Systems Development

MBA 8125 – Week 8
How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it

How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed
Course Overview

Diagram showing the flow of corporate strategy, IS strategy and plan, internal systems development, implementation, security, internal systems operations, and outsourced systems operations and infrastructure.
Learning Objectives

- Describe the systems development life cycle
- Apply project management principles to information systems projects
- Understand the user’s role in the development and implementation of information systems
- Identify generic system conversion strategies
- Examine challenges to structured stable-state thinking
- Understand that security starts with development efforts.
Challenges & Questions

- Can building new systems produce organizational change? Should it?
- How can a company can information systems that fit its business plan?
- What are the core activities in the systems development process?
- What are the alternative methods for building information systems?
Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
What is Software Engineering

“The study of methods for producing high quality software at minimum cost.”

- **Stakeholders:** Users, managers, designers, programmers, competitors.

- **Issues:** Efficiency, quality, delivery target, changing requirements, innovation, team interaction, team incentives, project organization and management, software structure, reusability, prototyping, formal description techniques, development tools.
Why Develop an Information System?

- Phenomena that trigger IS development
  - An opportunity (proactive)
  - A problem (reactive)
  - A directive
Ultimate Management Challenge

“... It’s hard work to make improvements without changing anything...”

Mikahail Gorbachev, from his address to the Supreme Soviet Congress, February, 1986
System Interdependence

Figure 1.2
(Laudon & Laudon 2006)
Definition of Success

“A successful software project is one whose deliverables satisfy and possibly exceed the stakeholders’ expectations, that is developed in a timely and economical fashion, and is resilient to change and adaptation.”

(adapted from Grady Booch and others)

- On time
- Within budget
- Meet expectations
- Adaptable (Why is this important?)
Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
- Managing Software Projects
- Security
Systems Development Life Cycle

STAGES

- Planning/definition
- Study/analysis
- Design
- Programming
- Installation
- Maintenance

END PRODUCTS

- Project proposal report
- System proposal report
- Design specifications
- Program code
- Testing and installation
- Postimplementation audit

OPERATIONS

Milestone 1: Project initiation
Milestone 2: Design solution decision
Milestone 3: Design specification sign-off
Milestone 4: Production decision

Year 1
Year 2
3-8 year lifespan

Copyright V.C. Storey, M.M. Moore, C. Stucke, D. Truex, 2007
Spiral (Iterative) Model
Iterations Retire Risks

Initial Project Risks
Initial Project Scope

Define scenarios to address highest risks

Plan Iteration N
• Cost
• Schedule

Develop Iteration N
• Collect cost and quality metrics

Assess Iteration N

Revise Overall Project Plan
• Cost
• Schedule
• Scope/Content

Revise Project Risks
• Reprioritize

Risks Eliminated
Importance of Testing

- Unit Test
- Integration Test
- Full System Test
- Stress Testing
- Field Testing
- Regression Testing

“Make sure the system works!”
Installation: Conversion Strategies

Strategies used to convert from one IS to another
Maintenance and Support

Activities in Systems Support
Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
- Managing Software Projects
- Security
Alternative Methodologies

- Other options for software development:
  - Rapid Prototyping
  - Iterative Development
  - Object-Oriented Development
  - “Off the Shelf” Applications
  - End User Development
  - Outsourcing
Alternative Methodologies: Rapid Prototyping

Build experimental system to demonstrate, evaluate approach; users refine needs

- **Prototype**: Preliminary working version of information system for demonstration, evaluation purposes

- **Problems:**
  - Omission of basic requirements.
  - Lack of documentation, testing.
  - Prototyping tools may not be capable of developing complex systems.
  - Users see prototype and expect to be able to use it.
## Prototyping Guidelines

<table>
<thead>
<tr>
<th>When to prototype</th>
<th>When not to prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small-scale systems</td>
<td>Large-scale systems</td>
</tr>
<tr>
<td>Systems solving unstructured problems</td>
<td>Complex systems</td>
</tr>
<tr>
<td>When it’s difficult for users to specify system requirements</td>
<td>Systems with interfaces to other systems</td>
</tr>
</tbody>
</table>
In iterative development, refinement of the system continues until users are satisfied.
Object-Oriented Development

- **Object**
  - basic unit of systems analysis and design
  - important entities ("things")
  - (e.g. customer, account, product, sale, invoice, employee, etc.)
  - combine **data** and **processes** used on the data
    (e.g. sale may have a process or method called calculate_total and account may have update_balance as a method)

Source: Laudon & Laudon 2006
Class and Inheritance

See [http://www.agilemodeling.com/style/classDiagram.htm](http://www.agilemodeling.com/style/classDiagram.htm)
Unified Modeling Language (UML)

- Industry standard for representing various views of an object-oriented system using a series of graphical diagrams

- Diagrams

  1. **Structural diagrams**: relationship between classes
  2. **Behavioral diagrams**: interactions in object-oriented system

