

AGENDA

- The Systems View
- Business Processes
- Systems Development Life Cycle and Structured Techniques
- Information Systems Controls

THE SYSTEMS VIEW

What is a system?

- **System:** A set of interrelated components that must work together to achieve some common purpose

Information System: The collection of IT, procedures, and people responsible for the capture, movement, management, and distribution of data and information

THE SYSTEMS VIEW

- The term “System” is used to refer to something broader than an information system:
- **Systems thinking** is:
 - A discipline for seeing wholes
 - A framework for seeing interrelationships rather than things
 - An antidote to feeling of helplessness when dealing with complexity
- **A systems perspective** is useful for understanding the relationships among business units and organizational events.

“It’s the SYSTEM’s fault!”

“The SYSTEM is down.”

“My SYSTEM can’t be beat!”

“Don’t buck the SYSTEM.”

THE SYSTEMS VIEW

- Each piece needs to be well-designed, but the pieces also need to work well *together*

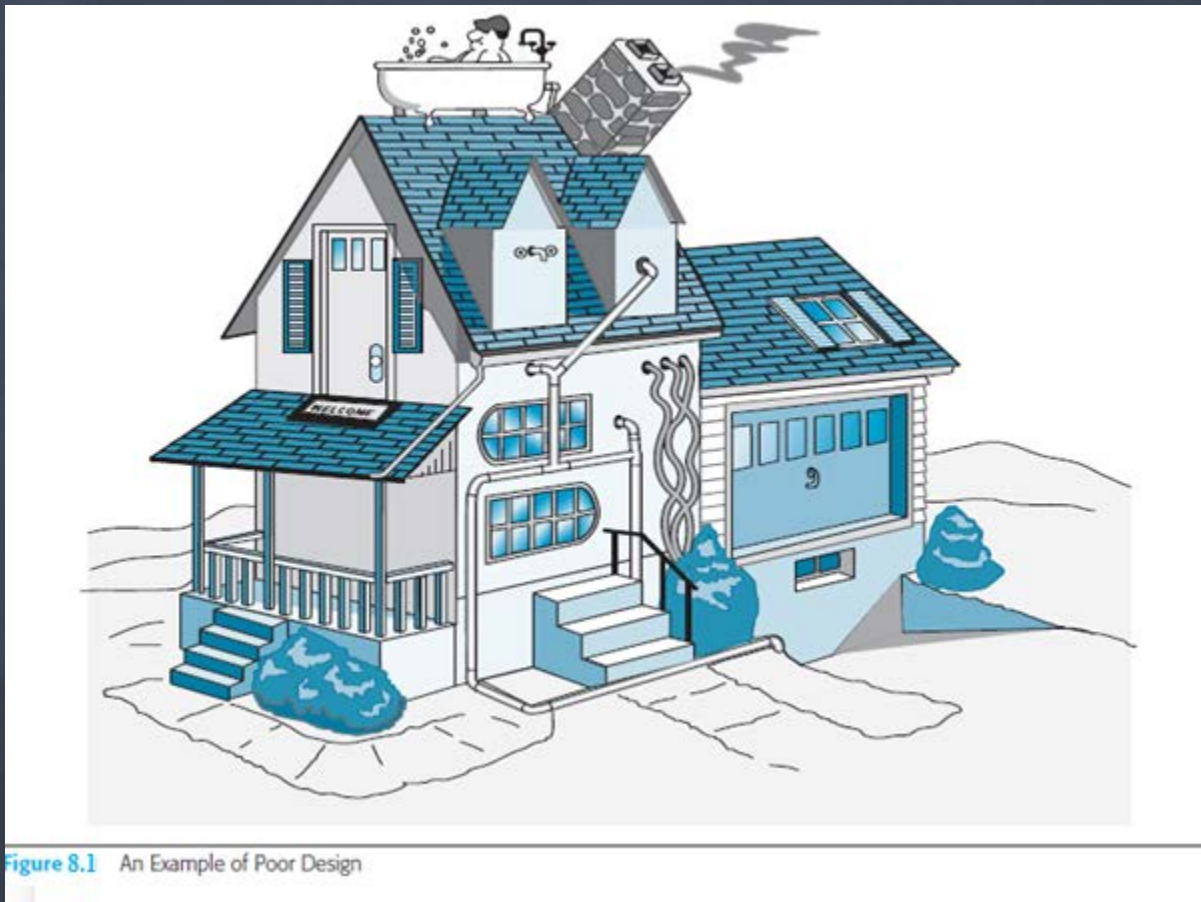
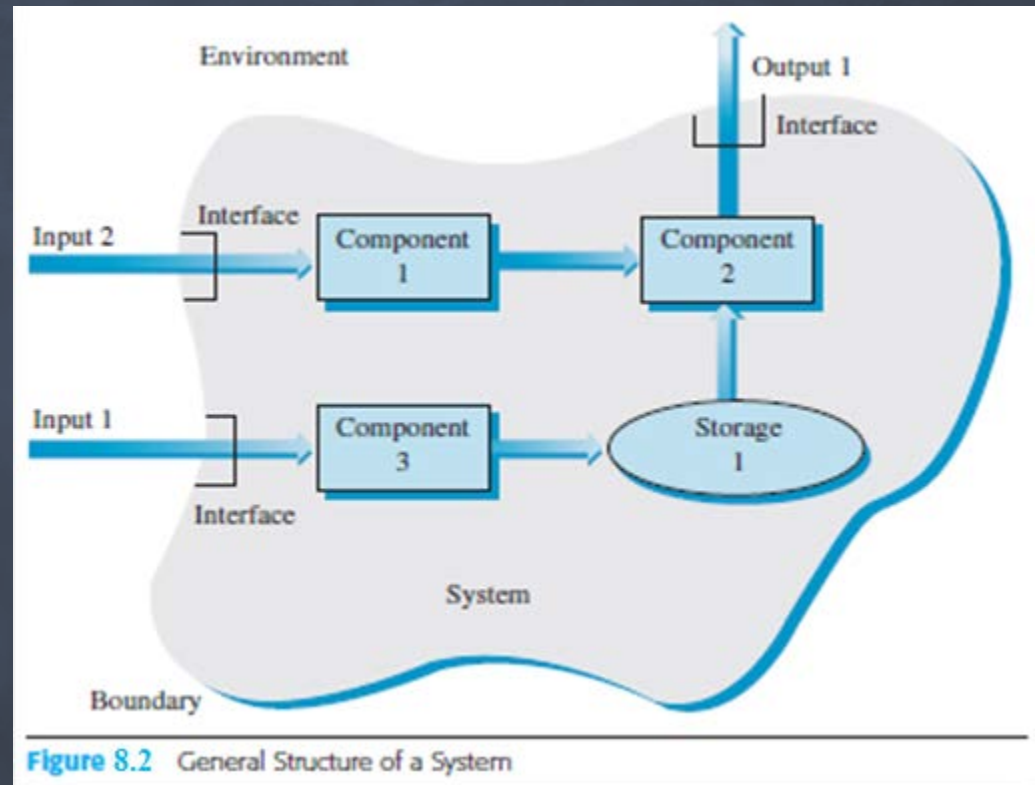


Figure 8.1 An Example of Poor Design

THE SYSTEMS VIEW

Seven key system elements:

1. Boundary
2. Environment
3. Inputs
4. Outputs
5. Components
6. Interfaces
7. Storage



SEVEN KEY SYSTEM ELEMENTS

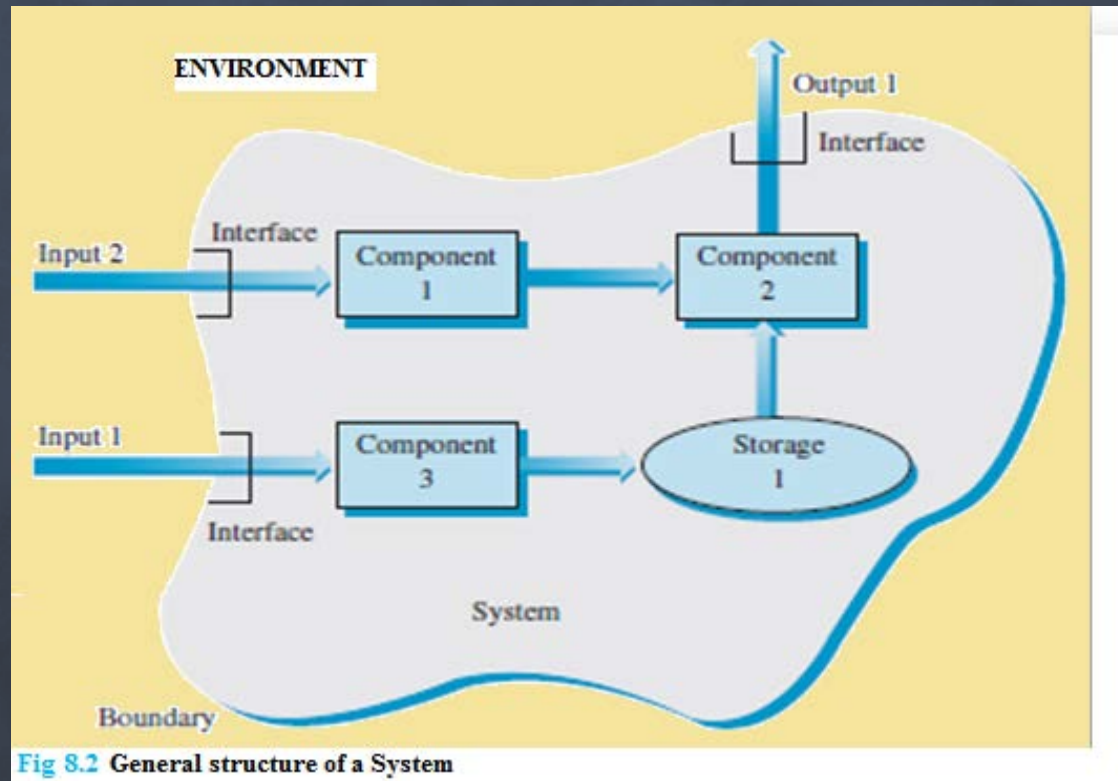
1. BOUNDARY

- Delineation of which elements are within the system and which are outside
- Where to draw the boundary depends on:
 - What can be controlled
 - What scope is manageable within a given time frame
 - The impact of a boundary change

SEVEN KEY SYSTEM ELEMENTS

2. ENVIRONMENT

- Everything outside the system



SEVEN KEY SYSTEM ELEMENTS

3. INPUTS

Resources from the environment that are consumed and manipulated within the system

4. OUTPUTS

Resources or products provided to the environment by the activities within the system

SEVEN KEY SYSTEM ELEMENTS

5. COMPONENTS

- Activities or processes within the system that transform inputs into intermediate forms or that generate system outputs
- Some system components can be viewed as systems with their own sets of interrelated components = subsystems

SEVEN KEY SYSTEM ELEMENTS

6. INTERFACES

The place where two components or the system and its environment meet or interact

7. STORAGE

Holding areas used for the temporary and permanent storage of information, energy, materials, etc.

