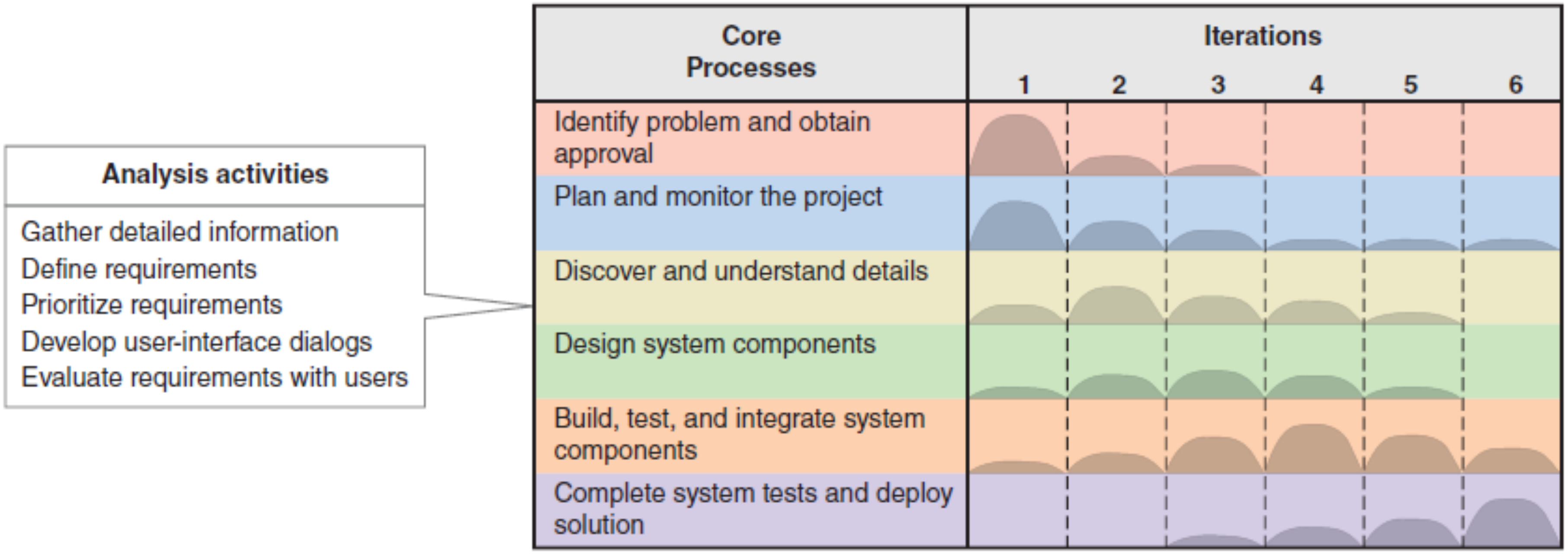
LECTURE 4: INVESTIGATING AND MODELING SYSTEM REQUIREMENTS

Instructor: Jiashu (Jessie) Zhao

Analysis Phase



Analysis activities
Gather detailed information
Define requirements
Prioritize requirements
Develop user-interface dialogs
Evaluate requirements with users
Core Processes
Iterations
1
2
3
4
5
6
Identify problem and obtain approval
Plan and monitor the project
Discover and understand details
Design system components
Build, test, and integrate system components
Complete system tests and deploy solution

Systems Analysis Activities

- Gather Detailed Information
  - Interviews, questionnaires, documents, observing business processes, researching vendors, comments and suggestions
- Define Requirements
  - Modeling functional requirements and non-functional requirements
- Prioritize Requirements
  - Essential, important, vs. nice to have
- Develop User-Interface Dialogs
  - Flow of interaction between user and system
- Evaluate Requirements with Users
  - User involvement, feedback, adapt to changes

System Requirements

- System requirements – specifications that define the functions of the new system
- Two sets of requirements:
  - Functional requirements
  - Nonfunctional requirements

Functional requirements

- Functional requirements
  - Activities system must perform (use cases)
  - Based on procedures and business functions
  - Documented in analysis models
- Examples
  - “generate electronic fund transfers”
  - “calculate payroll taxes”

Nonfunctional requirements

- Nonfunctional requirements – characteristics other than those activities the system must perform
  - Constraints and performance goals
- Includes:
  - Usability requirement
  - Reliability requirement
  - Performance requirement
  - Security requirement
  - + more
- Example:
  - “Tax amounts should be accurate, Calculate Tax amount should be easy to use”
    - Security
    - Safety
    - Privacy
  -

FURPS+ Requirements Acronym

Requirement categories	**FURPS + categories**	**Example requirements**
Functional	Functions	Business rules and processes
Nonfunctional	Usability	User interface, ease of use
	Reliability	Failure rate, recovery methods
	Performance	Response time, throughput
	Security	Access controls, encryption
	+ Design constraints	Hardware and support software
	Implementation	Development tools, protocols
	Interface	Data interchange formats
	Physical	Size, weight, power consumption
	Support	Installation and updates

Models and Modeling

- Requirements are describes by a collection of models

- Types of Models
  - Textual model– something written down, described
  - Graphical models– diagram, schematic
  - Mathematical models– formulas, statistics, algorithms

Models and Modeling

- 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

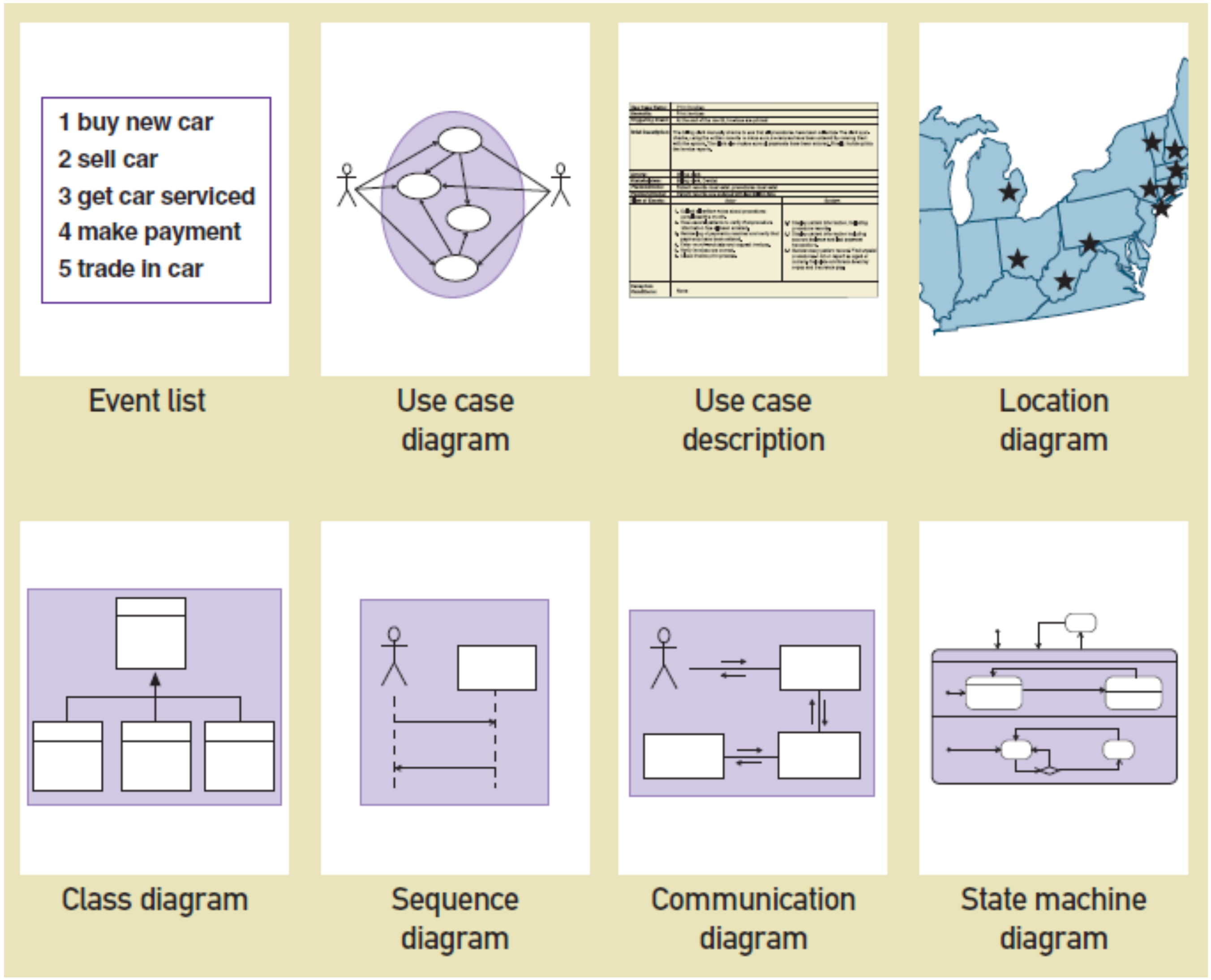
Reasons for Modeling

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

Overview of Models Used in Analysis and Design

- Analysis activities named “define system requirements”
  - Logical models
  - Provide detail without regard to specific technology (perfect technology assumption)
- Design models
  - Physical models
  - Provide technical details (non-perfect technology assumption)
  - Extend logical models

Some Analysis and Design Models



Stakeholders – The source of system requirements

Who do you involve and talk to?

