MANAGING INFORMATION TECHNOLOGY

Lecture 8
BASIC SYSTEMS CONCEPTS AND TOOLS
By: Prof. Lili Saghafi
AGENDA

• The Systems View

• Business Processes

• Systems Development Life Cycle and Structured Techniques

• Information Systems Controls
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
THE SYSTEMS VIEW

Seven key system elements:

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

<table>
<thead>
<tr>
<th>System</th>
<th>Payroll</th>
<th>Sales Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Time cards Vouchers</td>
<td>Customer orders</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Customer returns of goods</td>
</tr>
<tr>
<td>Outputs</td>
<td>Paychecks</td>
<td>Monthly sales by product</td>
</tr>
<tr>
<td></td>
<td>W-2 forms</td>
<td>Monthly sales by territory</td>
</tr>
<tr>
<td>Components</td>
<td>Calculate total pay</td>
<td>Accumulate sales by product and compare to</td>
</tr>
<tr>
<td></td>
<td>Subtract deductions</td>
<td>forecast</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Match time cards to employees</td>
<td>Translate customer zip code into territory</td>
</tr>
<tr>
<td></td>
<td>Sort paychecks by department</td>
<td>code</td>
</tr>
<tr>
<td>Storage</td>
<td>Employee benefits</td>
<td>Product list</td>
</tr>
<tr>
<td></td>
<td>Pay rates</td>
<td>Sales history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sales forecasts</td>
</tr>
</tbody>
</table>
COMPONENT DECOMPOSITION

Sales Summary System

Produce Sales Summary Subsystem
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
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

Copyright © 2011 Pearson Inc. publishing as Prentice Hall
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
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

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

Figure 8.6  Evaluating Business Processes (Keen, 1997)
# HAMMER’S 6 PRINCIPLES FOR BPR

## Six Key Principles for Redesigning Business Processes

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

Hammer 1990
WAYS THAT IT CAN ENABLE A NEW BUSINESS PROCESS

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

*Figure 8.7 How IT Enables New Ways to Work*
SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

- 3 SDLC Phases:

Figure 8.8
a standard approach that results in the production of well documented, quality software.
The diagram illustrates the relationship between SDLC (Software Development Life Cycle) phases, control objectives, and management control domains.

**SDLC Phases**:
- Project Definition
- User Requirements Definition
- System Requirements Definition
- Analysis and Design
- System Build/Prototype/Pilot
- Implementation and Training
- Sustainment

**Control Objectives**:
- Project Definition
- User Requirements Definition
- System Requirements Definition
- Analysis and Design
- System Build/Prototype/Pilot
- Implementation and Training
- Sustainment

**Management Control Domains**:
- Planning & Organization
- Acquisition & Implementation
- Delivery & Support
- Monitoring
Agile Software Development

**Iteration -1**
Select the Project
- Identify potential projects
- Prioritize potential projects
- Develop initial vision
- Consider project feasibility

**Iteration 0**
Warm Up
- Active stakeholder participation
- Obtain funding and support
- Start building the team
- Initial requirements envisioning
- Initial architecture envisioning
- Setup environment

**Construction Iterations**
Deliver a working system which meets the changing needs of stakeholders.
- Active stakeholder participation
- Collaborative development
- Model storming
- Test driven design (TDD)
- Confirmatory testing
- Evolve documentation
- Internally deploy software

**Release (End Game)**
Deploy Release N into Production
- Active stakeholder participation
- Final system testing
- Final acceptance testing
- Finalize documentation
- Pilot test the release
- Train end users
- Train production staff
- Internally deploy software

**Production**
Operate and Support Release N
- Operate the system
- Support the system
- Identify defects and enhancements
- Remove the final version of the system
- Data conversion
- Migrate users
- Update enterprise models

**Retirement**
Remove the system completely from production

Start work on Release N + 1
**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

Figure 8.11 Context Diagram for Accounts Payable System
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
Data Modeling
By : Prof. Lili Saghafi
Data Modeling and Database Models

• **Content** - What data should be collected?
• **Access** - What data should be given to what users?
• **Logical structure** - How will the data be organized to make sense to a particular user?
• **Physical organization** - Where will the data actually be located?
Data Modeling