KEY SYSTEM ELEMENTS : PAYROLL EXAMPLE

System	Payroll	Sales Tracking
Inputs	Time cards Vouchers	Customer orders Customer returns of goods
Outputs	Paychecks W-2 forms	Monthly sales by product Monthly sales by territory
Components	Calculate total pay Subtract deduc- tions	Accumulate sales by product and compare to forecast
Interfaces	Match time cards to employees Sort paychecks by department	Translate cus- tomer zip code into territory code
Storage	Employee benefits Pay rates	Product list Sales history Sales forecasts

FIGURE 8.3 System Component Examples

COMPONENT DECOMPOSITION

Sales Summary System

Produce Sales Summary Subsystem

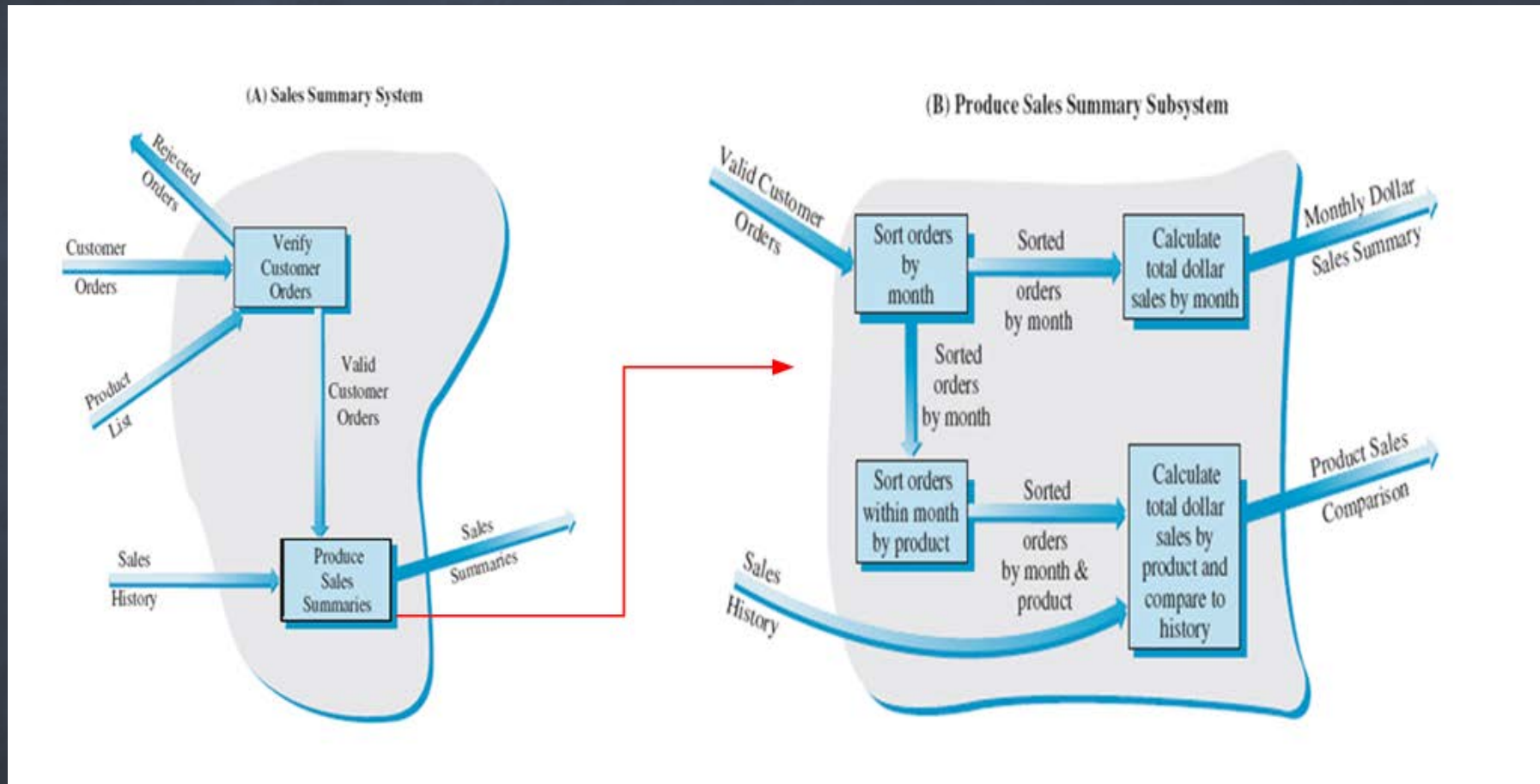


FIGURE 8.4

COMPONENT DECOMPOSITION

- **Hierarchical decomposition:** the process of breaking a system down into successive levels of subsystems
- Goals of hierarchical decomposition:
 - Cope with system complexity
 - Analyze or change only part of the system
 - Design and build each subsystem at different times
 - Direct the attention of a target audience
 - Allow components to operate more independently

INTERFACES

Functions of **Interfaces** include:

- **Filtering**

- Disposing of useless data (or noise)

- **Coding/decoding**

- Translating data from one format into another

- **Error detection and correction**

- Checking for compliance to standards and for consistency

- **Buffer**

- Allowing two subsystems to work together without being tightly synchronized

- **Security**

- Rejecting unauthorized requests for data and providing other protection mechanisms

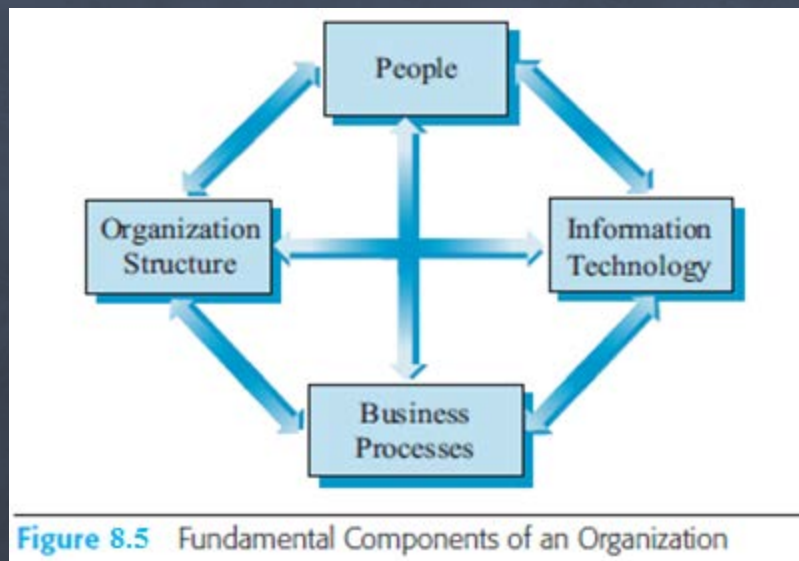
- **Summarizing**

- Condensing large volumes of input to reduce the amount of work needed by subsequent subsystems

THE SYSTEMS VIEW

Organizations as systems

- One useful framework for examining *how information systems fit into organizational systems* is based on the Leavitt diamond
- Four fundamental components in an *organization*:



THE SYSTEMS VIEW

- Leavitt Diamond tells us that:
 - *If one component is changed, the others will likely be affected as well*
- Example: New software
 - Business processes need to be redesigned
 - Organizational structures might need to be modified
 - People have to be trained

SYSTEMS ANALYSIS AND DESIGN (SA&D)

Five Key Design Principles for Information Systems

- *Two principles stem from key systems characteristics:*
 1. Choose an appropriate scope
 - Selecting the boundary for the IS greatly influences complexity and success of the project
 2. Logical before physical
 - You must know *what* an IS is to do before you can specify *how* a system is to operate

SYSTEMS ANALYSIS AND DESIGN (SA+D)

- *Three principles are problem-solving steps:*
 3. A problem is actually a set of problems and an appropriate strategy is to keep breaking down a problem into smaller, more manageable problems
 4. A single solution is not usually obvious to all stakeholders, so alternative solutions representing all parties should be generated before a final solution is selected
 5. The problem and your understanding of it could change, so a staged approach that incorporates reassessments and incremental commitment to a solution is best

BUSINESS PROCESSES

- Beginning in the early 1990s many organizations changed from a more functional approach to a more process-oriented approach to better compete

Business Process: Chain of activities required to achieve an outcome -- such as order fulfillment or materials acquisition

BUSINESS PROCESSES

- Business Process gurus in the early 1990s urged companies to *radically* change the way they did business -- by starting with a “clean slate” and utilizing information technology

“Don’t automate;
obliterate!”

- Michael Hammer

*Hammer, Michael. 1990. “Reengineering work: Don’t automate, obliterate.”
Harvard Business Review 68 (July-August): 104-112*