- People with interest in successful system implementation
- Three primary groups of stakeholders
  - Users (use system)
  - Clients (pay for and own system)
  - Technical staff (ensure system operation)
- Every type of stakeholder is identified by analyst - one of the most important first steps in determining systems requirements
- The second task is to identify the critical person from each stakeholder type to be available as the business expert.

Users as Stakeholders

- Horizontal users (i.e., across departments)
- Vertical users or hierarchy within a department (i.e., clerical staff, middle management, and senior executives)
  - Business users – perform day-to-day operations (transactions): provide information about daily operations and how system supports them
  - Information users - who need current information
  - Management users – who need summary information
  - Executive users – who need strategic information
  - External users may have access to system (e.g., via Internet)

Clients and Technical Staff as Stakeholders

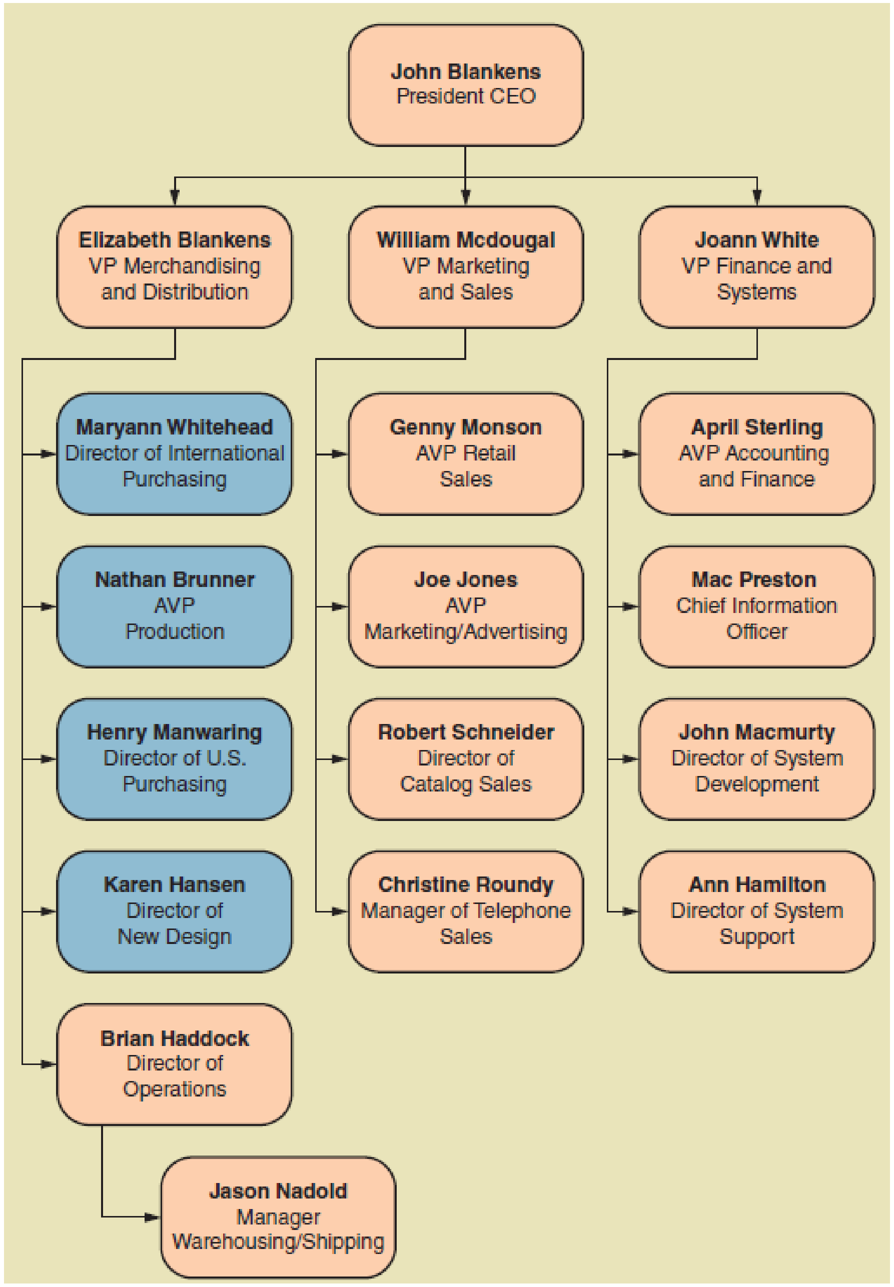
Client Stakeholders

- They pay for the project so they are important!
- Project team must provide project status reviews to the clients

Technical Stakeholders

- The technical staff includes people who establish and maintain the computing environment of the organization
- They are source of many technical requirements – provide guidance in such areas as programming language, computer platform, equipment, *etc*.

