Session 12
Managing software development
Approaches to software development

• Traditional systems development life cycle
• Prototyping
• Packaged software
• End-user development
• Outsourcing
• Open source
Traditional systems development life cycle ("waterfall" model)

- **Project definition**: High level requirements
- **Systems analysis**: Detailed requirements
- **Design**: Decomposition into modules
- **Programming**: Writing and testing code
- **Deployment**: Installation and operation (sale)
- **Maintenance**: Fixing bugs and changing features
Traditional systems development life cycle

- **Advantages**
  - For well-understood problems, produces predictable outcomes

- **Disadvantages**
  - Inflexible
  - Long delay before any useful results
    » May be obsolete by then
  - Often hard to know requirements until actual use
Prototyping

- Project definition
- Identify basic requirements
- Develop a working prototype
- Use the prototype
  - User satisfied?
    - Yes
    - No
      - Revise and enhance prototype
      - Deployment Maintenance
Prototyping

- **Advantages**
  - Especially useful when exact requirements are hard to know in advance
    - user interfaces
    - decision systems
    - electronic commerce?
  - Encourages user involvement

- **Disadvantages**
  - Hard to predict and control outcomes reliably
  - If repeated, significant reimplementations are needed, can be very expensive
  - May result in systems that are inefficient, unreliable, or hard to maintain
Packaged software

Project definition

Identify basic requirements

Select package

Customize package

Redesign organizational processes

Train users

Operation
Packaged software

- **Advantages**
  - By amortizing development and maintenance costs over many organizations, it is possible to get superior solutions at much lower cost

- **Disadvantages**
  - Customizing software can be *very* time-consuming and expensive
  - May have to change organization to fit software, rather than vice versa
End-user development

User recognizes need

User acquires development tools

User implements solution

User uses the solution

User satisfied?

Yes

Done

No

User revises and enhances solution
End-user development

• **Advantages**
  – Can be *much* faster  
  – Improved requirements determination  
  – Increased user involvement and satisfaction

• **Disadvantages**
  – Often, users lack the right implementation skills  
  – Many problems can’t be solved within the limitations of the tools  
  – Lack of quality assurance and standards for programs and data  
  – Lack of sharing of programs and data  
  – Reduced opportunity for reuse of results
Outsourcing

• Contract out the performance of any or all of the above steps to another firm

• Advantages
  – Economies of scale
  – Flexibility
  – Predictability
  – Freeing up human resources and capital

• Disadvantages
  – Loss of control
  – Vulnerability of strategic information
  – Dependency
Open source

Someone develops working prototype

Use the prototype

Some developer wants to do more?

Yes

Revise and enhance prototype

No

Project ends
Open source

- **Advantages**
  - Usually lower cost
  - Sometimes easier to adapt “packaged” software to own needs
  - “Philosophically” appealing to many people

- **Disadvantages**
  - Usually lower quality support
  - Only a few kinds of software are currently available in this format (Linux operating system, Apache web server, etc.)
Problems with software development

• **Computerworld magazine***
  – “Nearly one-third of all projects fail”
  – “More than half come in over budget”
  – “Only 16% of all projects come in on time and on budget”

• **Key factor for success or failure:**
  – “User involvement/input”

*Survey of 8000 projects from 385 companies.*
Facts and Fallacies about Software Development

• **Facts**
  – The most important factor in software development is the quality of the programmers.
  – The best programmers are up to 28 times better than the worst.
  – Adding people to a late project makes it later.
  – One of the most common causes of runaway projects is poor estimation.
  – The other most common cause of runaway projects is unstable requirements.
  – Requirements errors are the most expensive to fix during production.
  – Maintenance typically consumes 40 to 80 percent of software costs.
  – Enhancements represent roughly 60 percent of maintenance costs.

• **Fallacies**
  
  – *Software needs more methodologies.*
  
  – *You teach people how to program by showing them how to write programs.*

Why is software hard?