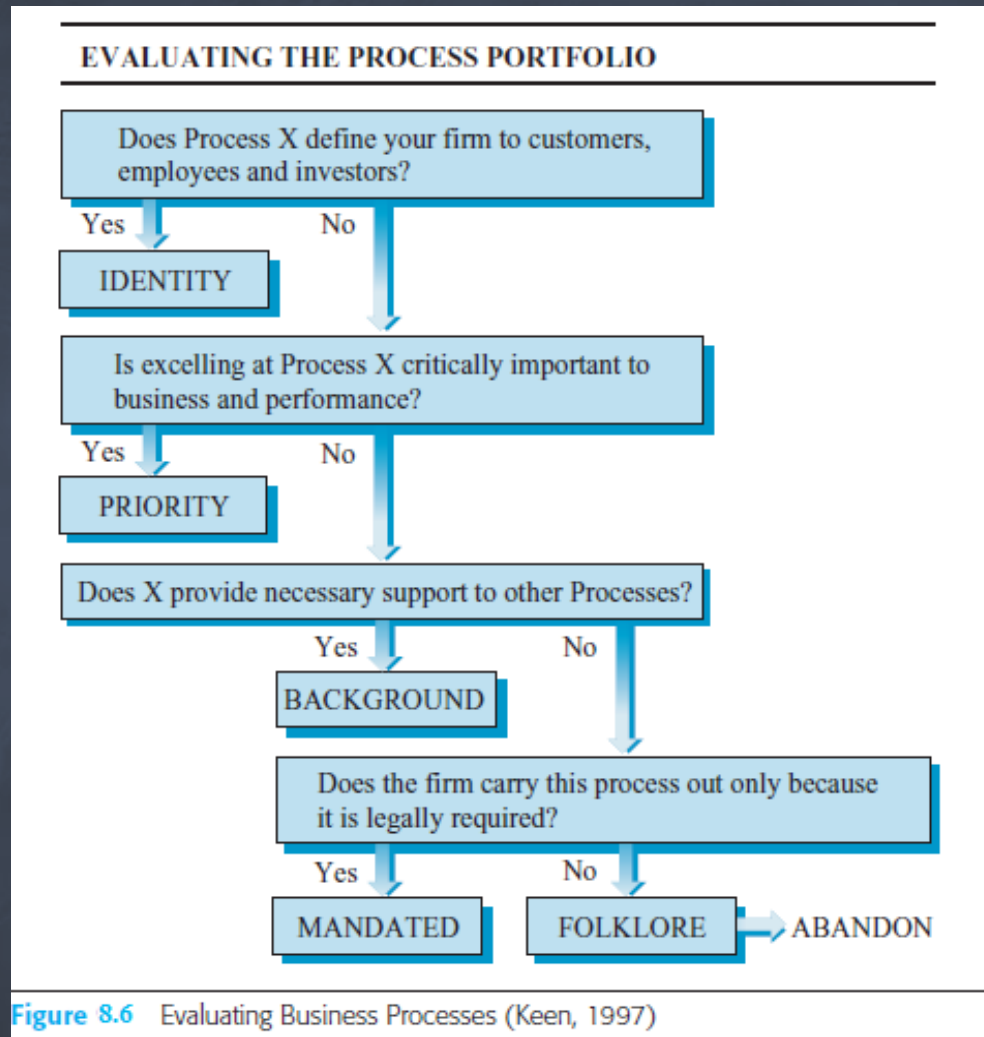
BUSINESS PROCESS REENGINEERING (BPR)

Business Process Reengineering (BPR): Radical business redesign initiatives that attempt to achieve dramatic improvements in business processes by questioning the assumptions, or business rules, that underlie the organization's structures and procedures

EARLY BPR EXAMPLES

- Accounts Payable at Ford Motor Company
 - 75% improvement gains after assumptions were questioned and a reengineered solution was identified
- Mutual Benefit Life Insurance
 - Changed a process that involved 19 people in five departments so that it could be accomplished by one person
 - Time to issue a policy decreased from 3 weeks to 3 hours

EVALUATING BUSINESS PROCESSES



HAMMER'S 6 PRINCIPLES FOR BPR

Six Key Principles for Redesigning Business Processes

- Organize business processes around outcomes, not tasks
- Assign those who use the output to perform the process
- Integrate information processing into the work that produces the information
- Create a virtual enterprise by treating geographically distributed resources as though they were centralized
- Link parallel activities instead of integrating their results
- Have people who do the work make all the decisions, and let controls built into the system monitor the process

Hammer 1990

WAYS THAT IT CAN ENABLE A NEW BUSINESS PROCESS

Old Ways to Work	Information Technology	New Ways to Work
Field personnel (such as sales and customer support staff) need to physically be located in an office to transmit and receive customer and product data	Portable computers with communications software and secure networks that allow remote access to company data	Field personnel access data and respond to messages wherever they are working
Client data is collected in different databases to support different points of contact with the client	Centralized databases that capture transactions from different parts of the business and are accessible via a network	Client data can be accessed simultaneously by employees working in different business units
Only experts can do a complex task (<i>see Mutual Benefit Life Insurance example</i>)	Expert systems that have knowledge rules used by company experts when they do this task	Generalists can do a complex task previously only done by an expert

Figure 8.7 How IT Enables New Ways to Work

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

- 3 SDLC Phases:

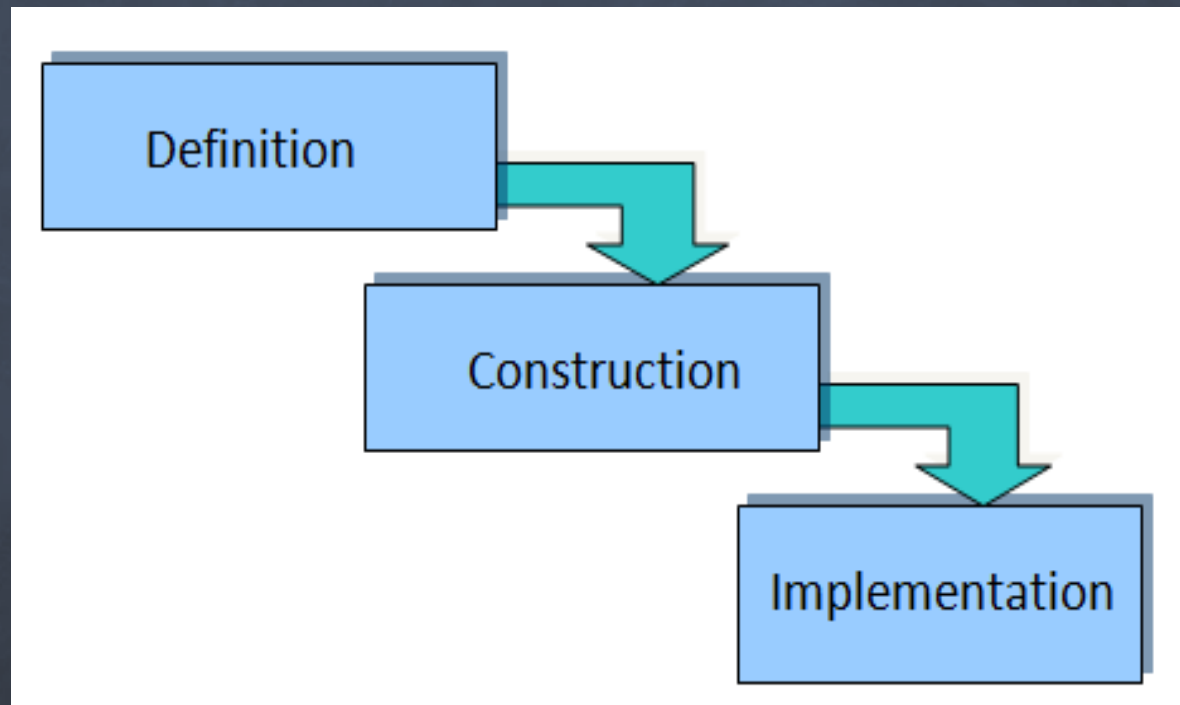


Figure 8.8

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

Definition: end users and systems analysts conduct a multistep analysis of the current business operations and the information system or systems in the area of concern

Construction: designing, building, and testing of a system that satisfies the requirements developed in the Definition phase

Implementation: install the new system, which often involves converting data and procedures from an old system

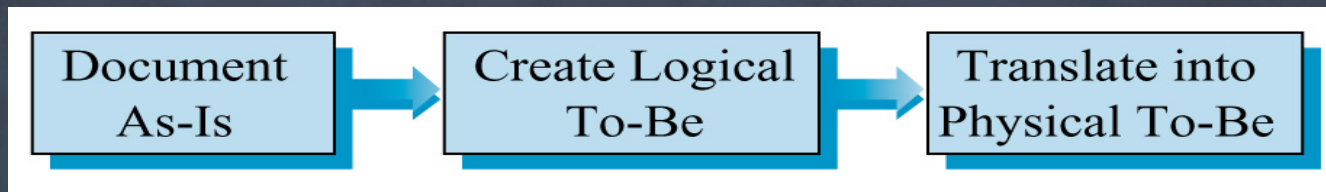
STRUCTURED TECHNIQUES

Tools to document system needs, requirements, functional features, dependencies, and design decisions

- **Procedural-oriented:**
 - Most common approach
 - Include data-oriented, sequential, process-oriented activities
- **Object-oriented:**
 - Newer approach
 - Works well for GUIs and multimedia applications

PROCEDURE- ORIENTED TECHNIQUES

- Describe what you have, define what you want, and describe how you will make what you want.



1. As-Is (what you have)
2. Logical To-Be (what you want)
3. Physical To-Be (how to make what you want)

PROCEDURE- ORIENTED TECHNIQUES

1. As-Is

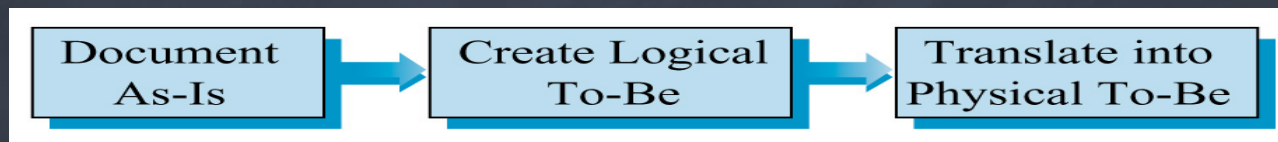
- Identifies existing processes, external participants, other databases or applications, and inputs and outputs

2. Logical To-Be

- Describes “what” rather than “how”
- High-level model of a nonexistent new system
- Identifies processes and data
- Does not identify who does activity, where accomplished, or type of hardware or software

3. Physical To-Be

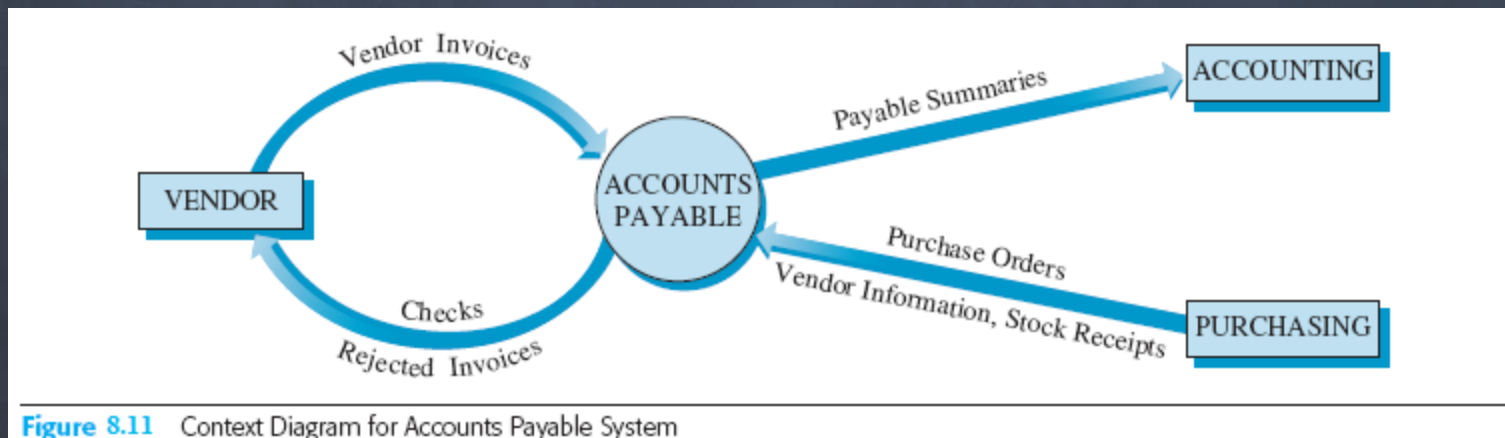
- Maps the logical requirements to available technology



