Modern Systems Analysis and Design

Prof. David Gadish

Chapter 1
The Systems Development Environment
Learning Objectives

✓ Define information systems analysis and design.
✓ Describe the different types of information systems.
✓ Describe the information Systems Development Life Cycle (SDLC).
✓ Explain Rapid Application Development (RAD), prototyping, Joint Application Development (JAD), and Computer Aided Software Engineering (CASE).
✓ 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.)

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
A Modern Approach to Systems Analysis and Design (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.
A Modern Approach to Systems Analysis and Design (Cont.)

Figure 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools.
Types of Information Systems and Systems Development

- **Transaction Processing Systems (TPS)**
  - Automate handling of data about business activities (transactions)
  - Process orientation

- **Management Information Systems (MIS)**
  - Converts raw data from transaction processing system into meaningful form
  - Data orientation
Types of Information Systems and Systems Development (Cont.)

- Decision Support Systems (DSS)
  - Designed to help decision makers
  - Provides interactive environment for decision making
  - Involves data warehouses, executive information systems (EIS)
  - Database, model base, user dialogue
Table 1-1 Systems Development for Different IS Types

<table>
<thead>
<tr>
<th>IS Type</th>
<th>IS Characteristics</th>
<th>Systems Development Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction processing system</td>
<td>High-volume, data capture focus; goal is efficiency of data movement and processing and interfacing different TPSs</td>
<td>Process orientation; concern with capturing, validating, and storing data and with moving data between each required step</td>
</tr>
<tr>
<td>Management information system</td>
<td>Draws on diverse yet predictable data resources to aggregate and summarize data; may involve forecasting future data from historical trends and business knowledge</td>
<td>Data orientation; concern with understanding relationships among data so data can be accessed and summarized in a variety of ways; builds a model of data that supports a variety of uses</td>
</tr>
<tr>
<td>Decision support system</td>
<td>Provides guidance in identifying problems, finding and evaluating alternative solutions, and selecting or comparing alternatives; potentially involves groups of decision makers; often involves semi-structured problems and the need to access data at different levels of detail</td>
<td>Data and decision logic orientations; design of user dialogue; group communication may also be key, and access to unpredictable data may be necessary; nature of systems requires iterative development and almost constant updating</td>
</tr>
</tbody>
</table>
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
Standard and Evolutionary Views of SDLC

Figure 1-3  The systems development life cycle

Figure 1-4  Evolutionary model SDLC
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.
Systems Development Life Cycle (SDLC) (Cont.)

- **Design** – a description of the recommended solution is converted into logical and then physical system specifications.

- **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>
The Heart of the Systems Development Process

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

Figure 1-10 A traditional waterfall SDLC

One phase begins when another completes, little backtracking and looping
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

- Prototyping
- Computer-Aided Software Engineering (CASE) Tools
- Joint Application Design (JAD)
Different Approaches to Improving Development (Cont.)

- Rapid Application Development (RAD)
- Agile Methodologies
- eXtreme Programming
Prototyping

- Iterative development process:
- Requirements quickly converted to a working system.
- System is continually revised.
- Close collaboration between users and analysts.
Figure 1-11  The prototyping methodology

Computer-Aided Software Engineering (CASE) Tools

- Diagramming tools enable graphical representation.
- Computer displays and report generators help prototype how systems “look and feel”.
Computer-Aided Software Engineering (CASE) Tools (Cont.)

- Analysis tools automatically check for consistency in diagrams, forms, and reports.
- Central repository for integrated storage of diagrams, reports, and project management specifications.
Computer-Aided Software Engineering (CASE) Tools (Cont.)

- 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 1-12 A class diagram from IBM’s Rational Rose
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>
Joint Application Design (JAD)

- Structured process involving users, analysts, and managers.
- Several-day intensive workgroup sessions.
- Purpose: to specify or review system requirements.
Rapid Application Development (RAD)

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

Figure 1-12  RAD life cycle
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.
Agile Methodologies (Cont.)

### TABLE 1-4 The Agile Manifesto

<table>
<thead>
<tr>
<th>The Manifesto for Agile Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seventeen anarchists agree:</td>
</tr>
<tr>
<td>We are uncovering better ways of developing software by doing it and helping others do it.</td>
</tr>
<tr>
<td>Through this work we have come to value:</td>
</tr>
<tr>
<td>• Individuals and interactions over processes and tools.</td>
</tr>
<tr>
<td>• Working software over comprehensive documentation.</td>
</tr>
<tr>
<td>• Customer collaboration over contract negotiation.</td>
</tr>
<tr>
<td>• Responding to change over following a plan.</td>
</tr>
</tbody>
</table>

That is, while we value the items on the right, we value the items on the left more.

We follow the following principles:

• Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
• Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
• Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
• Business people and developers work together daily throughout the project.
• Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
• The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
• Working software is the primary measure of progress.
• Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
• Continuous attention to technical excellence and good design enhances agility.
• Simplicity—the art of maximizing the amount of work not done—is essential.
• The best architectures, requirements, and designs emerge from self-organizing teams.
• At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

—Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas (www.agileAlliance.org)

(Source: From Fowler and Highsmith, 2001. 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 phases of development: inception, elaboration, construction, and transition.
- Each phase is organized into a number of separate iterations.
Phases of OOSAD-based Development

Figure 1-14 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 different types of information systems.
- Describe the information Systems Development Life Cycle (SDLC).
Summary (Cont.)

✓ Explain Rapid Application Development (RAD), prototyping, Joint Application Development (JAD), and Computer Aided Software Engineering (CASE).

✓ Describe agile methodologies and eXtreme programming.