Learning Objectives

- List the seven steps in the systems development life cycle (SDLC) and associated activities for each step.
- Describe prototyping and profile an example of a prototype.
- Describe the advantages of prototyping.
- Understand the user’s role in the development and implementation of information systems.
- Describe business process outsourcing (BPO).
- Define the role of outsourcing.

Challenges

- Can building new systems produce organizational change?
- How can a company build 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
- Insourcing/Outsourcing
Why Develop an Information System?

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

However, the 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

Evaluation criteria
- **Strategic alignment**: extent to which project is viewed as helping the organization achieve objectives and long-term goals
- **Potential benefits**: extent to which project viewed as improving profits, customer service etc. and duration of these benefits
- **Potential costs / resource and availability**: number and type of resources required and their availability
- **Project size /duration**: number of individuals and time needed to complete project
- **Technical difficulty /risks**: level of technical difficulty involved in successfully completing project within given time and resource constraints.

Source: Hoffer, George, Valacich, 2005
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?)

System Development Life Cycle

<table>
<thead>
<tr>
<th>SDLC PHASE</th>
<th>ACTIVITIES</th>
</tr>
</thead>
</table>
| 1. Planning | Define the system to be developed  
| | Set the project scope  
| | Develop the project plan |
| 2. Analysis | Gather business requirements |
| 3. Design | Design the technical architecture  
| | Design system models |
| 4. Development | Build technical architecture  
| | Build databases and programs |
| 5. Testing | Write test conditions  
| | Perform testing |
| 6. Implementation | Write user documentation  
| | Provide training |
| 7. Maintenance | Build a help desk  
| | Support system changes |

Systems Development Life Cycle

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

Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
- Insourcing/Outsourcing
The “WaterFall” Model (Barry Böehm)

- Waterfall methodology - a sequential, activity-based process in which each phase in the SDLC is performed sequentially from planning through implementation

Phase 1: Planning

3. Develop the project plan including tasks, resources, and timeframes
   - Project plan - defines the what, when, and who questions of system development
   - Project manager - an individual who is an expert in project planning and management, defines and develops the project plan and tracks the plan to ensure all key project milestones are completed on time
   - Project milestones - represent key dates for which you need a certain group of activities performed

Phase 2: Analysis

- Analysis phase - involves end users and IT specialists working together to gather, understand, and document the business requirements for the proposed system

- Business requirements - the detailed set of knowledge worker requests that the system must meet in order to be successful

- Joint application development (JAD) - knowledge workers and IT specialists meet, sometimes for several days, to define or review the business requirements for the system
Phase 2: Analysis

2. Prioritize the requirements
   - Requirements definition document – prioritizes the business requirements and places them in a formal comprehensive document

Phase 3: Design

- **Design phase** - build a technical blueprint of how the proposed system will work
- Two primary design activities:
  1. Design the technical architecture
     - Technical architecture - defines the hardware, software, and telecommunications equipment required to run the system

Phase 3: Design

2. Design system models
   - **Modeling** - the activity of drawing a graphical representation of a design
   - **Graphical user interface (GUI)** - the interface to an information system
   - **GUI screen design** - the ability to model the information system screens for an entire system

Phase 4: Development

- **Development phase** - take all of your detailed design documents from the design phase and transform them into an actual system
- Two primary development activities:
  1. Build the technical architecture
  2. Build the database and programs
     - Both of these activities are mostly performed by IT specialists
Phase 5: Testing

- **Testing phase** - verifies that the system works and meets all of the business requirements defined in the analysis phase
- Two types of testing
  1. Black Box – Only concerned with inputs and outputs
  2. White Box – Concerned with the processes

“Make sure the system works!”

Phase 6: Implementation

- **Implementation phase** - distribute the system to all of the knowledge workers and they begin using the system to perform their everyday jobs
- Two primary implementation activities
  1. Write detailed user documentation
     - **User documentation** - highlights how to use the system
  2. Provide training for the system users
     - **Online training** - runs over the Internet or off a CD-ROM
     - **Workshop training** - is held in a classroom environment and lead by an instructor

Phase 5: Testing

- Perform the testing of the system
  - **Unit testing** – tests individual units of code
  - **System testing** – verifies that the units of code function correctly when integrated
  - **Integration testing** – verifies that separate systems work together
  - **User acceptance testing (UAT)** – determines if the system satisfies the business requirements

Phase 6: Implementation

- Choose the right implementation method
  - **Parallel implementation** – use both the old and new system simultaneously
  - **Plunge implementation** – discard the old system completely and use the new
  - **Pilot implementation** – start with small groups of people on the new system and gradually add more users
  - **Phased implementation** – implement the new system in phases
Phase 7: Maintenance

- **Maintenance phase** - monitor and support the new system to ensure it continues to meet the business goals
- Three types of maintenance
  1. Corrective – Fix known problems (“bugs”)
  2. Adaptive – Upgrade functionality – no new features
  3. Enhancement - Add new functionality – new capabilities

Relative Cost of Repairing Errors

Source: Haag et al. *Management Information Systems for the Information Age*

Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
- Insourcing/ Outsourcing

Alternative Methodologies

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

Rapid Prototyping

- **Prototype**: Preliminary working version of information system for demonstration, evaluation purposes; user can refine needs specification
- **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

