Learning Objectives

- Explain the many reasons for creating information system models
- Describe three types of models and list some specific models used for analysis and design
- Explain how events can be used to define activities and use cases
- Identify and analyze events to which a system responds
Learning Objectives (continued)

◆ Explain how the concept of “things” in the problem domain also defines requirements
◆ Explain the similarities and the differences between data entities and objects
◆ Identify and analyze data entities and domain classes needed in the system
◆ Read, interpret, and create an entity-relationship diagram
◆ Read, interpret, and create a class diagram
Overview

- Document functional requirements by creating models

- Models created during analysis phase activity – *Define system requirements*

- Two concepts help identify functional requirements in the traditional approach and object-oriented approach
  - Events that trigger use cases
  - Things in the users’ work domain
Models and Modeling

- Analyst describes information system requirements using a collection of models
- Complex systems require more than one type of model
- Models represent some aspect of the system being built
- Process of creating models helps analyst clarify and refine design
- Models assist communication with system users
Reasons for Modeling (Figure 5-2)

- Learning from the modeling process
- Reducing complexity by abstraction
- Remembering all of the details
- Communicating with other development team members
- Communicating with a variety of users and stakeholders
- Documenting what was done for future maintenance/enhancement
Types of Models

- Different types of models are used in information systems development

  - **Mathematical** – formulas that describe technical aspects of the system

  - **Descriptive** – narrative memos, reports, or lists that describe aspects of the system

  - **Graphical** – diagrams and schematic representations of some aspect of the system
Some Descriptive Models (Figure 5-3)

A narrative description of processing requirements as verbalized by an RM0 phone-order representative:

“When customers call in, I first ask if they have ordered by phone with us before, and I try to get them to tell me their customer ID number that they can find on the mailing label on the catalog. Or, if they seem puzzled about the customer number, I need to look them up by name and go through a process of elimination, looking at all of the Smiths in Dayton, for example, until I get the right one. Next, I ask what catalog they are looking at, which sometimes is out of date. If that is the case, then I explain that many items are still offered, but that the prices might be different. Naturally, they point to a page number, which doesn’t help me because of the different catalogs, but I get them to tell me the product ID somehow...”

List of inputs for the RM0 customer support system:

- Item inquiry
- New order
- Order change request
- Order status inquiry
- Order fulfillment notice
- Back-order notice
- Order return notice
- Catalog request
- Customer account update notice
- Promotion package details
- Customer charge adjustment
- Catalog update details
- Special promotion details
- New catalog details
Overview of Models Used in Analysis and Design

- **Analysis phase activity named “define system requirements”**
  - Logical models
  - Provide detail without regard to specific technology

- **Design phase**
  - Physical models
  - Provide technical details
  - Extend logical models
Models Created by Analysis Activities (Figure 5-4)
Models Used in Design (Figure 5-5)

- Screen layout
- Report layout
- System flowchart
- Structure chart

- Database schema
- Network diagram
- Deployment diagram
Events, Activities, and Use Cases

- **Use Case**
  - An activity the system performs in response to a user request
  - A “case” where the system is used by actor

- **Techniques for identifying use cases**
  - Identify user goals
    - Each goal at the elementary business process (EBP) level is a use case
    - EBP – a task performed by one user, in one place in response to a business event, that adds measurable business value, and leaves system and data in consistent state
  - Event decomposition technique
  - CRUD analysis technique (create, read, update, delete)
Identifying Use Cases Based on User Goals (Figure 5-6)

<table>
<thead>
<tr>
<th>User/actor</th>
<th>User goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order clerk</td>
<td>Look up item availability</td>
</tr>
<tr>
<td></td>
<td>Create new order</td>
</tr>
<tr>
<td></td>
<td>Update order</td>
</tr>
<tr>
<td>Shipping clerk</td>
<td>Record order fulfillment</td>
</tr>
<tr>
<td></td>
<td>Record back order</td>
</tr>
<tr>
<td>Merchandising manager</td>
<td>Create special promotion</td>
</tr>
<tr>
<td></td>
<td>Produce catalog activity report</td>
</tr>
</tbody>
</table>
Event Decomposition

- Business events trigger elementary business processes (EBPs)
- EBPs are at correct level of analysis for use cases
- Identify business events to decompose system into activities/use cases
- Event decomposition is, therefore, used by
  - Traditional approach to identify activities
  - OO approach to identify use cases
Types of Events