Example: RMO Internal Stakeholders



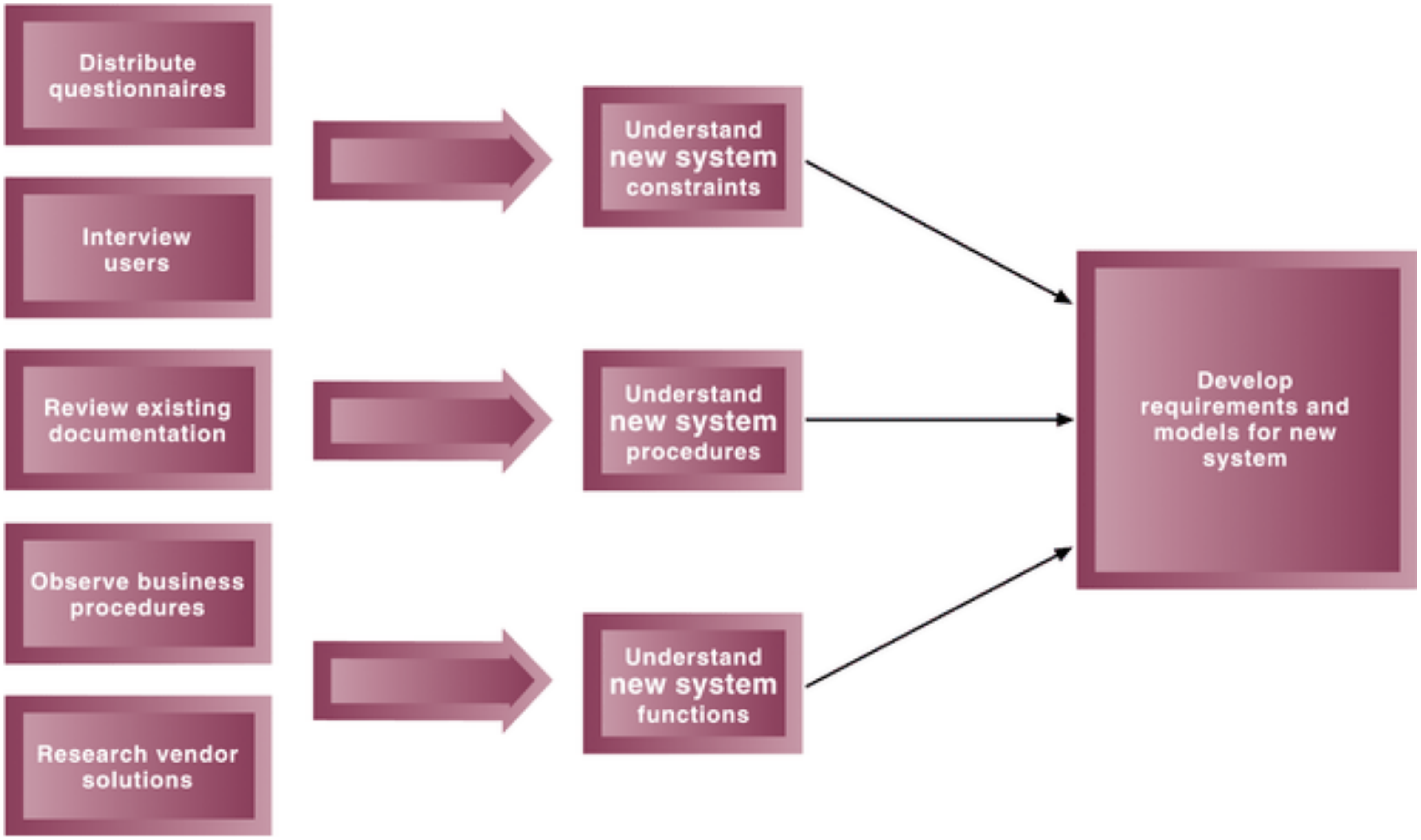
Identifying System Requirements

- Main Objective of Analysis Phase
  - To understand the business functions and develop system requirements
  - Question of studying existing systems first or not
  - Using structured approach, analyst first documents the existing system then extrapolates the requirements of the new system
  - Approach
    1. Develop <u>current</u> system physical models
    2. Extract the <u>current</u> system logical models
    3. Develop <u>new</u> system logical model
    4. Develop <u>new</u> system physical model

Information Gathering Techniques

- Interviewing users and other stakeholders
- Distributing and collecting questionnaires
- Reviewing inputs, outputs, and documentation
- Observing and documenting business procedures
- Researching vendor solutions
- Collecting active user comments and suggestions

The transition from information gathering to model building



Distribute questionnaires
Interview users
Review existing documentation
Observe business procedures
Research vendor solutions
Understand new system constraints
Understand new system procedures
Understand new system functions
Develop requirements and models for new system

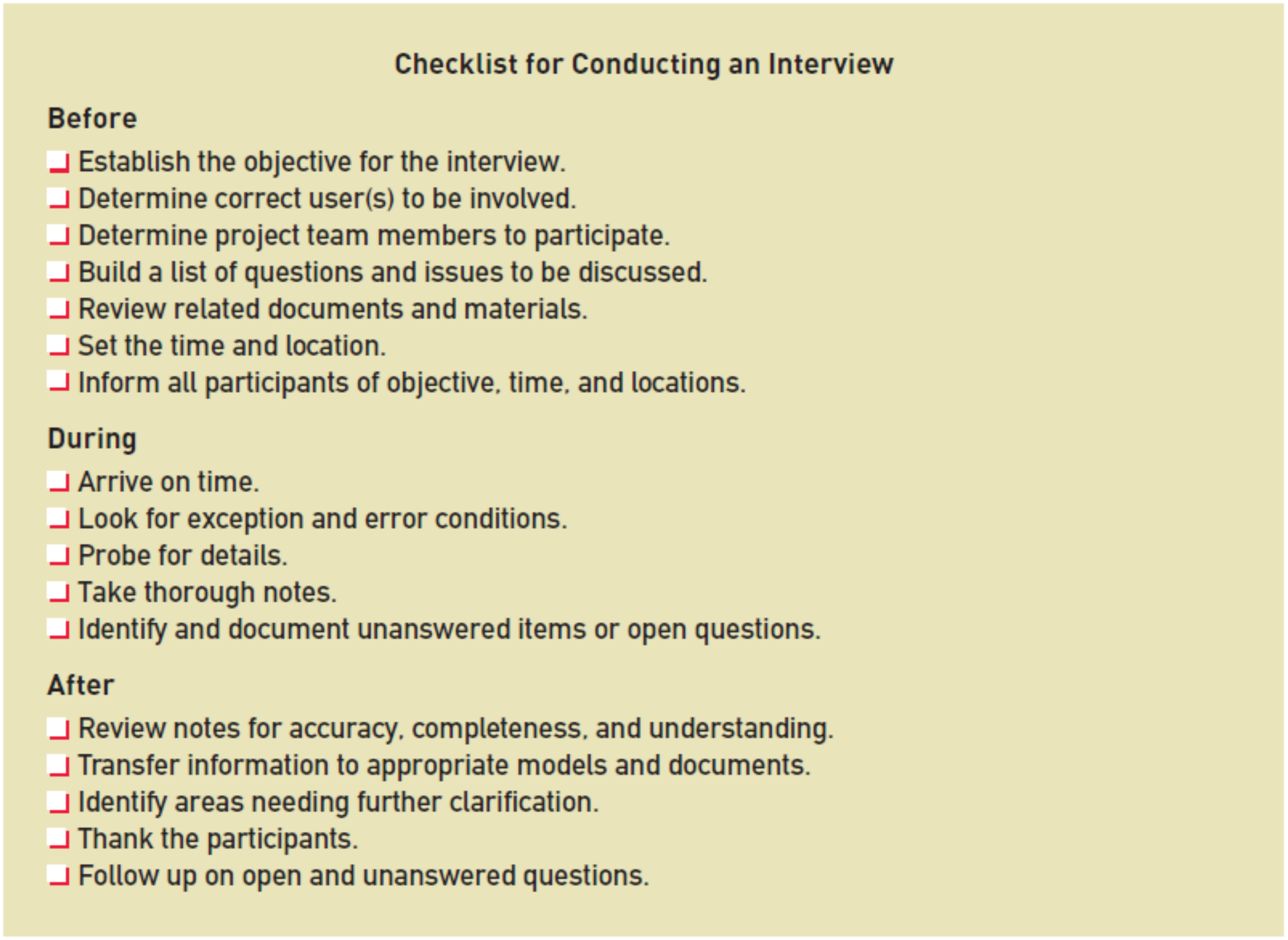
Interviewing Users and Other Stakeholders

- Prepare detailed questions
- Meet with individuals or groups of users
- Obtain and discuss answers to the questions
- Document the answers
- Follow up as needed in future meetings or interviews

Themes for Information Gathering Questions

Theme	Questions to users
What are the business operations and processes?	What do you do?
How should those operations be performed?	How do you do it? What steps do you follow? How could they be done differently?
What information is needed to perform those operations?	What information do you use? What inputs do you use? What outputs do you produce?

Preparing for Interview



Checklist for Conducting an Interview

Prepare

Before

- ☐ Establish the objective for the interview.
- ☐ Determine correct user(s) to be involved.
- ☐ Determine project team members to participate.
- ☐ Build a list of questions and issues to be discussed.
- ☐ Review related documents and materials.
- ☐ Set the time and location.
- ☐ Inform all participants of objective, time, and locations.