Prototype: Preliminary working version of information system for demonstration, evaluation purposes; user can refine needs specification

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 system requirements</td>
</tr>
</tbody>
</table>

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

- **Class Name**
- **Attributes (data)**
  - id
  - name
  - address
  - dateFirstPosition
  - pay
- **Methods (processes)**

Subclass Name
- More attributes
  - annualSalary
  - bonus
  - cafelalous
  - cafecoveredtime
  - cafecoveredposition
  - cafecoveredpay
- More methods

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

Class Diagram

A UML Use Case Diagram

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

† End-users develop system with little help from technical specialists

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

Agenda

- Software Engineering
- Systems Development Life Cycle
- Alternative Methodologies
- Insourcing/Outsourcing

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

OUTSOURCING

- Two primary choices to build IT systems (of great size and complexity):
  1. Insourcing - involves choosing IT specialists within your organization to develop the system
  2. Outsourcing - the delegation of specific work to a third party for a specified length of time, at a specified cost, and at a specified level of service
The main reasons behind the rapid growth of the outsourcing industry include the following:

- Globalization
- The Internet
- Growing economy and low unemployment rate
- Technology
- Deregulation

Outsourcing Options

IT outsourcing for software development can take one of four forms:

1. Purchase existing software
2. Purchase existing software and paying the publisher to make certain modifications
3. Purchase existing software and paying the publisher for the right to make modifications yourself
4. Outsource the development of an entirely new and unique system for which no software exists
Outsourcing Options

- There are three different forms of outsourcing:
  1. **Onshore outsourcing** - the process of engaging another company within the same country for services.
  2. **Nearshore outsourcing** - contracting an outsourcing arrangement with a company in a nearby country.
  3. **Offshore outsourcing** - contracting with a company that is geographically far away.

Offshore Outsourcing

- Primary outsourcing countries are:
  - India
  - China
  - Eastern Europe (including Russia)
  - Ireland
  - Israel
  - Philippines

The Advantages and Disadvantages of Outsourcing

- **Advantages:**
  - Focus on unique core competencies
  - Exploit the intellect of another organization
  - Better predict future costs
  - Acquire leading-edge technology
  - Reduce costs (See next slide)
  - Improve performance accountability

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>SALARY RANGE PER YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>China</td>
<td>$5,000 – 9,000</td>
</tr>
<tr>
<td>India</td>
<td>$6,000 – 10,000</td>
</tr>
<tr>
<td>Philippines</td>
<td>$6,500 – 11,000</td>
</tr>
<tr>
<td>Russia</td>
<td>$7,000 – 13,000</td>
</tr>
<tr>
<td>Ireland</td>
<td>$21,000 – 28,000</td>
</tr>
<tr>
<td>Canada</td>
<td>$25,000 – 50,000</td>
</tr>
<tr>
<td>United States</td>
<td>$60,000 – 90,000</td>
</tr>
</tbody>
</table>
The Advantages and Disadvantages of Outsourcing

- **Disadvantages:**
  - Reduces technical know-how for future innovation
  - Reduces degree of control
  - Increases vulnerability of your strategic information
  - Increases dependency on other organizations

Business Process Outsourcing (BPO)

- **Business process outsourcing (BPO)** is using a contractual service to completely manage, deliver and operate one or more (typically IT or call center-intensive) business processes or functions

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

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>Slow and expensive</td>
</tr>
<tr>
<td></td>
<td>Written specification and approvals</td>
<td></td>
<td>Discourages changes</td>
</tr>
<tr>
<td></td>
<td>Limited role of users</td>
<td></td>
<td>Massive paperwork to manage</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Requirements specified dynamically with</td>
<td>Rapid and inexpensive</td>
<td>Inappropriate for large, complex systems</td>
</tr>
<tr>
<td></td>
<td>experimental system</td>
<td>Useless when requirements are uncertain or when</td>
<td>Can gloss over steps in analysis, documentation,</td>
</tr>
<tr>
<td></td>
<td>Rapid, informal, and iterative process</td>
<td>and end-user interface is important</td>
<td>and testing</td>
</tr>
<tr>
<td></td>
<td>User interacts with prototype</td>
<td>Promotes user participation</td>
<td></td>
</tr>
<tr>
<td>End-user Development</td>
<td>Systems created by end-users using fourth-</td>
<td>Users control systems-building</td>
<td>Can lead to proliferation of uncontrolled</td>
</tr>
<tr>
<td></td>
<td>generation software tools</td>
<td>Saves development time and cost</td>
<td>information systems</td>
</tr>
<tr>
<td></td>
<td>Rapid and informal</td>
<td></td>
<td>Systems do not always meet quality assurance</td>
</tr>
<tr>
<td></td>
<td>Minimal role of IT dept</td>
<td></td>
<td>standards</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approach</td>
<td>Features</td>
<td>Advantages</td>
<td>Disadvantages</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Application</td>
<td>Commercial software eliminates need for internally developed software programs</td>
<td>Design, programming, installation, and maintenance work reduced Can save time and cost when developing common business applications Reduces need for internal information systems resources</td>
<td>May not meet organization’s unique requirements May not perform many business function well 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 Can produce systems when internal resources not available or technically deficient</td>
<td>Loss of control over the information systems function Dependence on the technical direction and prosperity of external vendors</td>
</tr>
</tbody>
</table>