- **External**
  - Outside system
  - Initiated by external agent or actor

- **Temporal**
  - Occur as result of reaching a point in time
  - Based on system deadlines

- **State**
  - Something inside system triggers processing need
Events Affecting a Charge Account Processing System that Lead to Use Cases (Figure 5-7)

External events occur in the environment

- “customer pays bill,” so use case is Record a payment
- “customer makes a charge,” so use case is Process a charge

Temporal events occur inside the system

- “time to send late notices,” so use case is Send late notices
- “time to produce end-of-week summary reports,” so use case is Produce summary reports

“customer changes address,” so use case is Maintain customer data

“time to send out monthly statements,” so use case is Produce monthly statements
External Event Checklist (Figure 5-8)

External events to look for include:
✓ External agent wants something resulting in a transaction
✓ External agent wants some information
✓ Data changed and need to be updated
✓ Management wants some information
Temporal Event Checklist (Figure 5-9)

Temporal events to look for include:
✓ Internal outputs needed
  ✓ Management reports (summary or exception)
  ✓ Operational reports (detailed transactions)
  ✓ Internal statements and documents (including payroll)
✓ External outputs needed
  ✓ Statements, status reports, bills, reminders
Identifying Events

- Can be difficult to determine
- Often confused with conditions and responses
- May be useful to trace a transaction’s life cycle
- Certain events left to design phase
  - System controls to protect system integrity
  - Perfect technology assumption defers events
Sequence of Actions that Lead Up to Only One Event Affecting the System (Figure 5-10)
Sequence of “Transactions” for One Specific Customer Resulting in Many Events (Figure 5-11)

- Customer requests a catalog
- Customer wants to check item availability
- Customer places an order
- Customer changes or cancels an order
- Customer wants to check order status
- Customer updates account information
- Customer returns the item
Events Deferred Until the Design Phase

(Figure 5-12)

- User wants to log on to the system
- User wants to change the password
- User wants to change preference settings
- System crash requires database recovery
- Time to back up the database
- Time to require the user to change the password

Don’t worry much about these until the design phase
Events in the RMO case

- Important external events involve customers
  - Customer checks item availability, customer places order, customer changes or cancels order

- Other external events involve departments
  - Shipping fulfills order, marketing sends promotion to customer, merchandising updates catalog

- Temporal events include periodic reports
  - Time to produce order summary reports, Time to produce fulfillment summary reports
Information about Each Event in an Event Table: Catalog of Information about Each Use Case (Figure 5-15)

<table>
<thead>
<tr>
<th>Event</th>
<th>Trigger</th>
<th>Source</th>
<th>Use case</th>
<th>Response</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer wants to check item availability</td>
<td>Item inquiry</td>
<td>Customer</td>
<td>Look up item availability</td>
<td>Item availability details</td>
<td>Customer</td>
</tr>
</tbody>
</table>

**The event that causes the system to do something.**

**Source:** For an external event, the external agent, or actor, is the source of the data entering the system.

**Response:** What output (if any) is produced by the system?

**Trigger:** How does the system know the event occurred? For external events, this is data entering the system. For temporal events, it is a definition of the point in time that triggers the system processing.

**Use case:** What does the system do when the event occurs? The use case is what is important to define for functional requirements.

**Destination:** What external agent gets the output produced?
## RMO Event Table (Figure 5-6 partial)

<table>
<thead>
<tr>
<th>Event</th>
<th>Trigger</th>
<th>Source</th>
<th>Use case</th>
<th>Response</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Customer wants to check item availability</td>
<td>Item inquiry</td>
<td>Customer</td>
<td>Look up item availability</td>
<td>Item availability details</td>
<td>Customer</td>
</tr>
<tr>
<td>2. Customer places an order</td>
<td>New order</td>
<td>Customer</td>
<td>Create new order</td>
<td>Real-time link</td>
<td>Credit bureau</td>
</tr>
<tr>
<td>3. Customer changes or cancels order</td>
<td>Order change request</td>
<td>Customer</td>
<td>Update order</td>
<td>Change confirmation</td>
<td>Customer</td>
</tr>
<tr>
<td>4. Time to produce order summary reports</td>
<td>&quot;End of week, month, quarter and year&quot;</td>
<td>Customer</td>
<td>Produce order summary reports</td>
<td>Order summary reports</td>
<td>Management</td>
</tr>
<tr>
<td>5. Time to produce transaction summary reports</td>
<td>&quot;End of day&quot;</td>
<td>Customer or management</td>
<td>Produce transaction summary reports</td>
<td>Transaction summary reports</td>
<td>Accounting</td>
</tr>
<tr>
<td>6. Customer or management wants to check order status</td>
<td>Order status inquiry</td>
<td>Customer or management</td>
<td>Look up order status</td>
<td>Order status details</td>
<td>Customer or management</td>
</tr>
</tbody>
</table>
“Things” in the Problem Domain

