Learning Objectives

Identify the trade-offs when using CASE
Describe organizational forces for and against adoption of CASE tools
Describe the role of CASE tools and how they are used to support the SDLC
List and describe the typical components of a comprehensive CASE environment
Describe the general functions of upper CASE tools, lower CASE tools, cross life-cycle CASE tools and the CASE repository
Learning Objectives

Describe visual and emerging development tools and how they are being used
Introduction

Computer-aided Software Engineering (CASE)

- Automated software tool used by systems analysts to develop information systems
- Used to support or automate activities throughout the systems development life cycle (SDLC)
- Increase productivity
- Improve overall quality of systems
The Use of CASE in Organizations

- Purpose of CASE is to facilitate a single design philosophy within an organization.
- Figure 4.1 shows various facilities of CASE that support each phase of the systems development life cycle (SDLC).
The Use of CASE in Organizations

Objectives of CASE

- Improve quality of systems developed
- Increase speed of development and design
- Ease and improve testing process through automated checking
- Improve integration of development activities via common methodologies
- Improve quality and completeness of documentation
- Help standardize the development process
- Improve project management
- Simply program maintenance
- Promote reusability
- Improve software portability
CASE and System Quality

- Majority of organizations adopt CASE to improve speed and quality of systems development projects.
- Widespread deployment has been slower than expected.
CASE and System Quality

Several factors that inhibit widespread deployment

- **Cost**
  - Between $5,000 and $15,000 per year to provide CASE tools to one systems analyst

- **Return on Investment**
  - Biggest benefits of CASE come in late stages of SDLC

- **Productivity Bottlenecks**
  - Inability of some tools to share information
  - Difficulty in providing tools for all stages of SDLC
The Outlook for CASE

- Functionality is increasing
- Cost is decreasing
- Reverse Engineering Tools
  - Automated tools that read program source code as input and create graphical and textual representations of program design-level information
- Reengineering Tools
  - Automated software that reads program source code, analyzes it and automatically or interactively alters an existing system to improve quality and/or performance
The Outlook for CASE

Organizational Culture can influence CASE Adoption
- Table 4-2 lists the impact of CASE upon individuals within an organization

Driving and Resisting Forces for CASE
- Tables 4-3 and 4-4 present driving and resisting organizational forces for CASE adoption
Components of CASE

- **Upper CASE**
  - CASE tools designed to support the information planning and the project identification and selection, project initiation and planning, analysis and design phases of the systems development life cycle.

- **Lower CASE**
  - CASE tools designed to support the implementation and maintenance phases of the systems development life cycle.
Components of CASE

- Cross life-cycle CASE
  - CASE tools designed to support activities that occur across multiple phases of the systems development life cycle
- Most CASE tools utilize a repository to store all diagrams, forms, models and report definitions
Components of CASE

Types of CASE tools

- Diagramming tools
- Computer display and report generators
- Analysis tools used to check for incomplete, inconsistent or incorrect specifications
- A central repository
- Documentation generators
- Code generators
Components of CASE

- Security Features
- Version Control
- Import/Export
- Backup and Recovery
CASE versus Traditional Systems Development

- Traditional approach does not offer support for integration of specification documents.
- Often, documentation is done after coding is completed in traditional systems development.
- Traditional approach often leads to out-of-date documentation.
CASE versus Traditional Systems Development

Traditional Systems Development
- Emphasis on coding and testing
- Paper-based specifications
- Manual coding of programs
- Manual documenting
- Intensive software testing
- Maintain code and documentation

CASE-Based Systems Development
- Emphasis on analysis and design
- Rapid interactive prototyping
- Automated code generation
- Automated documentation generation
- Automated design checking
- Maintain design specifications
CASE Diagramming Tools

- Enable representation of a system and components visually
- Effective for representing process flows, data structures and program structures
- Several types of diagrams:
  - Data Flow Diagrams (DFD) (Figure 4-4)
  - Functional Hierarchy Diagrams (Figure 4-5)
  - Entity-Relationship Diagrams (Figure 4-6)
CASE Form and Report Generator Tools

CASE tools that support the creation of system forms and reports in order to prototype how systems will look and feel to users.

Two Purposes

- Create, modify and test prototypes of computer display forms and reports
- Identify which data items to display or collect for each form or report
Enable automatic checking for incomplete, inconsistent or incorrect specifications in diagrams, forms and reports.

Types of analyses vary depending on the organization’s development methodology and features of CASE environment
CASE Repository

Integrated CASE (I-CASE)

- Automated systems development environment that provides numerous tools to create diagrams, forms and reports
- Provides analysis, reporting and code generation facilities
- Seamlessly shares and integrates data across and between tools
- Repository is central place to store information to share between tools
CASE Repository

- Holds complete information needed to create, modify and evolve a software system from project initiation and planning to code generation and maintenance

- Two Primary Segments
  - Information Repository
  - Data Dictionary
CASE Repository

Information Repository
- Combines information about an organization’s business information and its application portfolio
- Provides automated tools to manage and control access to repository

Business Information
- Data stored in corporate databases

Application Portfolio
- Application programs used to manage business
CASE Repository

Data Dictionary

- Computer software tool used to manage and control access to the information repository
- Contains all data definitions for all organizational applications
- Cross referencing
  - Enables one description of a data item to be stored and accessed by all individuals
  - Single definition for a data item is established and used
CASE Repository

Data Dictionary

- Entries have a standard definition
  - Element name and alias
  - Textual description of the element
  - List of related elements
  - Element type and format
  - Range of acceptable values
  - Other information unique to the proper processing of this element
CASE Repository and the SDLC

- During project initiation and planning phase, all information related to the problem being solved is stored in the repository
  - Problem domain, project resources, history and organizational context
- During analysis and design phases, store graphical diagrams and prototype forms and reports
- Data stored in repository are used for basis to generate code and documentation
CASE Repository

Additional Advantages

- Assistance with project management tasks
- Aids in software reusability
  - The ability to design software modules in a manner so that they can be used again and again in different systems without significant modification
Enable the easy production of both technical and user documentation

Allow creation of master templates used to verify that documentation conforms to all stages of SDLC
CASE Code Generation Tools

- Enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms and reports stored in the repository.
Visual and Emerging Development Tools

Object-Oriented Development Tools

- **Object**
  - A chunk of program and data that is built to perform common functions within a system
  - Easily reused
  - Encapsulation
    - Process of grouping data and instructions together

- Development environment includes pre-defined objects and facilitates reuse of code
Visual Development Tools

- Enable developers to quickly create user interfaces
- Popular tools include:
  - Microsoft Visual Studio
  - Delphi
  - Powerbuilder
  - ColdFusion
Summary

- Use of CASE in Organizations
- Categories of CASE Tools
  - Reverse Engineering
  - Re-engineering
- Components of CASE
  - Upper CASE
    - Diagramming tools
    - Form and report generators
    - Analysis tools
Summary

Components of CASE (continued)
- Lower CASE
  - Code generators
- Cross Life-cycle CASE
  - Project management tools

Repository and Data Dictionary

Visual and Emerging Development Tools