A UML Use Case Diagram

Source: Laudon & Laudon 2006

See http://www.agilemodeling.com/style/useCaseDiagram.htm
Alternative Methodologies: End-User Development

- End-users develop system with little help from technical specialists
  - Small, desktop applications
  - Users have autonomy over system

- Management Benefits
  - Improved requirements
  - Development controlled by users
  - Reduced application backlog

- Management Problems
  - Insufficient review / analysis
  - Lack of standards and controls
  - Proliferation of “private” information systems and data
Alternative Methodology: Acquiring Software Packages

- Commercial Off the Shelf (COTS) Packages
  - Set of prewritten application software programs commercially available
  - Modification of software package to meet organization’s needs

Figure 14.12 – Laudon & Laudon 2006
Alternative Methodology: Outsourcing

Purchase of an externally produced good or service that was previously produced internally

Advantages
- Economy
- Predictability
- Frees up human resources

Disadvantages
- Loss of control
- Vulnerability of strategic information
- Dependency

Is Outsourcing Fool's Gold? (Kroenke, 2006)
### Comparison of Systems Development Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Features</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Systems Lifecycle (Waterfall) | Sequential step-by-step process  
Written specification and approvals  
Limited role of users | Necessary for large complex systems and projects | Slow and expensive  
Discourages changes  
Massive paperwork to manage |
| Prototyping               | Requirements specified dynamically with experimental system  
Rapid, informal, and iterative process  
User interacts with prototype | Rapid and inexpensive  
Useful when requirements are uncertain or when end-user interface is important  
Promotes user participation | Inappropriate for large, complex systems  
Can gloss over steps in analysis, documentation, and testing |
| End-user Development      | Systems created by end users using fourth-generation software tools  
Rapid and informal  
Minimal role of IT dept | Users control systems-building  
Saves development time and cost  
Reduces application backlog | Can lead to proliferation of uncontrolled information systems  
Systems do not always meet quality assurance standards |
# Comparison of Systems Development Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Features</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Software Package</td>
<td>Commercial software eliminates need for internally developed software programs</td>
<td>Design, programming, installation, and maintenance work reduced &lt;br&gt;Can save time and cost when developing common business applications &lt;br&gt;Reduces need for internal information systems resources</td>
<td>May not meet organization’s unique requirements &lt;br&gt;May not perform many business function well &lt;br&gt;Extensive customization raises development costs</td>
</tr>
<tr>
<td>Outsourcing</td>
<td>Systems built and sometimes operated by external vendors</td>
<td>Can reduce or control costs &lt;br&gt;Can produce systems when internal resources not available or technically deficient</td>
<td>Loss of control over the information systems function &lt;br&gt;Dependence on the technical direction and prosperity of external vendors</td>
</tr>
</tbody>
</table>
Evolving Challenges and Solutions

Management Challenges

- Changing demands in application development in digital firm era.
- Agility and scalability critical goals and success factors. *Strategic agility (innovation) organization must design itself to be appropriately agile in response to external and internal forces* (Gardner, 2004)
- Interorganizational system requirements when networks of applications are managed by different business partners

Source: Laudon & Laudon 2006
Evolving Challenges and Solutions

Evolving Solutions

Component-Based Development:

• Building systems by assembling and integrating existing software components

Web Services and Service-Oriented Computing:

• Web services – tools to build new applications or enhancing existing systems. See [http://en.wikipedia.org/wiki/Web_service](http://en.wikipedia.org/wiki/Web_service)

• Web services -- software components deliverable over Internet; provide functions for organization’s existing systems or create new systems that link organization’s systems to those of other organizations. See [http://en.wikipedia.org/wiki/Service-oriented_architecture](http://en.wikipedia.org/wiki/Service-oriented_architecture)

Source: Laudon & Laudon 2006
Growing Systems in Emergent Organizations

As the new economic realities pressure your organization to change from stable to emergent, new practices for IT support are required.

Duane Truex,
Richard Baskerville, & Heinz Klein
Revoking Traditional ISD Goals

- Lengthy analysis & design are poor investments
- User satisfaction is improbable
- Abstract requirements are largely imaginary
- Complete and unambiguous specifications are ineffectual
- New systems development projects denote failure
Emergent Goals

- Always analysis
- Dynamic requirement negotiations
- Incomplete and usefully ambiguous specifications
- Continuous redevelopment
- Adaptability orientation
Figure. Alternative lifespan economies.
(from Truex et al, 1999, CACM)

Traditional view

Emergent & Deferred view
Supported by…

- Back channel communications
- Emergent IT organizations
- Proper awards systems
Emergent ISD: Goals and Techniques
(from Truex et al, CACM, 1999)
Conclusions

- Software engineering is a challenging process.
- Non-IS managers have an important role in software development.
- Multiple methodologies for developing software.
  - Information systems differ.
  - No development process works in all situations.
- Difficult to manage software projects.