- Define system requirements by understanding system information that needs to be stored

- Store information about things in the problem domain that people deal with when they do their work

- Analysts identify these types of things by considering each use case in the event table

  - What things does the system need to know about and store information about?
Types of Things (Figure 5-17)

- **Tangible things**
  - airplane
  - book
  - vehicle
  - document
  - worksheet

- **Roles played**
  - employee
  - customer
  - doctor
  - patient
  - end user
  - system administrator

- **Organizational units**
  - division
  - department
  - section
  - task force
  - workgroup

- **Devices**
  - sensor
  - timer
  - controller
  - printer
  - disk drive
  - keyboard
  - display
  - window
  - mouse
  - menu
  - button

- **Sites/locations**
  - warehouse
  - branch office
  - factory
  - retail store
  - desktop

- **Incidents, events, or interactions**
  - flight
  - service call
  - logon
  - logoff
  - contract
  - purchase
  - order
  - payment
Procedure for Developing an Initial List of Things

- **Step 1:** Using the event table and information about each use case, identify all *nouns*

- **Step 2:** Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed

- **Step 3:** Refine list and record assumptions or issues to explore
  
  - See Figure 5-18 for RMO example
Characteristics of Things

◆ Relationship

- Naturally occurring association among specific things
- Occur in two directions
- Number of associations is **cardinality** or **multiplicity**
  - Binary, unary, ternary, n-ary

◆ Attribute

- One specific piece of information about a thing
Relationships Naturally Occur Between Things (Figure 5-19)
Cardinality/Multiplicity of Relationships
(Figure 5-20)

Mr. Jones has placed no order yet, but there might be many placed over time.  
Cardinality/multiplicity is zero or more—optional relationship

A particular order is placed by Mr. Smith. There can’t be an order without stating who the customer is.  
Cardinality/multiplicity is one and only one—mandatory relationship

An order contains at least one item, but it could contain many items.  
Cardinality/multiplicity is one or more—mandatory relationship
Attributes and Values (Figure 5-21)

<table>
<thead>
<tr>
<th>All customers have these attributes:</th>
<th>Each customer has a value for each attribute:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer ID</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>103</td>
</tr>
<tr>
<td>First name</td>
<td>John</td>
</tr>
<tr>
<td></td>
<td>Mary</td>
</tr>
<tr>
<td></td>
<td>Bill</td>
</tr>
<tr>
<td>Last name</td>
<td>Smith</td>
</tr>
<tr>
<td></td>
<td>Jones</td>
</tr>
<tr>
<td></td>
<td>Casper</td>
</tr>
<tr>
<td>Home phone</td>
<td>555-9182</td>
</tr>
<tr>
<td></td>
<td>423-1298</td>
</tr>
<tr>
<td></td>
<td>874-1297</td>
</tr>
<tr>
<td>Work phone</td>
<td>555-3425</td>
</tr>
<tr>
<td></td>
<td>423-3419</td>
</tr>
<tr>
<td></td>
<td>874-8546</td>
</tr>
</tbody>
</table>
Data Entities

- Things system needs to store data about in traditional IS approach

- Modeled with entity-relationship diagram (ERD)

- Requirements model used to create the database design model for relational database
Objects

- Objects do the work in a system and store information in the object-oriented approach.

- Objects have behaviors and attributes:
  - **Class** – type of thing
  - **Object** – each specific thing
  - **Methods** – behaviors of objects of the class

- Objects contain values for attributes and methods for operating on those attributes.