• Enterprise data modeling (DFD)
• Planned data redundancy
• Data model
• Entity-relationship diagrams
Structured Analysis to develop Application

- Examines inputs, outputs, and processes
- Common method
- Process-centered technique
- Uses **three main tools**
  - Data flow diagrams (DFDs)
  - Data dictionary
  - Process descriptions
- Tools can be applied using computer-aided software engineering (CASE) tools
Data Flow Diagrams

- Data flow diagrams (DFDs) are graphical aids that describe an information system.
- DFDs represent a logical model that shows what a system does, not how it does it.
Systems analysts often use visual aids during presentations.
Data Flow Diagrams

• **Data flow diagram symbols**
  – Four basic symbols
    • Process
    • Data flow
    • Data store
    • External entity
  – Two popular symbol sets
    • Gane and Sarson
    • Yourdon
FIGURE 4-2 Data flow diagram symbols, symbol names, and examples from the Gane and Sarson and Yourdon symbol sets.
Data Flow Diagrams

– Process symbol
  • Symbol is a rectangle with rounded corners
  • Documented with process descriptions
  • Receive input data and produces output
  • Output has a different form, or content, or both
  • Details are shown in a process description
  • In DFDs the process symbol appears as a black box, underlying details not shown
Data Flow Diagrams

- Data flow symbol
  - Symbol is a line with an arrowhead showing direction
  - A path for data to move from one part of the system to another
  - Might represent one or many pieces of data
  - At least one data flow must enter and exit each process
FIGURE 4-3  Examples of correct combinations of data flow and process symbols.
Data Flow Diagrams

– Data flow symbol
  • Incorrect process and data flow combinations cause problems
    ➔ Spontaneous generation (miracle)
    ➔ Black hole
    ➔ Gray hole
Examples of incorrect combinations of data flow and process symbols: APPLY INSURANCE PREMIUM has no inputs and is called a spontaneous generation process. CALCULATE GROSS PAY has no outputs and is called a black hole process. CALCULATE GRADE has an input that is obviously unable to produce the output. This process is called a gray hole.
Data Flow Diagrams

– **Data store symbol**

  • Symbol is a rectangle open on the right side
  • Data store also is called a data repository
  • Represents data that is retained for later processing
  • Must be connected to a process with a data flow
  • Must have at least one outgoing and incoming data flow
FIGURE 4-5 Examples of correct uses of data store symbols in a data flow diagram.
Examples of incorrect uses of data store symbols: two data stores cannot be connected by a data flow without an intervening process, and each data store should have an outgoing and incoming data flow.
Data Flow Diagrams

– **External entity symbol**
  
  • Symbol is a square, usually shaded
  • Represents a person, organization, or other system that provides data or receives output from the system
  • External entities are called **terminators**
    
    ➔ **Source** (supplies data to the system)
    ➔ **Sink** (receives data from the system)
  • Must follow specific rules for connecting DFD symbols
FIGURE 4-7
Examples of correct uses of external entities in a data flow diagram.
Examples of incorrect uses of external entity symbols. An external entity must be connected by a data flow to a process, and not directly to a data store or to another external entity.
<table>
<thead>
<tr>
<th>DATA FLOW THAT CONNECTS</th>
<th>OKAY TO USE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A process to another process</td>
<td>YES</td>
</tr>
<tr>
<td>A process to an external entity</td>
<td>YES</td>
</tr>
<tr>
<td>A process to a data store</td>
<td>YES</td>
</tr>
<tr>
<td>An entity to another entity</td>
<td>NO</td>
</tr>
<tr>
<td>An entity to a data store</td>
<td>YES</td>
</tr>
<tr>
<td>A data store to another data store</td>
<td>YES</td>
</tr>
</tbody>
</table>

**FIGURE 4-9** Rules for connecting processes, data stores, and external entities in a DFD.
Data Flow Diagrams

• **Context diagrams**
  – Top-level view that shows the overall boundaries of the system
  – Represent the results of fact-finding
  – One process symbol, numbered 0 (zero) is drawn in the center
  – Data flows connect the process to the entities
  – Abbreviated symbols can be used to identify entities
FIGURE 4-10  Context diagram DFD for the grading system.
FIGURE 4-11 Context diagram DFD for an order system.
Data Flow Diagrams