Carry out

During

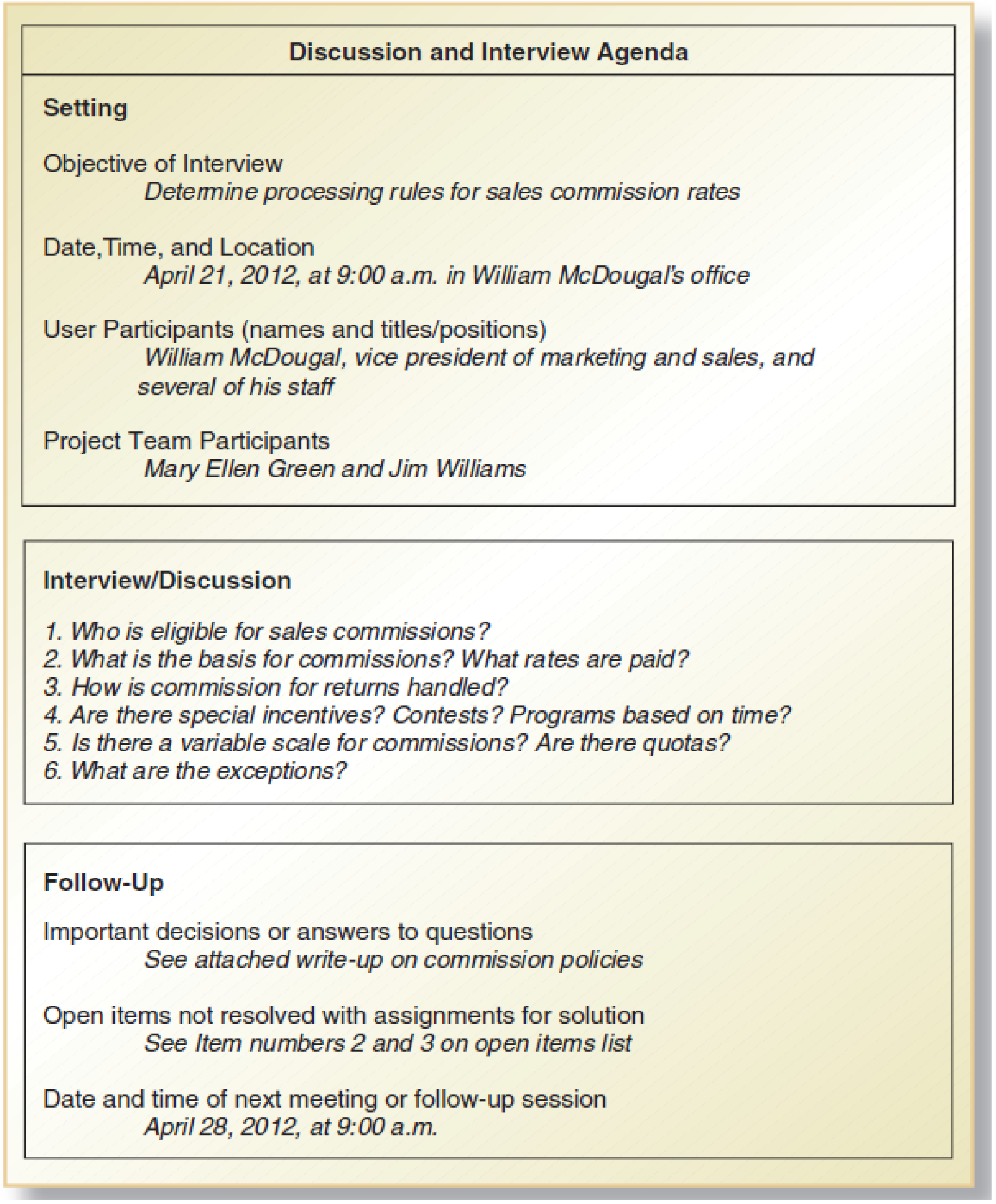
- ☐ Arrive on time.
- ☐ Look for exception and error conditions.
- ☐ Probe for details.
- ☐ Take thorough notes.
- ☐ Identify and document unanswered items or open questions.

Follow up

After

- ☐ Review notes for accuracy, completeness, and understanding.
- ☐ Transfer information to appropriate models and documents.
- ☐ Identify areas needing further clarification.
- ☐ Thank the participants.
- ☐ Follow up on open and unanswered questions.

Interview Session Agenda



Discussion and Interview Agenda

Setting

Objective of Interview
Determine processing rules for sales commission rates

Date,Time, and Location
April 21, 2012, at 9:00 a.m. in William McDougal's office

User Participants (names and titles/positions)
William McDougal, vice president of marketing and sales, and several of his staff

Project Team Participants
Mary Ellen Green and Jim Williams

Interview/Discussion

1. *Who is eligible for sales commissions?*
2. *What is the basis for commissions? What rates are paid?*
3. *How is commission for returns handled?*
4. *Are there special incentives? Contests? Programs based on time?*
5. *Is there a variable scale for commissions? Are there quotas?*
6. *What are the exceptions?*

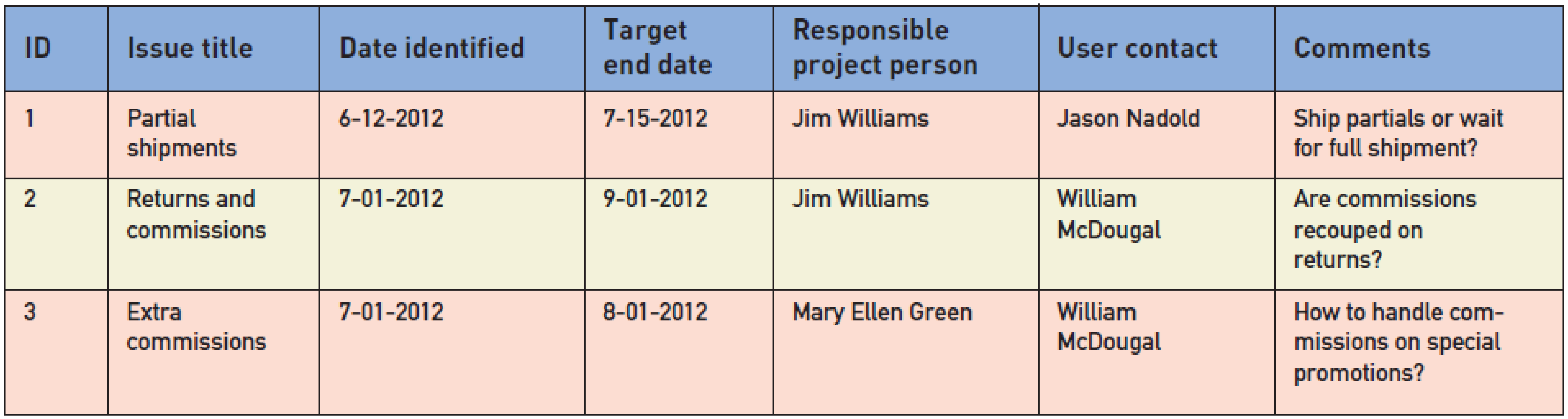
Follow-Up

Important decisions or answers to questions
See attached write-up on commission policies

Open items not resolved with assignments for solution
See Item numbers 2 and 3 on open items list

Date and time of next meeting or follow-up session
April 28, 2012, at 9:00 a.m.

Keeping an Open Items List



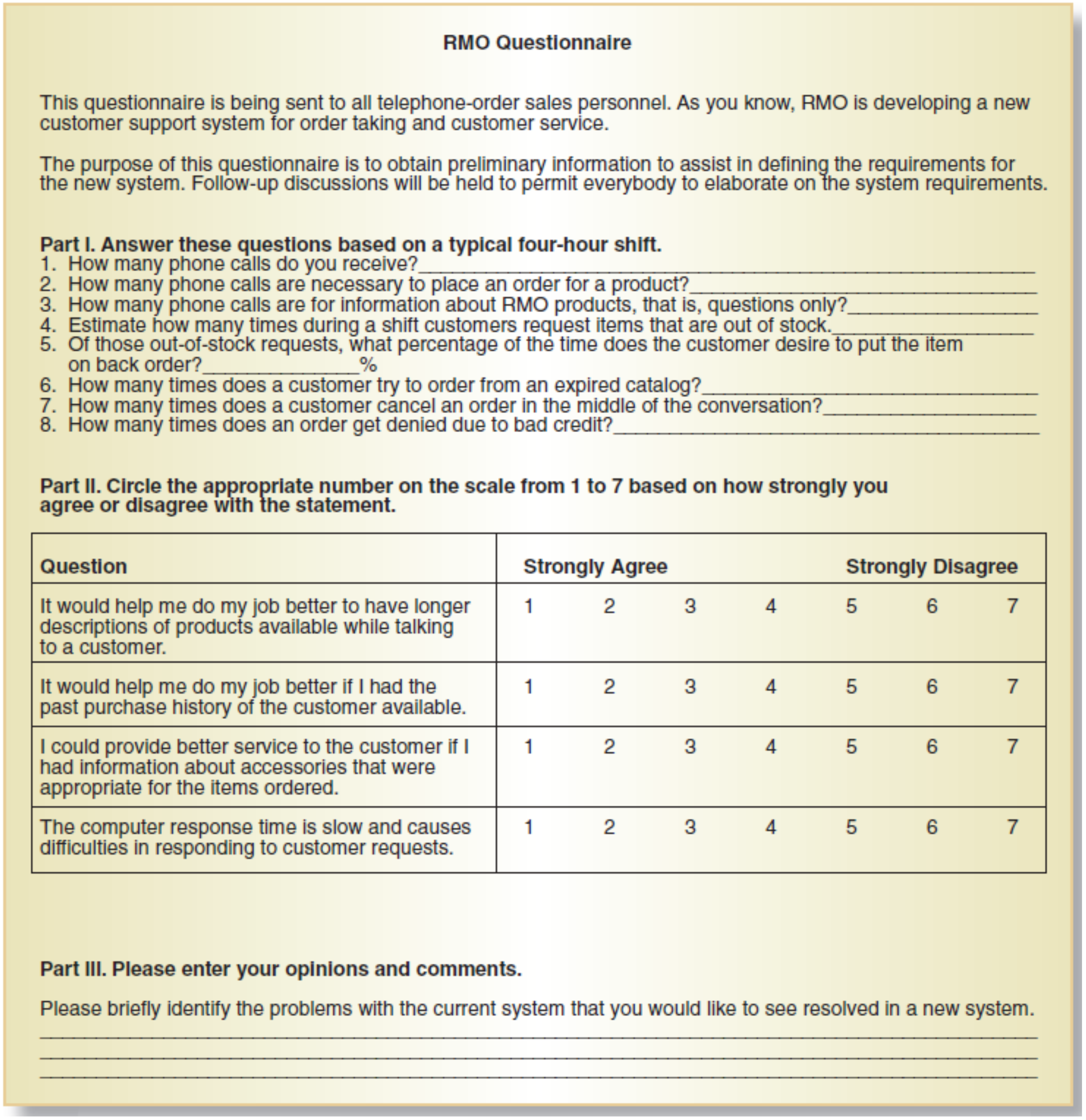
ID	Issue title	Date identified	Target end date	Responsible project person	User contact	Comments
1	Partial shipments	6-12-2012	7-15-2012	Jim Williams	Jason Nadold	Ship partials or wait for full shipment?
2	Returns and commissions	7-01-2012	9-01-2012	Jim Williams	William McDougal	Are commissions recouped on returns?
3	Extra commissions	7-01-2012	8-01-2012	Mary Ellen Green	William McDougal	How to handle com-missions on special promotions?