- An object is **encapsulated** – a self-contained unit.
Data Entities Compared with Objects
(Figure 5-22)
The Entity-Relationship Diagram (ERD)

**Figure 5-23**
A simple entity-relationship diagram

- A Customer can place zero or more Orders.
- An Order must be placed by exactly one Customer.
Cardinality Symbols of Relationships for ERD
Expanded ERD with Attributes Shown

**Figure 5-25**
An expanded ERD with attributes shown

- **Customer**
  - Cust number*
  - Name
  - Bill address
  - Home phone
  - Office phone

- **Order**
  - Order ID*
  - Order date
  - Amount

- **Order Item**
  - Item ID*
  - Quantity
  - Price

*Indicates the identifier or key
Customers, Orders, and Order Items

Figure 5-26
Customers, orders, and order items consistent with the expanded ERD

John
Order 1 Feb 4
Order 2 March 29

Mary
no orders for Mary yet!

Sara
Order 3 March 30

- First shirt
- Second shirt
- Belt
- Boots
- First sandals
- Second sandals
- First sandals
- Second sandals
- Third sandals
ERD with Many-to-Many Relationship

**Figure 5-27**
A university course enrollment ERD with a many-to-many relationship
Many-to-Many Relationship Converted to Associative Entity to Store Grade Attribute

**Figure 5-28**
A refined university course enrollment ERD with an associative entity
RMO Customer Support System ERD
(Figure 5-29)
The Class Diagram

- Unified Modeling Language (UML) diagram

- Domain model class diagram
  - Models things in the users’ work domain
  - Used to define requirements for OO (very similar to entities in ERD)

- Design class diagram
  - Models software classes
  - Adds methods as behaviors
  - Used in the design activity
UML Class Symbol (Figure 5-30)

- **Customer**
  - name
  - address
  - phone

- **Attributes**: all objects in the class have a value for each of these
  - addNew()
  - delete()
  - change()
  - connectToAccount()

- **The name of the class**

- **Methods**: what all objects of the class know how to do
Simple Domain Model Class Diagram
(Figure 5-31)

- No methods shown in domain model
  - Domain classes are not software classes
- Very similar to ERD in Figure 5-25
  - UML and domain model can be used in place of ERD in traditional approach
Multiplicity of Associations (Figure 5-32)
University Course Enrollment Domain Model Class Diagram (Figure 5-33)
Refined Model with Association Class and Grade Attribute (Figure 5-34)
More Complex Class Concepts

◆ Generalization/specialization hierarchies
  ● General superclasses to specialized subclasses
  ● Inheritance allows subclasses to share characteristics of their superclasses

◆ Whole-part hierarchies (object and its parts)
  ● Aggregation – parts can exist separately
  ● Composition – parts can’t exist separately
    ♦ Hand has fingers and thumb
A Generalization/Specialization Class Hierarchy for Motor Vehicles (Figure 5-35)

Trucks, Cars and Tractors are special types of Motor Vehicles

Sports Cars, Sedans, and Sport Utilities are special types of Cars
A Generalization/Specialization Class Hierarchy for RMO Orders (Figure 5-36)
Whole-Part Aggregation Relationships  
(Figure 5-37)
RMO Domain Model Class Diagram (Figure 5-41)
Design Class Diagram Notation:
Software Classes with Methods

Figure 5-38
A bank account system design class diagram (with methods)
Course Enrollment Design Class Diagram with Association Class (Figure 5-39)
Expanded Course Enrollment Design Class Diagram (Figure 5-40)
Where You Are Headed (Figure 5-42)
Summary

◆ Analysis phase – defines system requirements
◆ Models created to further learning process, reduce complexity, communicate with team members, and document requirements
◆ Many types of models used
  ● Mathematical, descriptive, graphical
◆ Key early step in modeling is to identify and list
  ● Events that require a use case in the system
  ● Things users deal with in work environment
Summary (continued)

- Use cases (activities) are identified from user goals and business events that trigger elementary business processes.

- Business events are memorable, can be described, and occur at a specific time and place.
  - External events, temporal events, and state events.

- Event table records event, trigger, source, use case, response, and destination.
  - A catalog of information about each use case.
Summary (continued)

◆ “Things” are what user deals with and system remembers, such as customer placing an order.

◆ Traditional approach uses entity-relationship diagrams (ERD) for data entities, attributes of data entities, and relationships between entities.

◆ Object-oriented approach uses UML class diagrams for classes, attributes, methods of class, and associations among classes.
  - Domain model class diagram (requirements activity)
  - Design class diagram (design activity)