PROCEDURE- ORIENTED TECHNIQUES

Context Diagram

- Diagrams system with regard to other entities and activities with which it interacts



PROCEDURE- ORIENTED TECHNIQUES

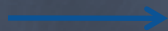
Data Flow Diagram (DFD)

- Diagrams the flows of information through the system
- Four symbols represent:

- External Entity



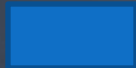
- Data Flow



- Process



- Data Store



PROCEDURE- ORIENTED TECHNIQUES: TOP-LEVEL DFD EXAMPLE

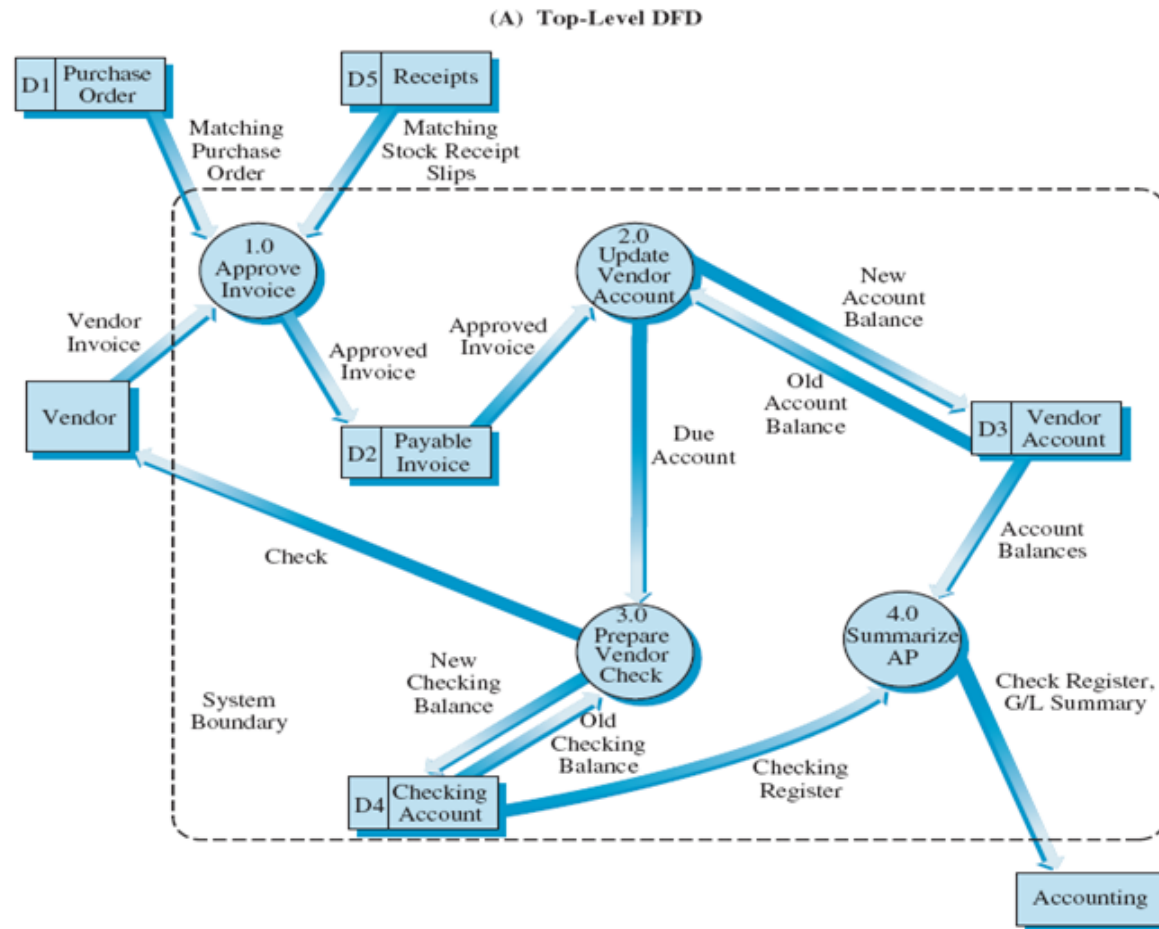


Figure 8.13

Two Levels of Data Flow Diagram for Accounts Payable System

PROCEDURE- ORIENTED TECHNIQUES

Data Dictionary/Directory

- Used to define data elements

Accounts Payable Project Data Dictionary Entry for PO Number

Label	PO Number
Alternate Names	Purchase Order Number. PO Number. PO#
Definition	Unique identifier for an individual purchase order: alpha character designates the division. The five digit number is assigned in sequential order at the time of creation.
Example	C07321
Field Name	PO_Num
Input Format	A##### (single alpha followed by five integers, no spaces or symbols allowed)
Output Format	Same as input format
Edit Rules	No values below 1000 allowed in numeric portion: currently using A-E as division code indicators.
Additional Notes	At conversion to the former system in 1991, numbers below 1000 were discontinued. Each division writes about 700-1,000 purchase orders per year. PO Numbers cannot be re-used.
Storage Type	Alphanumeric, no decimals
Default Value	None
Required	Each purchase order must have one PO Number.

Prepared by: JDAustin

Date: 8/27/17

Version No.: 1

Figure 8.14 Data Dictionary Sample Entry

PROCEDURE- ORIENTED TECHNIQUES

Entity-Relationship Diagram (ERD)

- Used to define relationships among entities

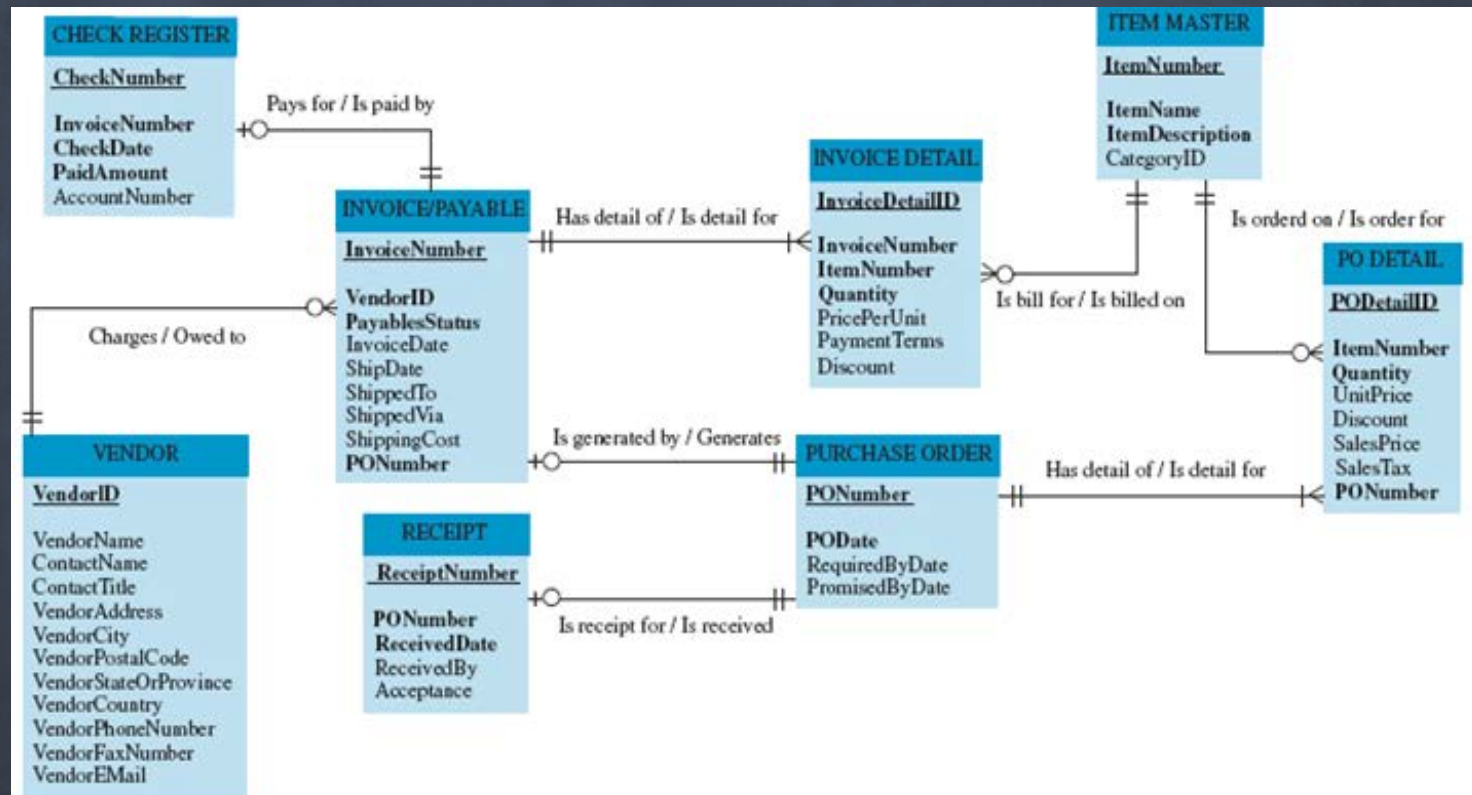


Figure 8.17 Entity-Relationship Diagram for Accounts Payable Database

PROCEDURE- ORIENTED TECHNIQUES

Physical To-Be Model

- Draft Layout of screen interface design

Vendor Invoice Form

* Indicates Required Field
For existing Invoice, enter InvoiceNumber
For existing Vendor, enter VendorID

InvoiceNumber * **NEW** VendorID **NEW** Address

PONumber * Vendor Name City

PayablesStatus * Contact Name St/Province

InvoiceDate mm/dd/yyyy Contact Title Postal Code

Shipped To Vendor Phone Country

Shipped Via Vendor Fax E-Mail

Shipping Cost

Click NEW button to add a new Detail, Click in Detail and REMOVE button to delete Detail

InvoiceDetailID *	ItemNumber *	Quantity *	PricePerUnit	PaymentTerms	Discount

NEW **REMOVE** **SUBMIT FORM** **CLEAR FORM**

Figure 8.18 Vendor Invoice Maintenance Form

OBJECT- ORIENTED (O-O) TECHNIQUES

- Primary advantages:
 - Facilitates object reuse & quick prototyping

	Procedural Approach	Object-Oriented Approach
<i>Defining the Task</i>	A team of business managers prepares a detailed design document specifying, as precisely as possible, how the program should do the task.	The O-O programmer searches a library of objects (prewritten chunks of software) looking for those that could be used for the business task.
<i>The Process</i>	Programmers divide up the design and write thousands of lines of code from scratch. If all goes well, the pieces work together as planned and the system fulfills the design requirements.	Within days, a few objects have been put together to create a bare-bones prototype. The business user gets to “test-drive” the prototype and provide feedback; by repeatedly refining and retesting the prototype, the business gets a system that fulfills the task.
<i>Elapsed Time</i>	Months.	Weeks.