Sample Questionnaire

Quantitative

Closed-ended

Open-ended



RMO Questionnaire

This questionnaire is being sent to all telephone-order sales personnel. As you know, RMO is developing a new customer support system for order taking and customer service.

The purpose of this questionnaire is to obtain preliminary information to assist in defining the requirements for the new system. Follow-up discussions will be held to permit everybody to elaborate on the system requirements.

Part I. Answer these questions based on a typical four-hour shift.

1. How many phone calls do you receive?____________________
2. How many phone calls are necessary to place an order for a product?____________________
3. How many phone calls are for information about RMO products, that is, questions only?________________
4. Estimate how many times during a shift customers request items that are out of stock.________________
5. Of those out-of-stock requests, what percentage of the time does the customer desire to put the item on back order?______________%
6. How many times does a customer try to order from an expired catalog?____________________
7. How many times does a customer cancel an order in the middle of the conversation?________________
8. How many times does an order get denied due to bad credit?____________________

Part II. Circle the appropriate number on the scale from 1 to 7 based on how strongly you agree or disagree with the statement.

Question	Strongly Agree						Strongly Disagree
It would help me do my job better to have longer descriptions of products available while talking to a customer.	1	2	3	4	5	6	7
It would help me do my job better if I had the past purchase history of the customer available.	1	2	3	4	5	6	7
I could provide better service to the customer if I had information about accessories that were appropriate for the items ordered.	1	2	3	4	5	6	7
The computer response time is slow and causes difficulties in responding to customer requests.	1	2	3	4	5	6	7

Part III. Please enter your opinions and comments.

Please briefly identify the problems with the current system that you would like to see resolved in a new system.

____
____
____

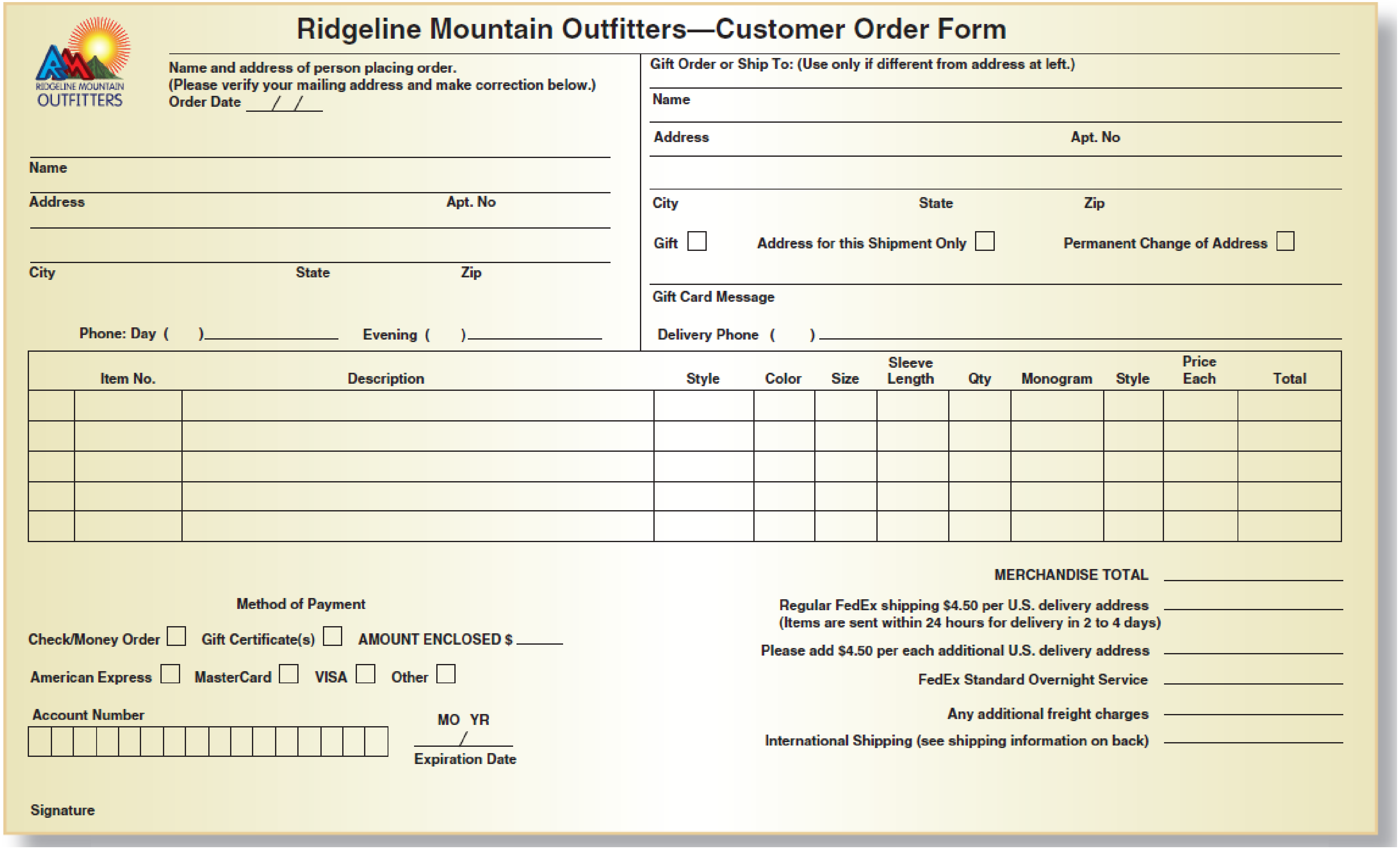
Limitations of Questionnaires

- Not well suited for learning about processes, workflows or techniques (e.g. to answer "How do you do this process?" - better to interview or observe)
- Questions that encourage elaboration are called "open-ended" – can be put in a questionnaire but if there are too many of these, questionnaires tend not to get returned!
- Relies in user's memory of their use of systems (researches show this differs from what they actually do in many cases!)

Review Existing Reports, Forms and Procedure Descriptions

- Review of existing documents very useful
  - Can help you get a preliminary understanding of processes involved in a company
  - Can be used in conjunction with interviews
    - Can be used to develop interview questions
    - Can be used in interviews themselves
      - As visual aids
      - Working documents for discussion
  - Review of forms helps to identify business rules
  - Helps to discover discrepancies and redundancies

Review Inputs, Outputs, and Procedures



RIDGELINE MOUNTAIN OUTFITTERS

Ridgeline Mountain Outfitters—Customer Order Form

Name and address of person placing order.
(Please verify your mailing address and make correction below.)
Order Date ___/___/___

Name

Address Apt. No

City State Zip

Phone: Day ( ) ________ Evening ( ) ________

Gift Order or Ship To: (Use only if different from address at left.)

