Chapter 1
Systems Development in an Organization Context
Learning Objectives

✓ Define information systems analysis and design.
✓ Describe the information Systems Development Life Cycle (SDLC).
✓ Explain Rapid Application Development (RAD), prototyping, Computer Aided Software Engineering (CASE), and Service-Oriented Architecture (SOA).
✓ Describe agile methodologies and eXtreme programming.
Introduction

Information Systems Analysis and Design

- Complex organizational process
- Used to develop and maintain computer-based information systems
- Used by a team of business and systems professionals
Introduction (Cont.)

- Information Systems Analysis and Design (cont.)
  - based on
    1. The understanding of the organization’s objectives, structure, and processes.
    2. Knowledge of how to exploit IT for advantages.
  - One of the important results is Application Software
  - Our goal in this course is to emphasize application software development, the primary responsibility of a system analyst.
Introduction (Cont.)

- **Application Software**
  - Computer software designed to support organizational functions or processes

- **Systems Analyst**
  - Organizational role most responsible for analysis and design of information systems
    - Study the problems and needs of organization.
    - Determine how people, methods and IT combined to improve the organization.
FIGURE 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools
A Modern Approach to Systems Analysis and Design

- 1950s: focus on efficient automation of existing processes
- 1960s: advent of 3GL, faster and more reliable computers
- 1970s: system development becomes more like an engineering discipline
A Modern Approach to Systems Analysis and Design (Cont.)

- 1980s: major breakthrough with 4GL, CASE tools, object oriented methods
- 1990s: focus on system integration, GUI applications, client/server platforms, Internet
- The new century: Web application development, wireless PDAs, component-based applications
Developing Information Systems

- **System Development Methodology** is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.
Systems Development Life Cycle (SDLC)

Traditional methodology used to develop, maintain, and replace information systems.

Phases in SDLC:
- Planning
- Analysis
- Design
- Implementation
- Maintenance
SDLC Characteristics

- Circular Processes
  - In any phase, the project can return to an earlier phase if necessary
  - It can complete some activities in one phase in parallel with some activities of another phase
  - Sometimes it is iterative; phases are repeated as required until an acceptable system is found
  - Sometimes it considered to be a spiral, cycle through the phase at different levels of detail
- Software is the most end product; other outputs include documentation and training.
Standard and Evolutionary Views of SDLC

FIGURE 1-2
The systems development life cycle

FIGURE 1-3 Evolutionary model
Systems Development Life Cycle (SDLC) (Cont.)

- **Planning** – an organization’s total information system needs are identified, analyzed, prioritized, and arranged
- **Analysis** – system requirements are studied and structured
- **Design** – a description of the recommended solution is converted into logical and then physical system specifications
Systems Development Life Cycle (SDLC) (Cont.)

- **Logical design** – all functional features of the system chosen for development in analysis are described independently of any computer platform.

- **Physical design** – the logical specifications of the system from logical design are transformed into the technology-specific details from which all programming and system construction can be accomplished.
Systems Development Life Cycle (SDLC) (Cont.)

- **Implementation** – the information system is coded, tested, installed and supported in the organization

- **Maintenance** – an information system is systematically repaired and improved
<table>
<thead>
<tr>
<th>Phase</th>
<th>Products, Outputs, or Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Priorities for systems and projects; an architecture for data, networks, and selection hardware, and IS management are the result of associated systems. Detailed steps, or work plan, for project. Specification of system scope and planning and high-level system requirements or features. Assignment of team members and other resources. System justification or business case.</td>
</tr>
<tr>
<td>Analysis</td>
<td>Description of current system and where problems or opportunities are with a general recommendation on how to fix, enhance, or replace current system. Explanation of alternative systems and justification for chosen alternative.</td>
</tr>
<tr>
<td>Design</td>
<td>Functional, detailed specifications of all system elements (data, processes, inputs, and outputs). Technical, detailed specifications of all system elements (programs, files, network, system software, etc.). Acquisition plan for new technology.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Code, documentation, training procedures, and support capabilities.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>New versions or releases of software with associated updates to documentation, training, and support.</td>
</tr>
</tbody>
</table>
FIGURE 1-7
The analysis–design–code–test loop

FIGURE 1-8
The heart of systems development

Current practice combines analysis, design, and implementation into a single iterative and parallel process of activities.
Traditional Waterfall SDLC

One phase begins when another completes, with little backtracking and looping.

FIGURE 1-9
A traditional waterfall SDLC
Problems with Waterfall Approach

- System requirements “locked in” after being determined (can't change)
- Limited user involvement (only in requirements phase)
- Too much focus on milestone deadlines of SDLC phases to the detriment of sound development practices
Different Approaches to Improving Development

- Computer-aided software engineering (CASE) Tools
- Rapid Application Development (RAD)
- Service-Oriented Architecture (SOA)
- Agile Methodologies
- eXtreme Programming
Computer-Aided Software Engineering (CASE) Tools

- **Diagramming tools** enable graphical representation.
- **Computer displays and report generators** help prototype how systems “look and feel”.
- **Analysis tools** automatically check for consistency in diagrams, forms, and reports.
A central repository provides integrated storage of diagrams, reports, and project management specifications.