FIGURE 8.20 The Promise of Object-Oriented Approaches (Based on Verity and Schwartz, 1991)

OBJECT-ORIENTED CONCEPTS

- Encapsulation
 - An object contains data and related operations
 - Allows loosely coupled modules and reuse
- Inheritance
 - One class of objects can inherit characteristics from others
- Polymorphism
 - The ability to treat child objects the same as parent objects (i.e., call methods exactly the same)

OBJECT-ORIENTED TECHNIQUES

Unified Modeling Language (UML)

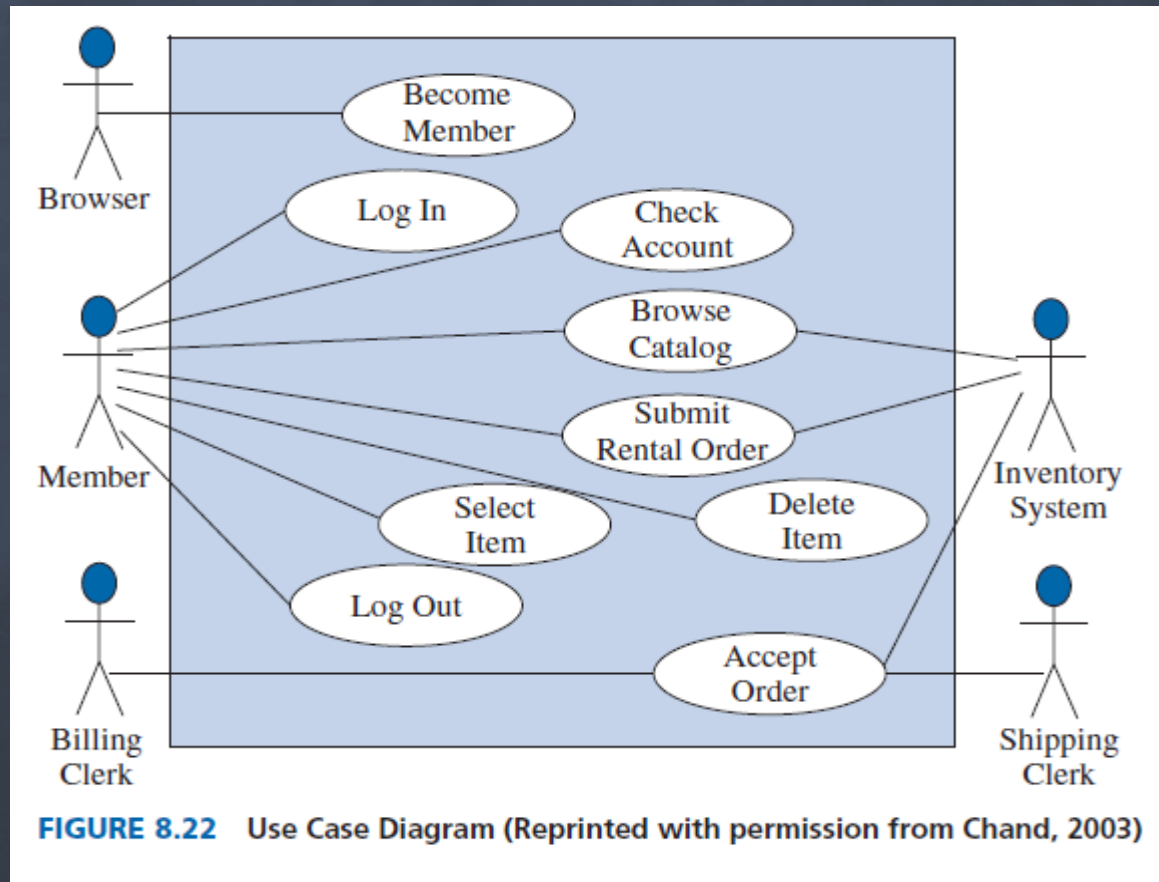
- A set of standardized techniques and notations for O-O analysis and design
- Examples of UML diagrams:
 - Use Case diagram
 - Sequence diagram
 - Class diagram

UML
UML Diagrams

UNIFIED MODELING LANGUAGE (UML)

Use Case Design

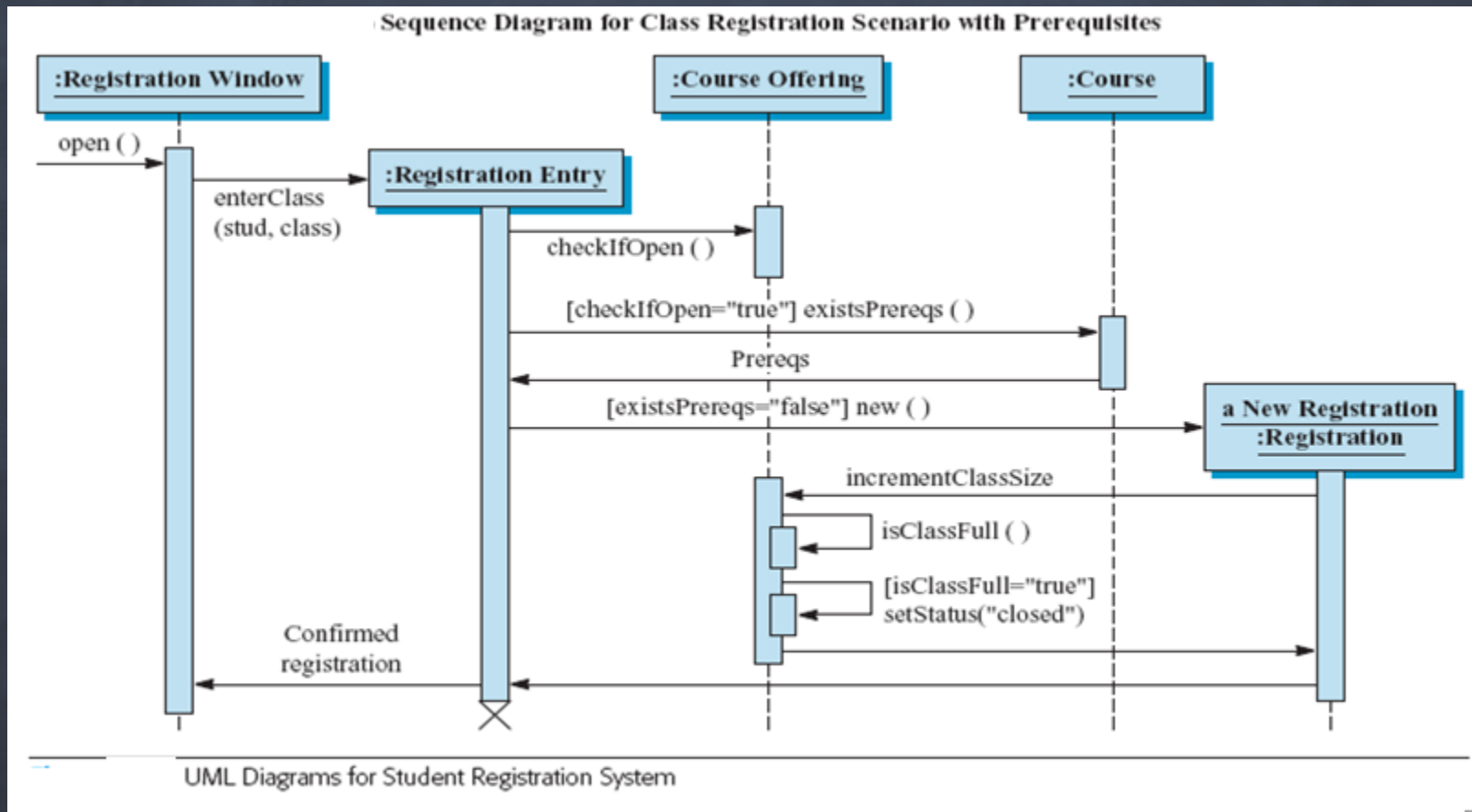
- Represents the interaction of users with the system



UNIFIED MODELING LANGUAGE (UML)