Name

Address Apt. No

City State Zip

Gift ☐ Address for this Shipment Only ☐ Permanent Change of Address ☐

Gift Card Message

Delivery Phone ( ) ________

	Item No.	Description	Style	Color	Size	Sleeve Length	Qty	Monogram	Style	Price Each	Total

Method of Payment

Check/Money Order ☐ Gift Certificate(s) ☐ AMOUNT ENCLOSED $ ____

American Express ☐ MasterCard ☐ VISA ☐ Other ☐

Account Number

MO YR
___/___
Expiration Date

Signature

MERCHANDISE TOTAL	
Regular FedEx shipping $4.50 per U.S. delivery address (Items are sent within 24 hours for delivery in 2 to 4 days)	
Please add $4.50 per each additional U.S. delivery address	
FedEx Standard Overnight Service	
Any additional freight charges	
International Shipping (see shipping information on back)	

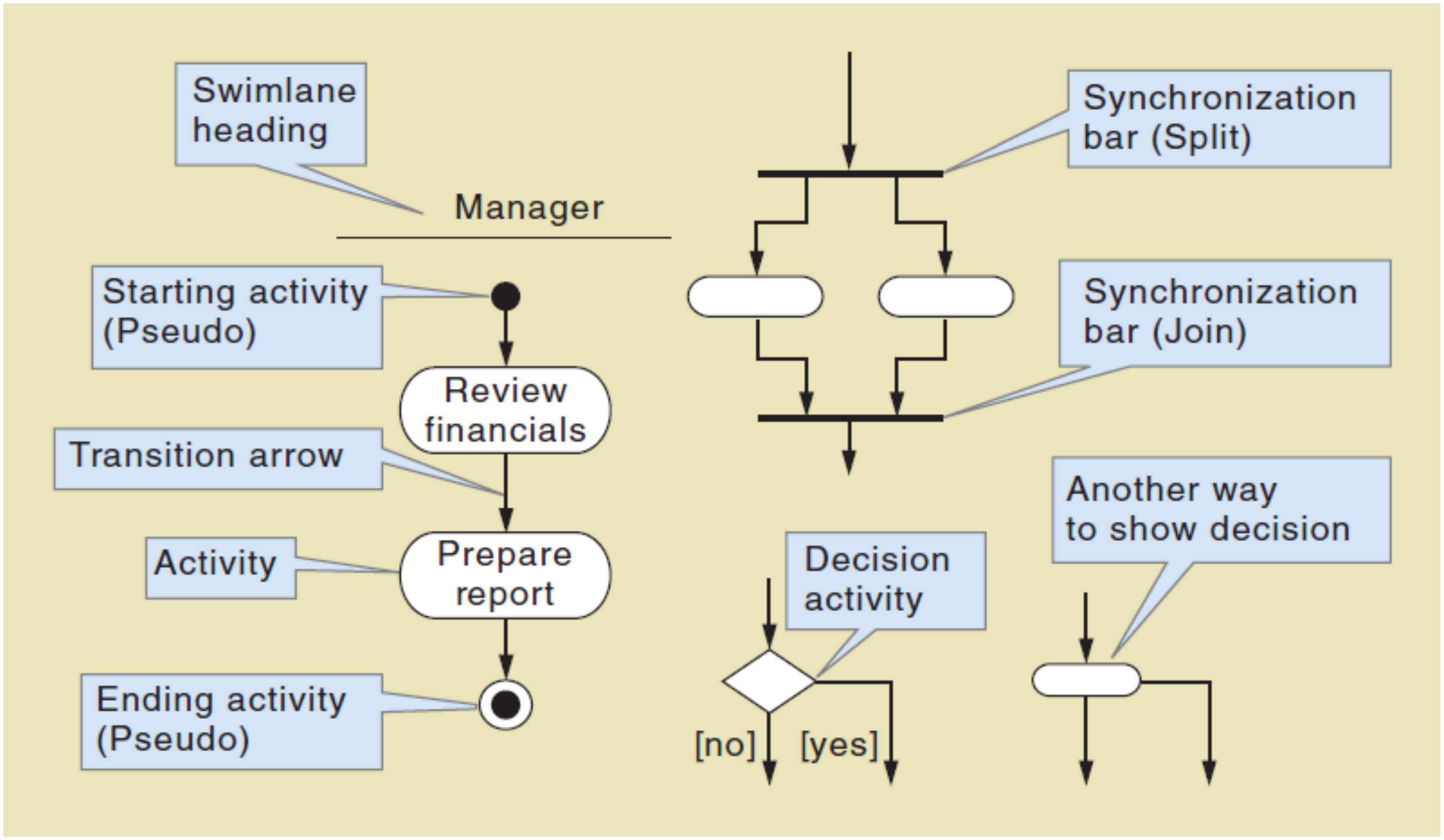
Additional Techniques

- Observe and Document Business Processes
  - Watch and learn
  - Document with Activity diagram
- Research Vendor Solutions
  - See what others have done for similar situations
  - White papers, vendor literature, competitors
- Collect Active User Comments and Suggestions
  - Feedback on models and tests
  - Users know it when the see it

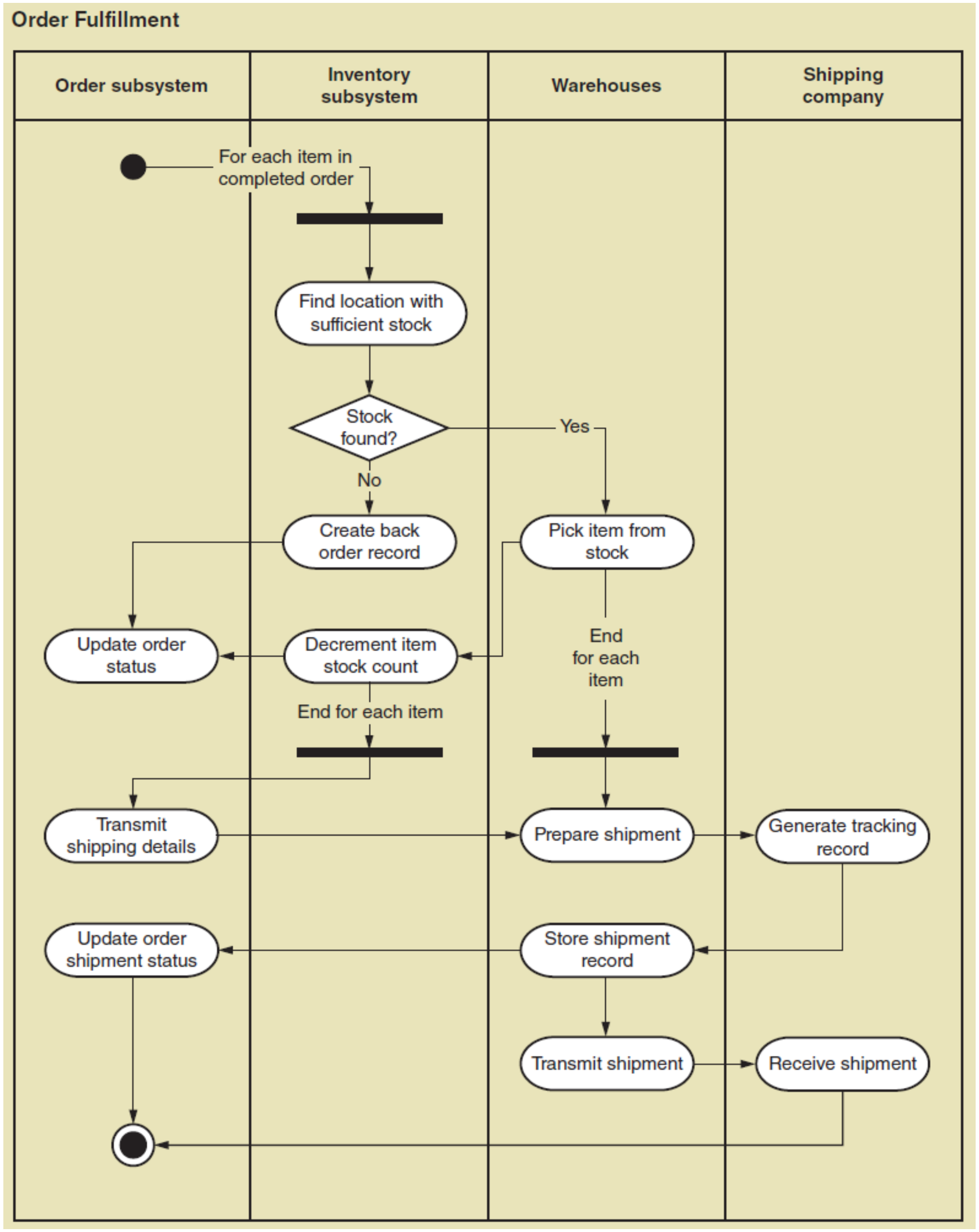
Documenting Workflows with Activity Diagrams

