many people to have on the project team, what technical skills are needed, when to add people, and who the people will be
- Deciding how to organize the team
- Choosing which lifecycle model to use (to be discussed)
- Managing risks
- Making strategic decisions such as how to control the product’s feature set and whether to buy or build pieces of the project.

Management Fundamentals (3/4)

Tracking

- Meeting its schedule, cost, and quality targets
- Management-level tracking: Task lists, status meetings, status reports, milestone reviews, budge reports
- Technical-level tracking: technical audits, technical reviews, and quality gates that control whether you consider milestones to be complete


Begin
End
Ideal project's visibility
Efficient project's
Well-run waterfall-lifecycle-model project's visibility
Typical project's visibility
Project progress

•SEI found 75% of 59 sites assessed between 1987 – 1993 needed to improve their project tracking and oversight (Kitson and Masters 1993)
•Tracking is a fundamental software management activity. If you don't track a project, you can't manage it.

Management Fundamentals (4/4)

Measurement

- Collecting metrics data to analyze software quality and productivity
- Collect historical project size (e.g. lines of code) for future planning reference

Development Fundamentals

Technical Fundamentals

Requirement management
Design management
Software configuration management

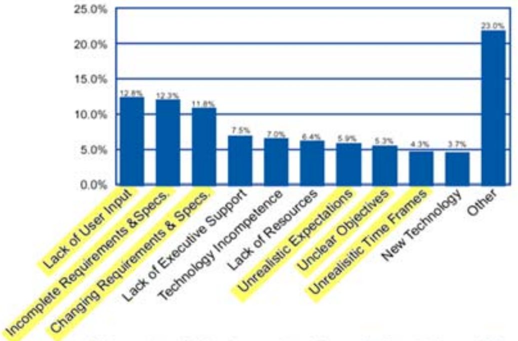
Technical Fundamentals (1/3)

1. Requirement Management (1/2)

- Requirement Management
- The process of gathering requirements;
  - recording them in a document, email, user-interface storyboard, executable prototype, or some other form;
  - tracking the design and code against them;
  - managing changes to them for the rest of the project.
- The top three reasons that projects were delivered late, over budget, and with less functionality than desired all had to do with requirements-management practices:
  - lack of user input
  - incomplete requirements
  - changing requirements

Why Software Projects Fail

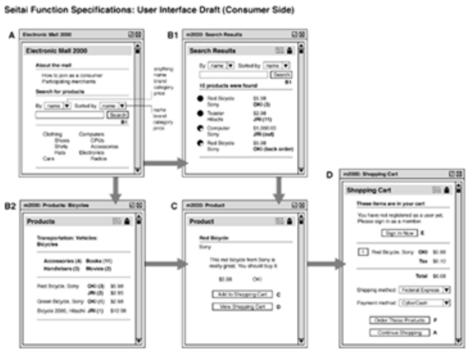
- Average overrun: 89.9% on cost, 121% on schedule, with 61% of content



352 companies - 8,000 software projects. Source: *The Standish Group, 1995*

Boehm (2006), "A View of 20th and 21st Century Software Engineering", ICSE 2006 Keynote Address

UI Storyboard



http://www.kevcom.com/software/veosystems/ui.gif

Technical Fundamentals (1/3)

1. Requirement Management (2/2)

- Requirements-analysis methodologies including structured analysis, data structured analysis, and object-oriented analysis, …
- System-modeling practices such as *class diagrams*, dataflow diagrams, *entity-relationship diagrams*, data-dictionary notation, and state-transition diagrams
- Communication practices such as Joint Application Development (JAD), user-interface prototyping, and general interview practices
- The relationships between requirements management and the different lifecycle models including evolutionary prototyping, staged releases, spiral, waterfall, and code-and-fix (to be discussed)

Technical Fundamentals (2/3)

2. Design Management (1/2)

- Major design styles (such as object design, structured design, and data structure design)
- Foundational design concepts (such as information hiding, modularity, abstraction, encapsulation, cohesion, coupling, hierarchy, inheritance, polymorphism, basic algorithms, and basic data structures)
- Standard design approaches to typically challenging areas (including exception handling, internationalization and localization, portability, string storage, input/output, memory management, data storage, floating-point arithmetic, database design, performance, and reuse)

Technical Fundamentals (2/3)
2. Design Management (2/2)

- Design considerations unique to the application domain you're working in (financial applications, scientific applications, embedded systems, real-time systems, safety-critical software, or something else)
- Architectural schemes (such as subsystem organization, layering, subsystem communication styles, and typical system architectures)
- Use of design tools

Technical Fundamentals (3/3)
3. Software Configuration

- Evaluating proposed changes
- Tracking changes
- Handling multiple versions
- Keeping copies of project artifacts as they existed at various times.
- Open source solutions
  - CVS (Concurrent Versions System)
    - http://www.gnu.org/software/cvs
  - SVN (SubVersioN)
    - http://subversion.tigris.org/