• Conventions for data flow diagrams
  – Each context diagram must fit on one page
  – Process name in the context diagram should be the name of the information system
  – Use unique names within each set of symbols
  – Do not cross lines
  – Use abbreviated identifications
  – Use a unique reference number for each process symbol
Context diagram DFD for a manufacturing system.
Data Flow Diagrams

- **Diagram 0**
  - Displays more detail than the context diagram
  - Shows entities, major processes, data flows, and data stores
  - Shows entities, major processes, data flows, and data stores
  - Other characteristics
    1. Can contain diverging data flows
    2. Exploded (partitioned or decomposed) version of process 0
    1. Diagram 0 is the child of the parent context diagram
    4. Also can be called an overview or level 0 diagram
    1. Can contain functional primitives
FIGURE 4-13  Context diagram and diagram 0 for the grading system.
Diagram 0 DFD for the order system.
DFD fragment for the Checkout/Sales Transaction event
Large companies depend on powerful order entry systems to handle thousands of daily orders and to maintain a high level of customer service and satisfaction.

FIGURE 4-15
What are the mistakes in this diagram (DFD)?
PROCEDURE- ORIENTED TECHNIQUES

Data Dictionary/Directory

- Used to define data elements

<table>
<thead>
<tr>
<th>Label</th>
<th>PO Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate Names</td>
<td>Purchase Order Number. PO Number. PO#</td>
</tr>
<tr>
<td>Definition</td>
<td>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.</td>
</tr>
<tr>
<td>Example</td>
<td>C07321</td>
</tr>
<tr>
<td>Field Name</td>
<td>PO_Num</td>
</tr>
<tr>
<td>Input Format</td>
<td>A##### (single alpha followed by five integers, no spaces or symbols allowed)</td>
</tr>
<tr>
<td>Output Format</td>
<td>Same as input format</td>
</tr>
<tr>
<td>Edit Rules</td>
<td>No values below 1000 allowed in numeric portion: currently using A–E as division code indicators.</td>
</tr>
<tr>
<td>Additional Notes</td>
<td>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.</td>
</tr>
<tr>
<td>Storage Type</td>
<td>Alphanumeric, no decimals</td>
</tr>
<tr>
<td>Default Value</td>
<td>None</td>
</tr>
<tr>
<td>Required</td>
<td>Each purchase order must have one PO Number.</td>
</tr>
</tbody>
</table>

Prepared by: JDAustin  Date: 8/27/11  Version No.: 1
PROCEDURE- ORIENTED TECHNIQUES

Entity-Relationship Diagram (ERD)
- Used to define relationships among entities
PROCEDURE- ORIENTED TECHNIQUES

Physical To-Be Model

• Draft Layout of screen interface design
OBJECT-ORIENTED (O-O) TECHNIQUES

- Primary advantages:
  - Facilitates object reuse & quick prototyping

---

<table>
<thead>
<tr>
<th>Defining the Task</th>
<th>Procedural Approach</th>
<th>Object-Oriented Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A team of business managers prepares a detailed design document specifying, as precisely as possible, how the program should do the task.</td>
<td>The O-O programmer searches a library of objects (prewritten chunks of software) looking for those that could be used for the business task.</td>
</tr>
</tbody>
</table>

| The Process                | 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. |

| Elapsed Time               | 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 activity diagram for the Enroll in University use case.
conceptual class diagram.
System-level sequence diagram.
System use case diagram.
Enroll in University

- Registrar
- Applicant

- Student

- International Student

- <<extend>>
- Perform Security Check
- Enroll Family Member in University

- <<include>>
- Enroll in Seminar
Use Case Design

- Represents the interaction of users with the system
Sequence Diagram

UNIFIED MODELING LANGUAGE (UML)
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

<table>
<thead>
<tr>
<th>Life Cycle Phase</th>
<th>Control Mechanism</th>
</tr>
</thead>
</table>
| Definition and Construction | • Methodology Standards  
                              | • Validation Rules and Calculations  
                              | • System Testing               |
| Implementation            | • Security  
                              | • Backup and Recovery  
                              | • Auditing Roles               |
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
Lecture 7
E-BUSINESS SYSTEMS
• By: Prof. Lili Saghafi