Sequence Diagram

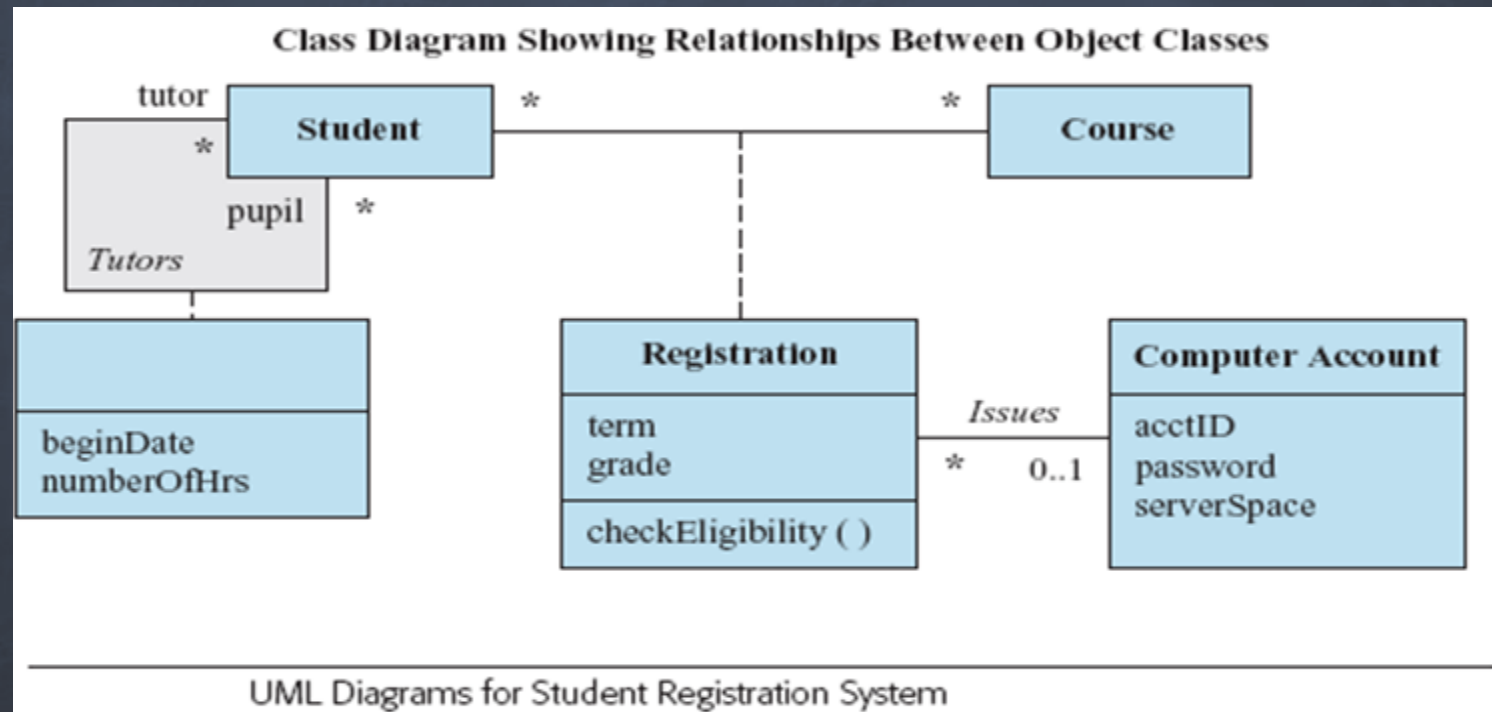
- Captures the messages that pass between objects



UNIFIED MODELING LANGUAGE (UML)

Class Diagram

- Represents each object's attributes, methods, and relationships with other objects



INFORMATION SYSTEMS CONTROLS

- Controls can be built into an information system, to mitigate some business risks, throughout the SDLC process.
- Three types of control mechanisms
 - Management policies
 - Operating procedures
 - Auditing function

Life Cycle Phase	Control Mechanism
Definition and Construction	• Methodology Standards • Validation Rules and Calculations • System Testing
Implementation	• Security • Backup and Recovery • Auditing Roles

Figure 8.24 Pre- and Post-Installation Controls

SYSTEMS CONCEPTS SUMMARY

- Systems Thinking is a hallmark of good management in general.
- Systems characteristics are important for IS work:
 - Determining the system boundary
 - Component decomposition
 - Designing a system interface
- Structured techniques are still most common, but Object-Oriented techniques (including UML) have become more prevalent
- Systems controls need to be identified and implemented throughout the systems development cycle

MANAGING INFORMATION TECHNOLOGY

7th EDITION

CHAPTER 9

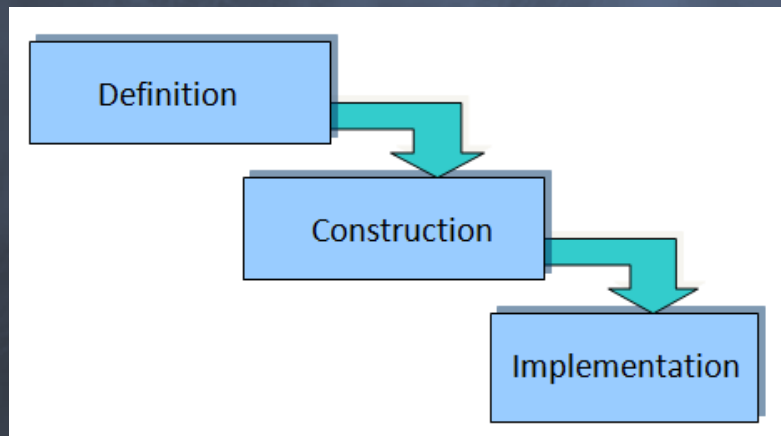
METHODOLOGIES FOR CUSTOM SOFTWARE DEVELOPMENT

METHODOLOGIES FOR CUSTOM SOFTWARE DEVELOPMENT

- Large firms purchase software packages whenever feasible, but development of custom software still highly important
- Custom methodology also used by software companies who develop products for many different buyers
- Approaches for developing custom applications:
 - Traditional Systems Development Life Cycle (SDLC)
 - Prototyping
 - Rapid Application Development (RAD)
 - Agile Development

SYSTEMS DEVELOPMENT LIFE CYCLE

- **Systems development life cycle (SDLC)**
 - Highly structured process for developing customized applications
 - Requires a lot of documentation and formal reviews at end of each major step
 - Output from one step = input to next step (Waterfall model)



SDLC Waterfall

SYSTEMS DEVELOPMENT LIFE CYCLE

SDLC Waterfall: 8 Steps in 3 phases

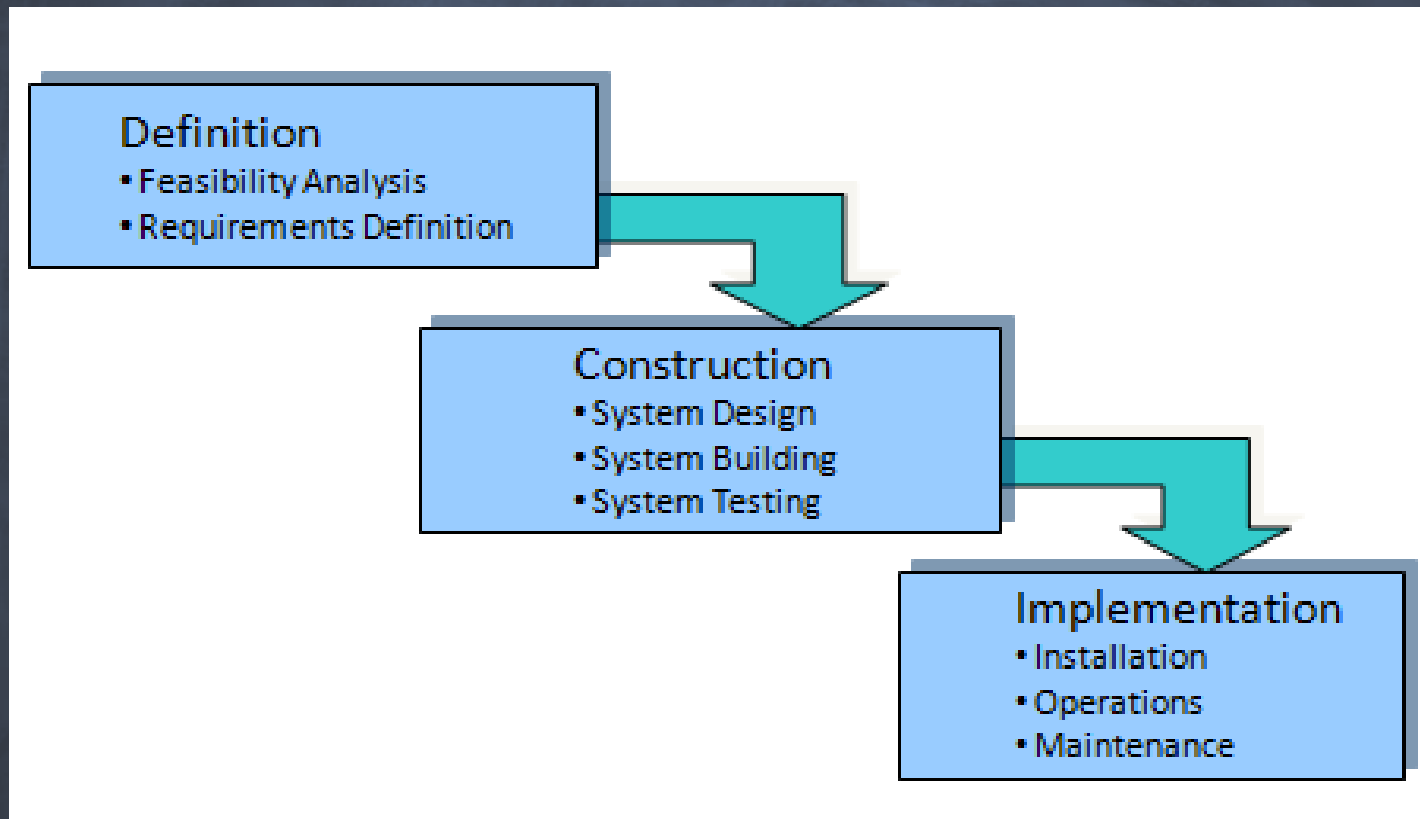


Figure 9.1

SYSTEMS DEVELOPMENT LIFE CYCLE

Typical SDLC project costs by Steps in 3 Phases

Development Activities	Percentage of Total Cost	Dollar Cost
Definition Phase		
Feasibility analysis	5	\$ 50,000
Requirements definition	25	250,000
Construction Phase		
System design	15	150,000
Coding and initial testing	15	150,000
System testing	13	130,000
Documentation and procedures	12	120,000
Implementation Phase		
Installation planning, data cleanup, and conversion	15	150,000
Total	100%	\$1,000,000

FIGURE 9.2 Cost Breakdown for \$1 Million SDLC Project

SDLC DEFINITION PHASE

Definition Phase

Feasibility Analysis (3 types)

Technical

Operational

Economic

SDLC DEFINITION PHASE

Technical Feasibility

- Primary responsibility of the IS analyst
- Based on:
 - Knowledge of current and emerging technological solutions
 - IT expertise of in-house personnel
 - Anticipated infrastructure needed to both develop and support the proposed system

SDLC DEFINITION PHASE

Operational Feasibility

- Primary responsibility of the business manager
- Entails assessing the degree to which a proposed system addresses the business issues that gave rise to the idea for a new information system

Economic Feasibility

- Business managers and IS analysts work together to prepare a cost/benefit analysis
- IS analyst responsible for establishing the developmental costs for the project

SDLC DEFINITION PHASE

Feasibility analysis

- **Typical Deliverable: 10-20 page document:**
 - Executive overview and recommendations
 - Description of what system would do and how it would operate
 - Analysis of costs and benefits
 - Development plan

SDLC DEFINITION PHASE

Requirements Definition

- Focuses on processes, data flows, and data interrelationships rather than a specific physical implementation
- Requirements are gathered by:
 - Interviewing individuals or work groups
 - Reviewing documents
 - Observing employees doing their jobs

SDLC DEFINITION PHASE

Requirements Definition

- **Deliverable = systems requirements document:**
 - Detailed descriptions of inputs and outputs, processes used to convert input data to outputs
 - Formal diagrams and output layouts
 - Revised cost/benefit analysis
 - Revised plan for remainder of project