- Workflow– sequence of processing steps that completely handles one business transaction or customer request
- Activity Diagram– describes user (or system) activities, the person who does each activity, and the sequential flow of these activities
  - Useful for showing a graphical model of a workflow

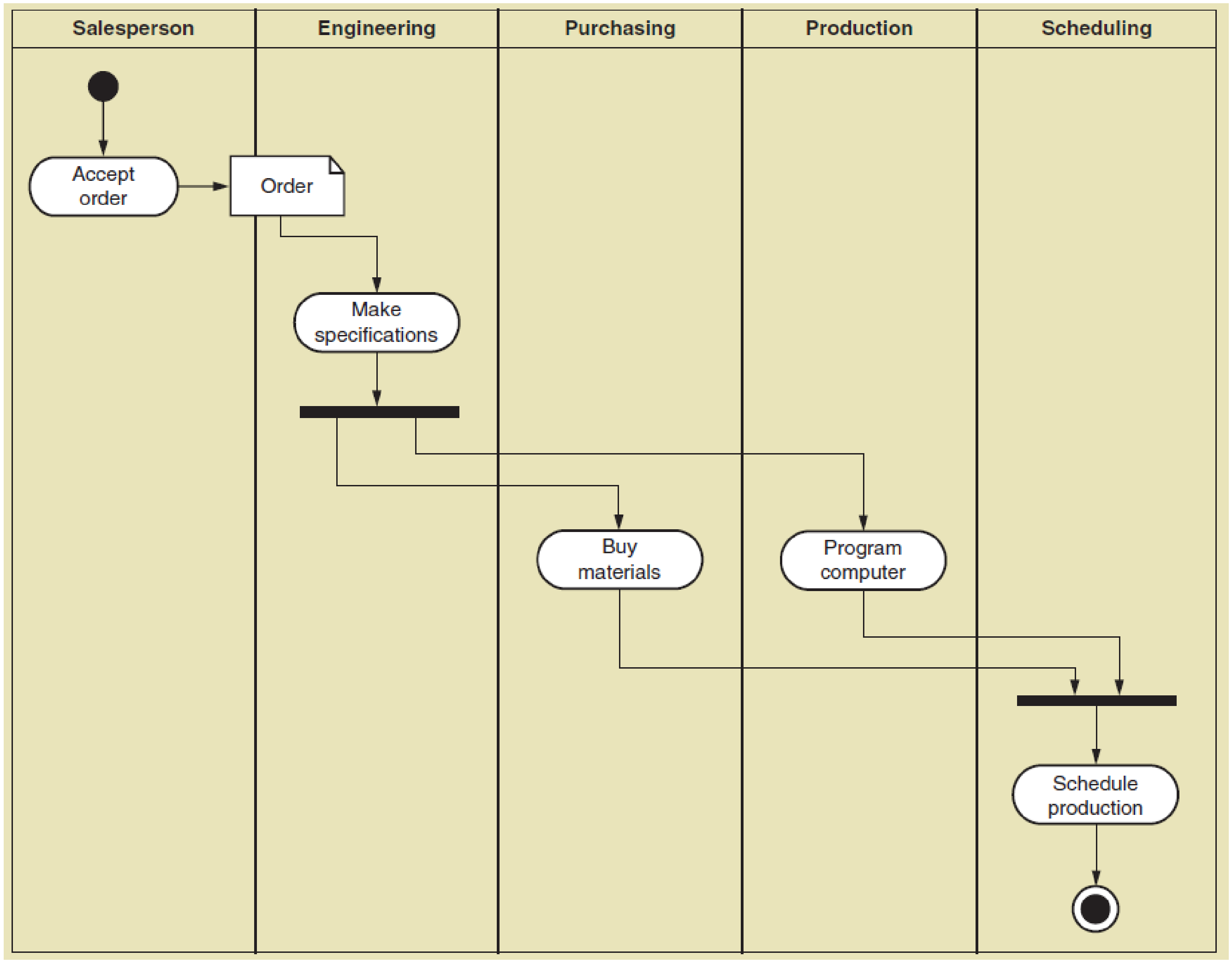
Activity Diagrams Symbols



Activity Diagram for RMO Order Fulfillment



Activity Diagram with Concurrent Paths



Building Prototypes

- Prototype: A preliminary working model of a larger system
- Uses:
  - To test feasibility
  - To help identify processing requirements
  - To compare different design and interface alternatives
- Different names
  - Throwaway prototypes
  - Discovery prototypes – used in analysis phase
  - Design prototypes – used in design
  - Evolving prototypes
    - Prototypes that grow and eventually may end up becoming the final system

Prototypes as tools during Analysis

- During analysis a discovery prototype
  - E.g. use of a prototype to determine screen formats and processing sequences (then thrown away)
  - Can show users or clients ideas early on to refine requirements (also used later on in design phase a lot)
- Characteristics of Prototypes
  - Operative: a prototype should be a working model, with some real functionality
  - Focused: a prototype should be focused on a single objective
  - Quick: the prototype should be built and modified quickly (so can validate an approach, and if it is wrong fix it fast in an iterative process)

Validating the Requirements

- Make sure gathered information is correct (fixing a requirements error later in SDLC can cost hundreds of times more than it would have cost to fix it during the requirements definition!)
- In software development, each project is unique, so each set of system requirements should be reviewed and tested as much as possible
- In programming we can proof the accuracy of the code using tests (i.e. by executing the program by entering appropriate input data and observing the resultant output. We cannot test the requirements that way
- In system analysis jargon it is called verify and validate (V&V) the system requirements
  - Verification means determining that the requirements are internally consistent (test whether the field definitions are consistent throughout all of the subsystems of a system)
  - Validation means ensuring that the requirements are complete and correctly express users' needs
- Structured walkthrough is effective means of implementing quality control early in project

Structured walkthrough

- Reviewing of the findings from your investigation
- Reviewing of the models based on those findings
- Objectives: to find errors, omissions and problems by documenting the requirements as the analysts understand them and then reviewing them with the project team
- Structured walkthroughs are not be used to discuss solutions for the errors found

Structured walkthrough

What and When

- First item to review is documentation that was developed as part of the analysis phase. It can be:
  - A narrative describing a process
  - A flow chart showing a workflow or model diagram documenting an entire procedure
- Better to conduct several smaller walkthroughs every week or two, than bigger ones with reviewing a number of documentation
- Very important to be scheduled as soon as documents have been created!

Structured walkthrough

Who

- Two main parties involved in walkthroughs:
  - Person (or persons) who need their work to be reviewed
  - Group who reviews it

How

- Requires the same steps as an interview (i.e., preparation, execution and follow-up)

Execution

– During the walkthrough analyst presents material point by point
– Walks through each diagram or section of a document explaining each component (one effective technique is to define a sample test case and process it through the defined flow)
– The reviewers look for inconsistencies or problems and point them out
– A librarian (helper for presenter) documents the comments, errors and suggestions
– Corrections and solutions are not made during the walkthrough
– If there is a complex error may reschedule for another meeting
– Reviewer should only provide focused feedback
– Presenter can integrate feedback later on when gets entire set of comments

Follow-up

– Making required corrections
– Additional walkthrough may be needed

Modeling System Requirements

- Use Cases and Events
  - Identify and Validate the Use Cases
  - Use Case Descriptions
  - Use case diagrams
- “Things” in the Problem Domain
  - The association among things
  - The Domain Model Class Diagram
  - The Entities and Entity-Relationship Diagram (ERD)

Use Cases

- Use case— an activity that the system performs, usually in response to a request by a user
- Use cases define functional requirements
- Analysts decompose the system into a set of use cases (functional decomposition)
  - Name each use case using Verb-Noun
- Two techniques for Identifying use cases
  - User goal technique
  - Event decomposition technique
- Validate and refine use cases
  - CRUD technique