Assignment #1

Assignment #1
Part I (1/3)

- Form a group **throughout the rest of the term** to work on a software project. (international colaboration pls.)
  - The project is about starting an e-business for **2nd hand textbook** trading!
  - This project involves developing two software systems
    - A two-tier software program for selling and finding used textbooks through telephone.
    - A three-tier web application for direct trading online.
  - Please consider your team a company, thus you need a name!
- In this assignment, you and your partners need to have at least one brainstorming session regarding your **business model**.
  - Here is a link on how to conduct a brainstorming session
  - http://projects.edtech.sandi.net/staffdev/tpss99/processguides/brainstorming.html

Assignment #1
Part I (2/3)

- After the brainstorming session, you need to come up with a business proposal with fancy functions or crazy ideas that you can come up to build a attractive system for people to trade used computer parts(and hopefully you can make a profit out of these transactions).
- As stated previously, we're going to develop two software for doing e-business. Therefore, this assignment (due in one week), needs to include
  - A business proposal for used computer parts trading through telephone and on-line
  - A **software requirement** for the 2-tier standalone application
  - A **software requirement** for the 3-tier web application
  - Your software requirement (at this stage) needs to include all the functionalities that you can think of.

Assignment #1
Part I (3/3)

- Prepare a 10-minute presentation to present your business proposal in the class on **10/05/2011**.
  - Consider your "company" is trying to get funding from venture capitalists/investors to invest your company, so this presentation is really important.
  - Some references on preparing oral presentation:
    - http://www.ee.uconn.edu/SeniorDesign/designone/Lecture%2008%20Proposals%20and%20Oral%20Presentations.ppt

Assignment #1
Part II

- Please read through Chapter 2 and 3 of “Rapid Development”.
- Please try to read Chapter 4, 7, and 8 of Rapid development before the next class.

5 key elements of business proposal

- **Solutions**: After you have written a lead paragraph on the company's needs and problems, follow up with a solid presentation of how your business can provide solutions. The key here is to promise solutions you can deliver.
- **Benefits**: All winning business proposals, clearly outline for the company the benefits to be gained from doing business with you. If your small business can offer complete confidentiality and meet tight deadlines state it in your benefits section.
- **Credibility**: This is often the overlooked portion of a business proposal but all winning proposals glow with credibility. If you have worked with clients in the same field or have an award-winning business, then third-party endorsements will build credibility.
- **Samples**: A business proposal with samples and evidence of your ability to deliver is vital to gaining the winning bid. A small sample of your work can show your ability to do the job.
- **Targeted**: A winning business proposal is all about communication. Speak in a language spoken by your intended audience. If the proposal evaluators are from an engineering background or financial department use the appropriate jargon.

REF: http://sbinformation.about.com/cs/bizlettersamples/a/proposal.htm

A simple proposal formula

- **Who**: who will do the work, who will manage the work, who does the customer call if there is a problem, who is responsible for what
- **What**: what needs to be done/delivered, what will be required to do it, what can the customer expect, what it will cost
- **Where**: where will the work be done, where will it be delivered
- **How**: how will the work be done, how will it be deployed, how will it be managed, how will you achieve quality assurance and customer satisfaction, how will risks be mitigated, how long will it take, how will the work benefit the customer
- **When**: when will you start, when will key milestones be scheduled, when will the project be complete, when is payment due
- **Why**: why have you chosen the approaches and alternatives you have selected, why the customer should select you

REF: http://www.captureplanning.com/articles/11562.cfm

Brainstorming process (1/2)

1. In a small or large group select a leader and a recorder (they may be the same person).
2. Define the problem or idea to be brainstormed. Make sure everyone is clear on the topic being explored.
3. Set up the rules for the session. They should include
   - letting the leader have control.
   - allowing everyone to contribute.
   - ensuring that no one will insult, demean, or evaluate another participant or his/her response.
   - stating that no answer is wrong.
   - recording each answer unless it is a repeat.
   - setting a time limit and stopping when that time is up.

http://projects.edtech.sandi.net/staffdev/tpss99/processguides/brainstorming.html

Brainstorming process (2/2)

4. Start the brainstorming. Have the leader select members of the group to share their answers. The recorder should write down all responses, if possible so everyone can see them. Make sure not to evaluate or criticize any answers until done brainstorming.
5. Once you have finished brainstorming, go through the results and begin evaluating the responses. Some initial qualities to look for when examining the responses include:
   - looking for any answers that are repeated or similar.
   - grouping like concepts together.
   - eliminating responses that definitely do not fit.
   - Now that you have narrowed your list down some, discuss the remaining responses as a group.