Approved by business managers before next phase begins

SDLC CONSTRUCTION PHASE

Construction Phase

- Begins only after the systems requirements document from the Definition phase is approved
- Three steps:
 - System design
 - System building
 - System testing

SDLC CONSTRUCTION PHASE

System Design

- Includes:
 - Deciding what hardware and software to use
 - Designing structure and content of databases
 - Defining programs and their interrelationships
- Critical step for quality system:

Accurate	Reliable
Auditable	Robust
Changeable	Secure
Efficient	User friendly
Flexible	Well documented

Figure 9.3 Characteristics of High Quality Systems

SDLC CONSTRUCTION PHASE

System Design

- **Deliverable:** detailed design document
 - Models, such as diagrams of system's physical structure
 - Descriptions of databases
 - Detailed specification for each program in the system
 - Plan for the remaining steps of the Construction phase

SDLC CONSTRUCTION PHASE

System Building

- Includes:
 - Producing the computer programs
 - Developing or enhancing the databases and files to be used by the system
 - Procuring new hardware and support software
- Documentation is a major mechanism of communicating among members of the project team

SDLC CONSTRUCTION PHASE

System Testing – by IS specialists

- Time-intensive step (if executed well)
 - Each module of code is tested
 - Modules are assembled into subsystems and tested
 - Subsystems are combined and entire system is integration tested

SDLC CONSTRUCTION PHASE

System Testing – by Users

User Acceptance Testing: Ensures that the system performs reliably and does what it is supposed to do in the user environment

Who might participate in User Testing?

SDLC IMPLEMENTATION PHASE

Implementation Phase

- Success of this phase is highly dependent upon business manager involvement
 - Installation
 - Operations
 - Maintenance

SDLC IMPLEMENTATION PHASE

Installation

- Includes:
 - Building files and databases
 - Converting relevant data from one or more old systems to the new system
 - Training system users

Installations that involve converting data from an old system to a new one can be as difficult to implement as systems to automate totally new functions or processes

SDLC IMPLEMENTATION PHASE

Installation

Four Approaches to Convert from an old System:

- **Parallel:** organization operates old system in parallel with new system until new system is working sufficiently
- **Pilot:** new system is introduced to only one part of the organization first
- **Phased:** new system is implemented one component at a time
- **Cutover:** old system is totally abandoned as soon as the new system is implemented

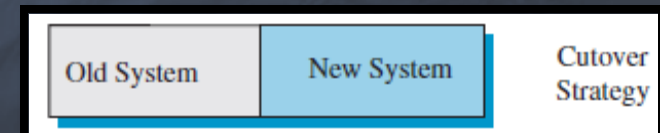
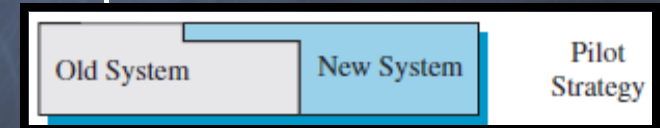
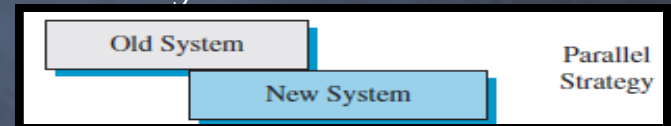


Fig 9.4

SDLC IMPLEMENTATION PHASE

Operations

- New application that is operational is referred to as in “production mode”
- Project team is disbanded at this time or shortly thereafter
- Requires two types of documentation
 - *System* documentation for IS specialists who operate and maintain the system
 - *User* documentation for those who use the system

SDLC IMPLEMENTATION PHASE

Maintenance

- The process of making changes to a system after it has been put into “production mode”
- Reasons for maintenance
 - Correct errors in the system
 - Adapt the system to changes in the business environment
 - Enhance or improve the system beyond the original system requirements

SDLC IMPLEMENTATION PHASE

Maintenance

- In the past, maintenance step has incurred about 80% of total costs over a system's life
- In the 1990s, systems development resources heavily devoted to system maintenance versus new system development:
 - 75% to run and maintain existing systems
 - 35% to build/buy new systems

SDLC IMPLEMENTATION PHASE

Maintenance

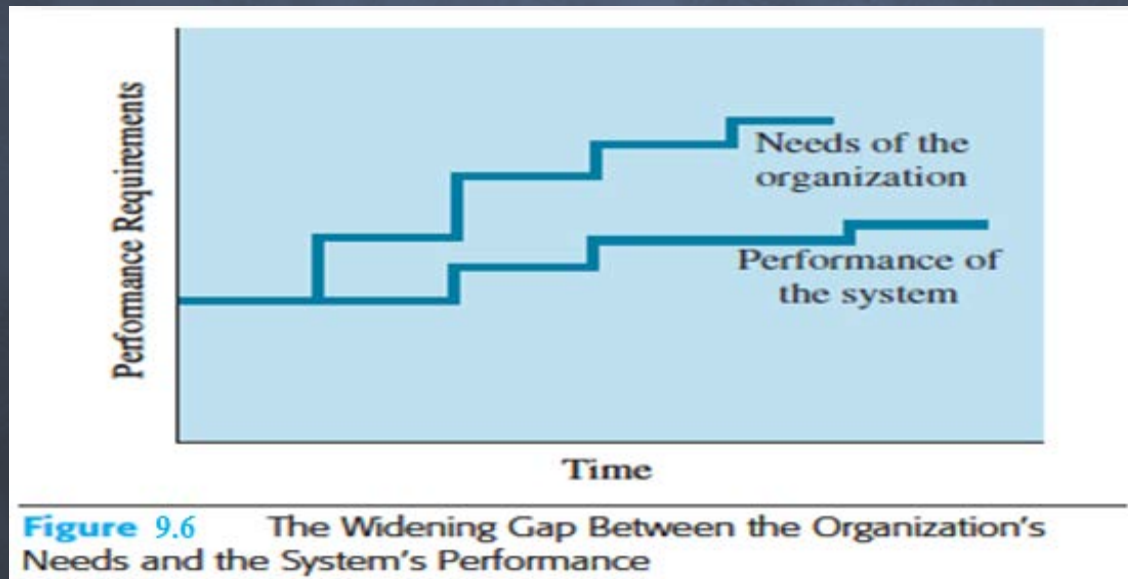
- Common Problems:
 - Documentation may not be updated when changes to the system are made, causing problems for future maintenance
 - Changes to one part of the system may have an unanticipated effect on other parts of the system (i.e., a *ripple effect*)
 - Programmers generally prefer new development, not maintenance, so work may go to least experienced programmers

SDLC IMPLEMENTATION PHASE

Maintenance

Business managers need to be aware that:

- Maintenance can introduce new errors into the system
- If IT resources not available, there may be long delays before a requested system change is worked on, creating gaps in system performance and the needs of an organization



SYSTEMS DEVELOPMENT LIFE CYCLE

The SDLC project team

- Usually a temporary team for specific project
- Includes appropriate representatives from business units, as well as IS personnel
- Led by **project manager**
 - Usually from IS, but can be from business unit (or both)
 - Responsible for success of project: *delivering quality system on time and within budget*

SYSTEMS DEVELOPMENT LIFE CYCLE

Three project characteristics associated with successful outcomes:

1. Manageable project size
2. Accurate requirements definition
 - Cost of corrections increases as development life cycle advances
3. Executive sponsorship

SYSTEMS DEVELOPMENT LIFE CYCLE

Advantages and Disadvantages

Advantages

- Highly structured, systematic process
- Careful controls on the process, including impact on related systems
- Thorough requirements definition
- Clear milestones with business management sign-offs

Disadvantages

- Does not account well for evolving requirements during project
- Time-consuming (and costly) process
- Top-down commitment required
- Resistance in practice to feedback or going back to a prior phase
- Role of system user may be narrowly defined
- Fixed milestone and completion dates

FIGURE 9.7 Advantages and Disadvantages of Traditional SDLC Approach

PROTOTYPING METHODOLOGY

Prototyping

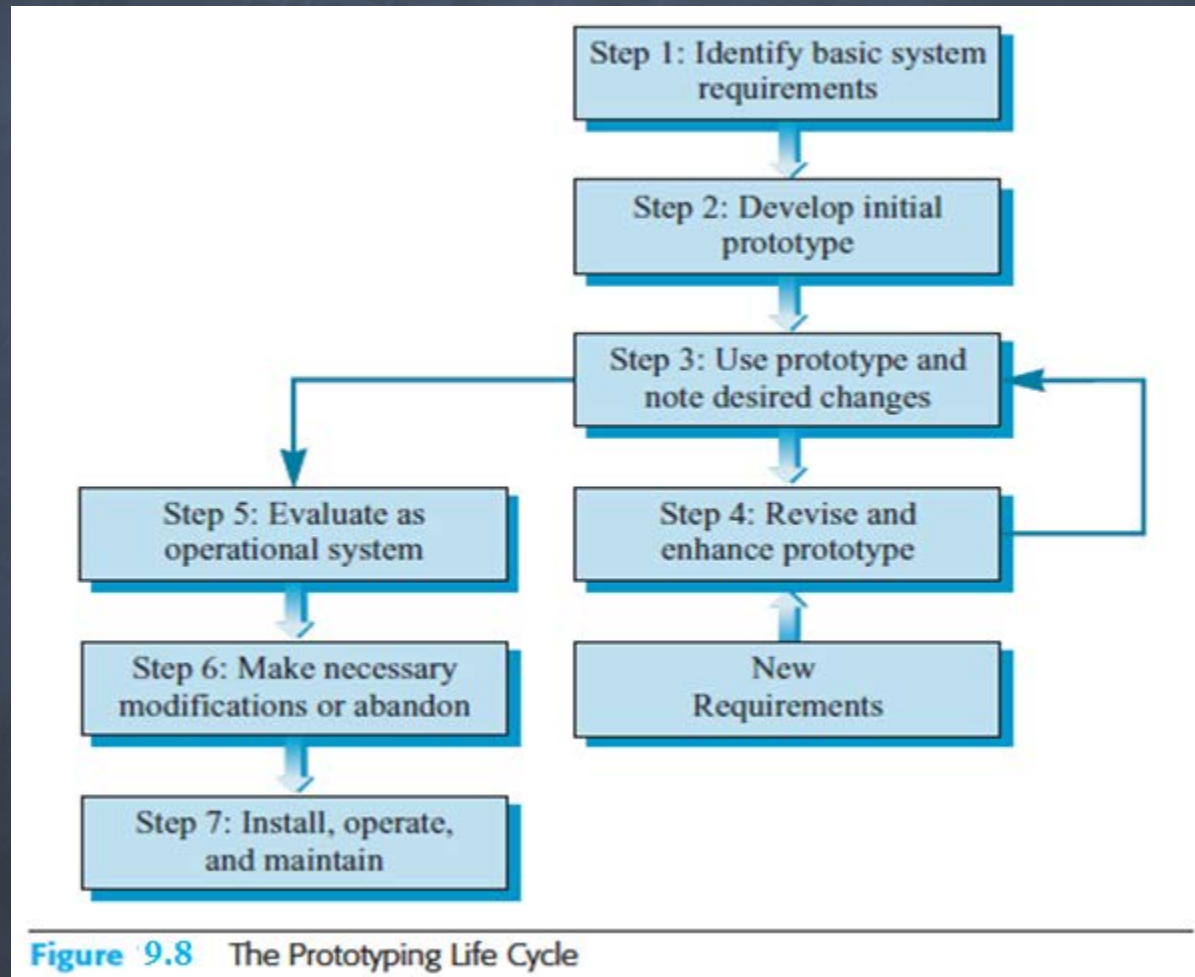
- Is a type of evolutionary development process: enables creation of system (or part of system) quickly, then system is revised after initial trial(s) by user(s)
- Takes advantage of fourth generation procedural languages and relational database management systems
- Can be used as a complete alternative to the SDLC or within an SDLC process