Documentation generators standardize technical and user documentation.

Code generators enable automatic generation of programs and database code directly from design documents, diagrams, forms, and reports.
CASE Tools (Cont.)

FIGURE 1-10
A class diagram from IBM’s Rational Rose
(Source: IBM)
## CASE Tools (Cont.)

<table>
<thead>
<tr>
<th>SDLC Phase</th>
<th>Key Activities</th>
<th>CASE Tool Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project identification and selection</td>
<td>Display and structure high-level organizational information</td>
<td>Diagramming and matrix tools to create and structure information</td>
</tr>
<tr>
<td>Project initiation and planning</td>
<td>Develop project scope and feasibility</td>
<td>Repository and documentation generators to develop project plans</td>
</tr>
<tr>
<td>Analysis</td>
<td>Determine and structure system requirements</td>
<td>Diagramming to create process, logic, and data models</td>
</tr>
<tr>
<td>Logical and physical design</td>
<td>Create new system designs</td>
<td>Form and report generators to prototype designs; analysis and documentation generators to define specifications</td>
</tr>
<tr>
<td>Implementation</td>
<td>Translate designs into an information system</td>
<td>Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Evolve information system</td>
<td>All tools are used (repeat life cycle)</td>
</tr>
</tbody>
</table>

**Table 1-2: Examples of CASE Usage within the SDLC**
Rapid Application Development (RAD)

- Methodology to radically decrease design and implementation time
- Involves: extensive user involvement, prototyping, integrated CASE tools, and code generators
Rapid Application Development (RAD) (Cont.)

FIGURE 1-11
RAD life cycle
Service-Oriented Architecture (SOA)

- An approach to systems development based on building complete systems through **assembling software components**, each of which model generic business functions
Service-Oriented Architecture (SOA) (Cont.)

FIGURE 1-12
Illustration of a service, a credit check, used by applications and other services
Agile Methodologies

- Motivated by recognition of software development as fluid, unpredictable, and dynamic

- Three key principles
  - Adaptive rather than predictive
  - Emphasize people rather than roles
  - Self-adaptive processes
The Agile Methodologies group argues that software development methodologies adapted from engineering generally do not fit with real-world software development.
When to use Agile Methodologies

- If your project involves:
  - Unpredictable or dynamic requirements
  - Responsible and motivated developers
  - Customers who understand the process and will get involved
<table>
<thead>
<tr>
<th>Factor</th>
<th>Agile Methods</th>
<th>Traditional Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Well matched to small products and teams. Reliance on tacit knowledge limits scalability.</td>
<td>Methods evolved to handle large products and teams. Hard to tailor down to small projects.</td>
</tr>
<tr>
<td>Criticality</td>
<td>Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.</td>
<td>Methods evolved to handle highly critical products. Hard to tailor down to products that are not critical.</td>
</tr>
<tr>
<td>Dynamism</td>
<td>Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.</td>
<td>Detailed plans and Big Design Up Front excellent for highly stable environment but a source of expensive rework for highly dynamic environments.</td>
</tr>
<tr>
<td>Personnel</td>
<td>Requires continuous presence of a critical mass of scarce experts. Risky to use no-agile people.</td>
<td>Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic.</td>
</tr>
<tr>
<td>Culture</td>
<td>Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos).</td>
<td>Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order).</td>
</tr>
</tbody>
</table>

(Source: Boehm and Turner, 2004. Used by permission.)
eXtreme Programming

- Short, incremental development cycles
- Automated tests
- Two-person programming teams
eXtreme Programming (Cont.)

- Coding and testing operate together

Advantages:
- Communication between developers
- High level of productivity
- High-quality code
Object-Oriented Analysis and Design (OOAD)

- Based on objects rather than data or processes
- **Object**: a structure encapsulating attributes and behaviors of a real-world entity
Object-Oriented Analysis and Design (OOAD) (Cont.)

- **Object class**: a logical grouping of objects sharing the same attributes and behaviors
- **Inheritance**: hierarchical arrangement of classes enable subclasses to inherit properties of superclasses
Rational Unified Process (RUP)

- An object-oriented systems development methodology
- RUP establishes four phase of development: inception, elaboration, construction, and transition.
- Each phase is organized into a number of separate iterations.
FIGURE 1-13
Phases of OOSAD-based development
Our Approach to Systems Development

- The SDLC is an organizing and guiding principle in this book.
- We may construct artificial boundaries or artificially separate activities and processes for learning purposes.
- Our intent is to help you understand all the pieces and how to assemble them.
Summary

In this chapter you learned how to:

- Define information systems analysis and design.
- Describe the information Systems Development Life Cycle (SDLC).
- Explain Rapid Application Development (RAD), prototyping, Computer Aided Software Engineering (CASE), and Service-Oriented Architecture (SOA).
- Describe agile methodologies and eXtreme programming.