Systems Development

MBA 8120 – Week 5

Course Overview

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
- Understand that security starts with development efforts.
Challenges

- Can building new systems produce organizational change?
- 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
- Managing Software Projects
- Security

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

<table>
<thead>
<tr>
<th>STAGES</th>
<th>END PRODUCTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning/definition</td>
<td>Project proposal report</td>
</tr>
<tr>
<td>Study/analysis</td>
<td>System proposal report</td>
</tr>
<tr>
<td>Design</td>
<td>Design specifications</td>
</tr>
<tr>
<td>Programming</td>
<td>Program code</td>
</tr>
<tr>
<td>Installation</td>
<td>Testing and installation</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Postimplementation audit</td>
</tr>
<tr>
<td>Milestone 1 Project initiation</td>
<td>Year 1</td>
</tr>
<tr>
<td>Milestone 2 Design solution decision</td>
<td>Year 2</td>
</tr>
<tr>
<td>Milestone 3 Design specifications sign-off</td>
<td>2-6 year lifespan</td>
</tr>
<tr>
<td>Milestone 4 Production decision</td>
<td></td>
</tr>
</tbody>
</table>
Spiral (Iterative) Model

Initial Project Risks
- Initial Project Scope
- Revise Overall Project Plan
- Cost
- Schedule
- Scope/Content

Plan Iteration N
- Cost
- Schedule

Develop Iteration N
- Collect cost and quality metrics

Assess Iteration N

Iterations Retire Risks

Make sure the system works!

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

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

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.

Alternative Methodologies:
Iterative Development

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)

Class and Inheritance
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

Source: Laudon & Laudon 2006

See [http://www.agilemodeling.com/essays/UMLDiagrams.htm](http://www.agilemodeling.com/essays/UMLDiagrams.htm)

---

**A UML Use Case Diagram**

Source: Laudon & Laudon 2006

See [http://www.agilemodeling.com/style/useCaseDiagram.htm](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

Source: Laudon & Laudon 2006

See [http://www.agilemodeling.com/style/useCaseDiagram.htm](http://www.agilemodeling.com/style/useCaseDiagram.htm)
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>
<tbody>
<tr>
<td>Systems Lifecycle</td>
<td>Sequential step-by-step process</td>
<td>Necessary for large complex systems and projects</td>
<td>More and expansive changes</td>
</tr>
<tr>
<td>(Waterfall)</td>
<td>Written specification and approvals</td>
<td></td>
<td>Masses paperwork to manage</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Requirements specified dynamically with experimental system</td>
<td>Rapid and unambiguous</td>
<td>Inappropriate for large, complex systems</td>
</tr>
<tr>
<td></td>
<td>Rapid, informal, and iterative process</td>
<td>Useful when requirements are uncertain or when user interface is important</td>
<td>Can gloss over steps in analysis, documentation, and testing</td>
</tr>
<tr>
<td></td>
<td>User interacts with prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End-user Development</td>
<td>Systems created by and for end users using fourth-generation software tools</td>
<td>Users control system-assisted building</td>
<td>Can lead to proliferation of uncontrolled information</td>
</tr>
<tr>
<td></td>
<td>Rapid and informal</td>
<td>Users development time and cost</td>
<td>Systems do not always meet quality assurance standards</td>
</tr>
<tr>
<td></td>
<td>Minimized role of IT dept</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright V.C. Storey, M.M. Moore, C. Stucke 2006
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 clients need for internally developed software programs</td>
<td>Design, programming, installation, and maintenance work involved. Can save time and cost when developing custom business applications. Maintain need for internal information systems innovation.</td>
<td>May not meet organization’s unique requirements. May not perform many business functions well. Extensive customization raises development costs.</td>
</tr>
<tr>
<td>Outsourcing</td>
<td>Systems built and sometimes operated by external vendors</td>
<td>Cost reduction or control costs. Can produce systems where internal resources not available or technically deficient.</td>
<td>Loss of control over the information systems function. Dependence on the technical direction and propriety 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

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
- 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
Architectural Evolution

Standalone systems
Adaptive Patchwork
Data Warehousing
Technical Connectivity
Enterprise Integration
Component Architecture

Technological integration
Data integration
Process integration

Source: Dr. Stefano Grazioli
University of Virginia
Used with permission

Agenda

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

Managing Software Projects

Cost of Errors
Schedule Estimation and Planning
Project Risk
Project Evaluation
Classic Mistakes
Management Challenges

- Systems development projects
  - Often backlog, late, excessive costs, missing capabilities

- Information systems must be conceived, designed, implemented, and maintained.
- Difficult to define requirements, costs, and benefits

Project Risk

- Software not a product; it is an embodiment of knowledge
  - Customer Knowledge
  - Technical Knowledge
- Difficult to extract this knowledge

Source: Tiwana and Keil 2004
One Minute Risk Assessment Tool

Source: Tiwana and Keil 2004

Project Evaluation

Project Risk

<table>
<thead>
<tr>
<th>High</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cautiously examine</td>
<td>Identify and develop</td>
</tr>
<tr>
<td>Avoid</td>
<td>Routine projects</td>
</tr>
</tbody>
</table>

Potential Benefits to Firm

High

Low

Classic Mistakes of Project Management

- Personnel Mistakes
- Customer Mistakes
- Process Mistakes
- Product Mistakes
- Technology Mistakes
Personnel Mistakes
- Undermined Motivation
- Weak players / problem employees / heroics
- Adding personnel to already late project
  [Brook’s law: adding more people to a late project makes the project later.]
- Noisy, crowded work surroundings

Sponsor/Customer Mistakes
- Unrealistic expectations
- No stakeholder buy-in
- Insufficient user input
- Wishful thinking

Process Mistakes
- Overly optimistic schedules
- Failure to manage unique risks
- Contractor failure
- Insufficient planning
- Abandonment of plans
- Shortchanged process / quality assurance
Product Mistakes
- Feature creep
- Developer gold-plating
- Schedule slip / added tasks
- Advancing state of art

Technology Mistakes
- Silver bullet tools
- Overestimated savings from tools
- Switching tools in middle

Why Do We Make the Same Mistakes?
- Short-Term Fix
- “No time to do it right”
- External pressure / panic / ignorance
Security
- Process of creating a secure system
  - Decide how users will be authenticated
  - Determine user groups
  - List primary features and functions of the system
  - Determine how restrictions will be enforced
  - Allocate permissions to user groups for specific features and functions
  - Be sure that applications accept input that is no longer than what is expected (This avoids what is called a buffer overflow that could be used to seize control of a system. See http://en.wikipedia.org/wiki/Buffer_overflow.)

Conclusions
- Software engineering is challenging process
- Non-IS managers have 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