PROTOTYPING METHODOLOGY

- **Examples of Prototyping goals:**
 - “First-of-a-series” – a completely operational prototype used as a pilot
 - “Selected features” – only some essential features included in prototype, more added later

PROTOTYPING METHODOLOGY

The Prototyping Steps:



PROTOTYPING METHODOLOGY

Project Team:

- IS team members who can quickly build systems using advanced tools
- Business users committed to working closely with IS developers to try out and refine prototype

PROTOTYPING METHODOLOGY

Advantages:

- Only basic requirements needed at front end
- Used to develop systems that radically change how work is done, so users can evaluate
- Allows firms to explore use of new technology
- Working system available for testing more quickly
- Less strong top-down commitment needed at front end
- Costs and benefits can be derived after experience with initial prototype
- Initial user acceptance likely to be higher

PROTOTYPING METHODOLOGY

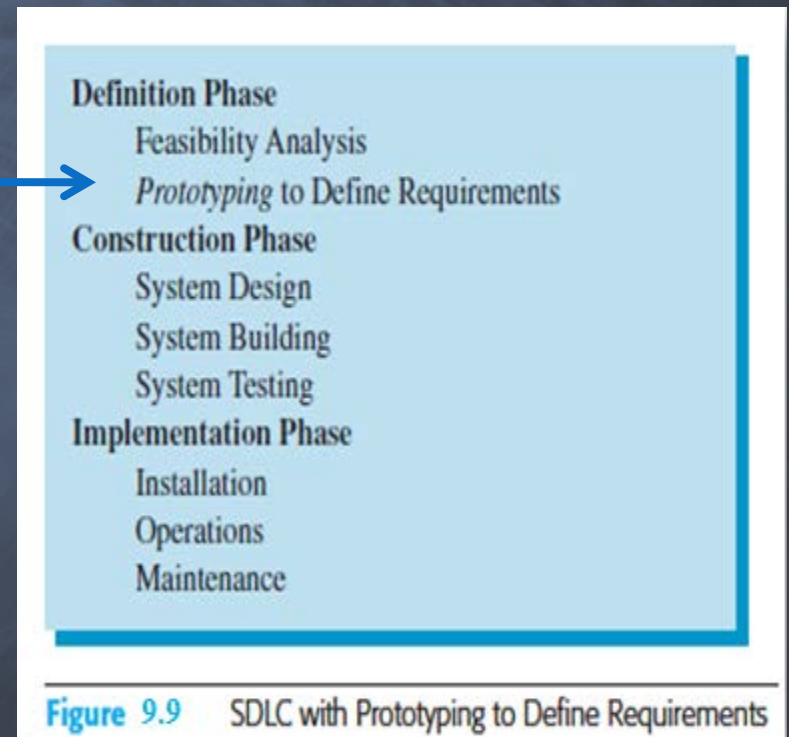
Disadvantages:

- End prototype may lack security and control features needed for the final system
- May not undergo rigorous testing
- Final documentation may be less complete
- More difficult to manage user expectations

PROTOTYPING METHODOLOGY

Prototyping *within* an SDLC Definition Phase

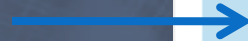
1. To help users define system requirements
– such as input and output screens



PROTOTYPING METHODOLOGY

Prototyping *within* an SDLC Definition Phase

2. Used for a pilot implementation of a working prototype before Construction using SDLC approach



Prototyping/Piloting Phase

Determine Basic Requirements

Prototype the System

Pilot the Prototype

SDLC Construction Phase

System Design Modifications

System Building

System Testing

SDLC Implementation Phase

Installation

Operations

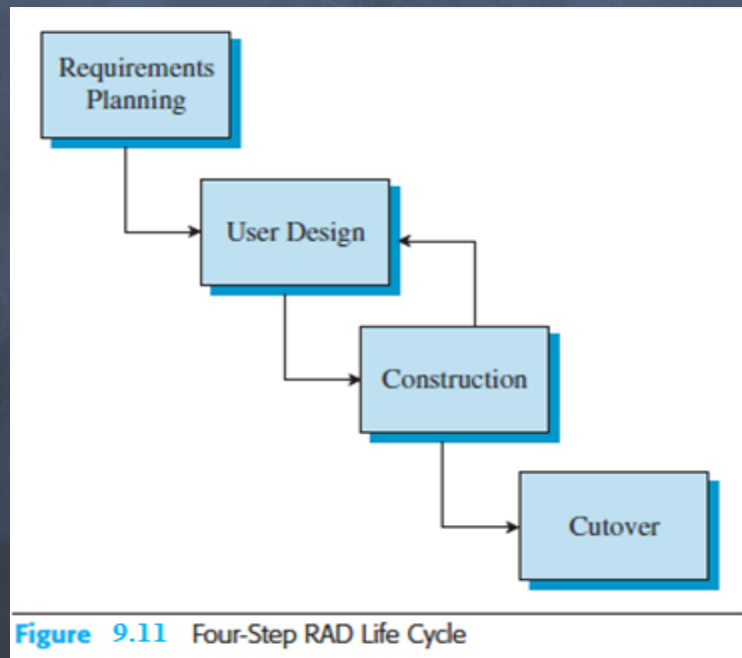
Maintenance

Figure 9.10 Prototyping/Piloting Phase Replaces SDLC Definition Phase

RAPID APPLICATION DEVELOPMENT (RAD)

RAPID APPLICATION DEVELOPMENT (RAD)

- Hybrid methodology: combines aspects of SDLC and prototyping
- Goal = produce a system more quickly than an SDLC alone

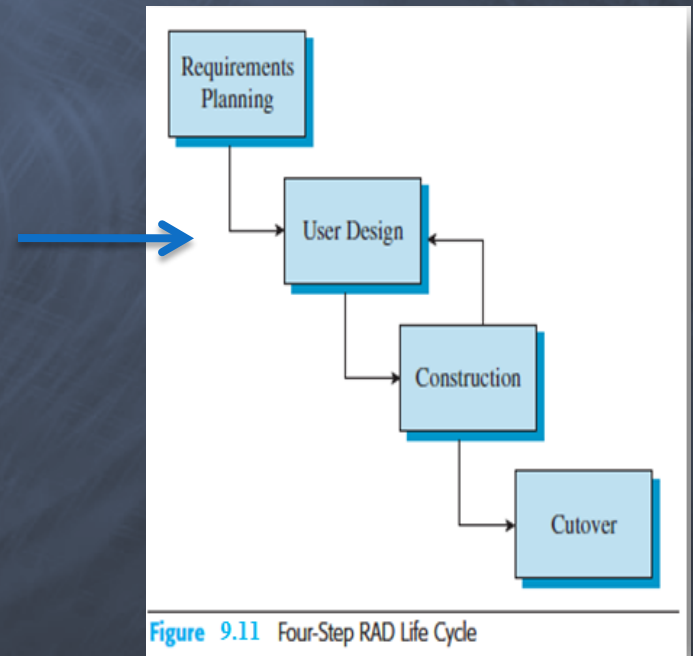


RAPID APPLICATION DEVELOPMENT (RAD)

A common RAD technique is:

JOINT APPLICATION DEVELOPMENT (JAD)

- Team of users and IS specialists engage in an intense and structured process in order to minimize total time required for gathering information from multiple participants



RAPID APPLICATION DEVELOPMENT (RAD)

Advantages

- Dramatic savings in development time
- Focuses on essential system requirements
- Ability to rapidly change system design at user request

Disadvantages

- Quality may be sacrificed for speed
- Time-consuming commitments for key user personnel
- Possible shortcuts on internal standards and module reusability

FIGURE 9.13 RAD Advantages and Disadvantages

CASE TOOLS

Computer Aided Software Engineering (CASE)

Any software tool used to automate one or more steps of a software development methodology

- *Diagramming tools:* support graphic representations for process, data, and control structure diagrams
- *Computer display and report generators:* used to prototype user interface for input (screen displays, forms) and reports as part of requirements definition
- *Analysis tools:* automatic checkers for missing, inconsistent, or incorrect specifications in diagrams, forms, and reports
- *Central repository:* integrated storage of system specs, diagrams, reports, and project management documents
- *Documentation generators:* produce technical and user documentation in standard formats
- *Code generators:* automatic generation of program and database definition code from diagrams, forms, reports, and other design documents

Figure 9.12 Types of CASE Tools (Adapted from Valacich, George, and Hoffer, 2006)

AGILE DEVELOPMENT

- Alternative methodology for smaller projects
- Based on four key values:
 - Simplicity
 - Communication
 - Feedback
 - Courage

AGILE DEVELOPMENT

- **eXtreme programming (XP)**
 - Programmers write code in pairs
 - Use simple design and frequent testing
 - Three program design characteristics:
 1. System must communicate everything you want to communicate
 2. System must contain no duplicate code
 3. System should have the fewest number of components as possible

AGILE DEVELOPMENT

Scrum

- Well-orchestrated movement of project updates between team members
 - Similar to the coordination in a rugby scrum
- Emphasizes:
 - Independent project teams
 - Coordination and communication between and within teams
 - Iterative and continuous monitoring of work
 - Highly efficient work methods
- Approach utilizes:
 - Daily Scrum meeting
 - Scrum of Scrum meeting
 - Sprint planning meeting
 - Sprint review meeting

Rugby Scrum