Identifying Use Cases with the User Goal Technique

User	User goal and resulting use case
Potential customer	Search for item Fill shopping cart View product rating and comments
Marketing manager	Add/update product information Add/update promotion Produce sales history report
Shipping personnel	Ship items Track shipment Create item return

RMO Consolidated Sales and Marketing System (CSMS)

Event Decomposition Technique

- Event– something that occurs at a specific time and place, can be described, and should be remembered by the system
- Event Decomposition Technique - More Comprehensive and Complete Technique
    - Identify the events that occur to which the system must respond.
    - For each event, name a use case (verb-noun) that describes what the system does when the event occurs

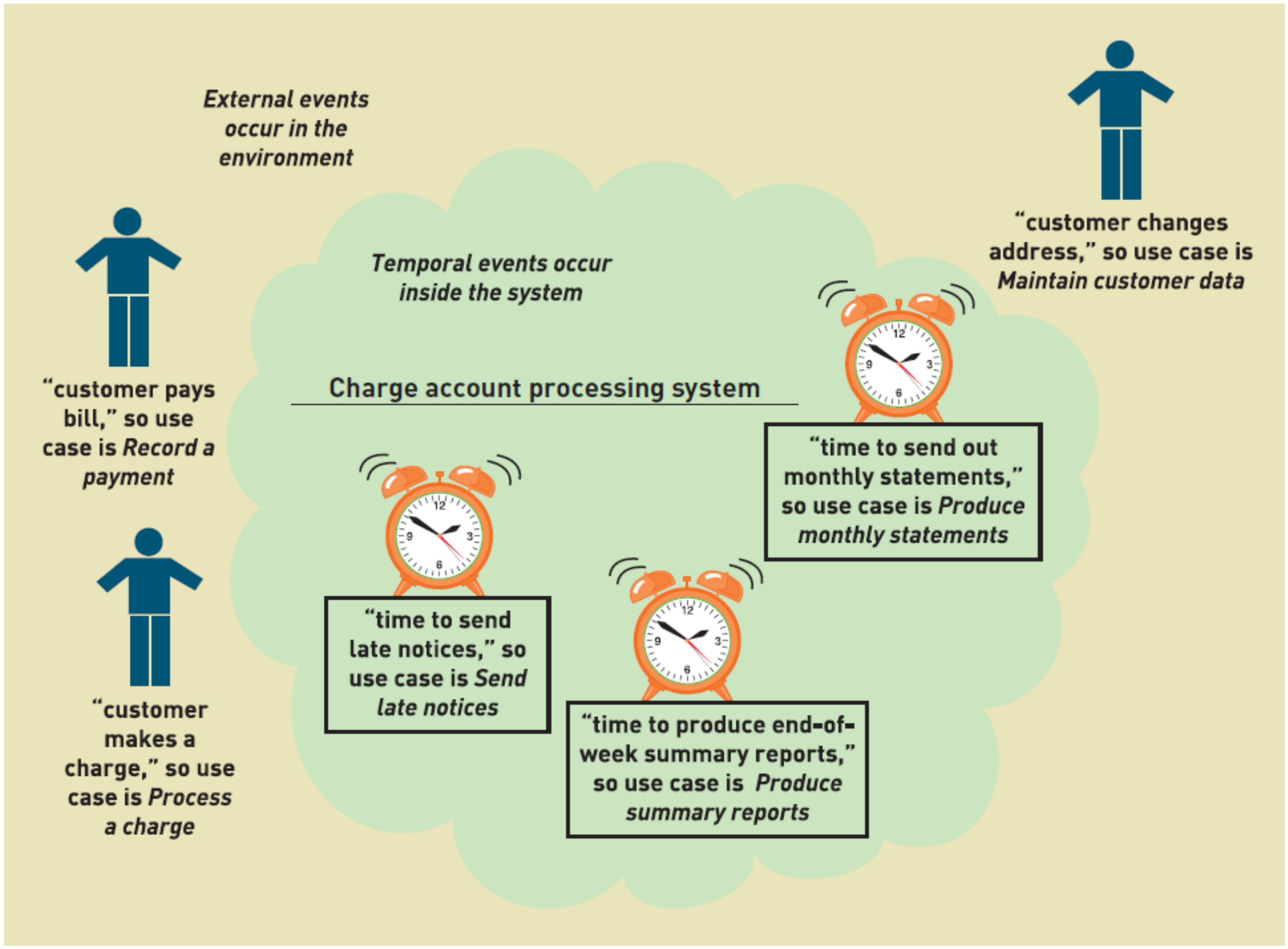
Event Decomposition Technique

- Help decompose at the EBP level of analysis:
  - elementary business process (EBP) is a fundamental business process performed by one person, in one place, in response to a business event
  - E.g. Is it a proper event?
    - “typing in a customer name”
    - “working with customer all day”
    - “adding a new customer”

Types of Events

- External Event
  - Occurs outside the system,
  - Initiated by an external agent or actor (e.g. “customer places an order”, and “manager requests for information”)
- Temporal Event
  - Occurs as a result of reaching a point in time
  - Based on deadlines (e.g. “time to produce biweekly payroll”)
- State (Internal) Event
  - Occurs when something happens inside the system that triggers some process (e.g. “reorder point is reached for inventory item”)

Events and Use Cases in a Charge Account Processing System


External events occur in the environment
Temporal events occur inside the system
Charge account processing system
"customer pays bill," so use case is *Record a payment*
"customer makes a charge," so use case is *Process a charge*
"customer changes address," so use case is *Maintain customer data*
"time to send out monthly statements," so use case is *Produce monthly statements*
"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*

External Event Checklist

- External agent or actor wants something resulting in a transaction
  - Customer buys a product
- External agent or actor wants some information
  - Customer wants to know product details
- External data changed and needs to be updated
  - Customer has new address and phone
- Management wants some information
  - Sales manager wants update on production plans

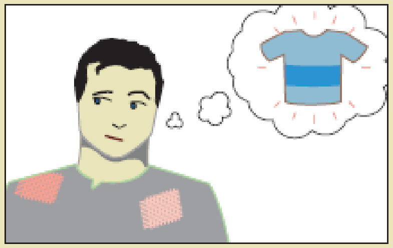
Temporal Event Checklist

- Internal outputs needed at points in time
  - Management reports (summary or exception)
  - Operational reports (detailed transactions)
  - Internal statements and documents (including payroll)
- External outputs needed at points of time
  - Statements, status reports, bills, reminders

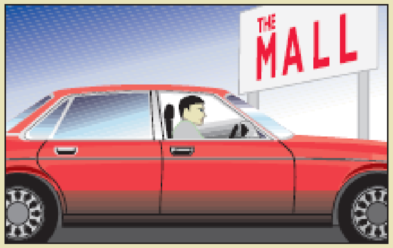
Identifying Events

- Events versus conditions and responses
  - Event vs. a sequence of prior conditions that leads up to the event – which event directly affects the system
  - External event vs. system's response
    - e.g. when customer buys the shirt, system requests credit card number, the customer supplies the credit card.
    - "the customer supplies the credit card " is not an event for the information system, but just a part of interaction that occurs while completing the original transaction, i.e. "the customer buying the shirt" which is the real event
    - To determine: ask whether any long pauses or intervals occur (i.e., can the system transaction be completed without interruption? Or, is the system at rest again waiting for another transaction?)

Finding the actual event that affects the system

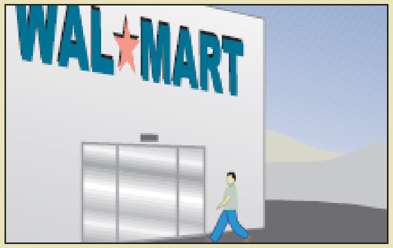


Customer thinks about getting a new shirt

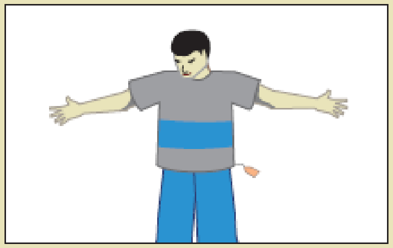


Customer drives to the mall

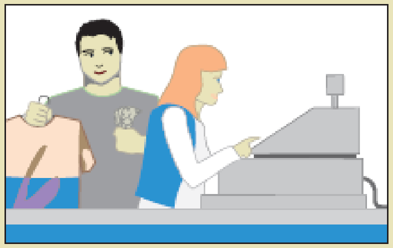
Customer tries on a shirt at Sears



Customer goes to Walmart



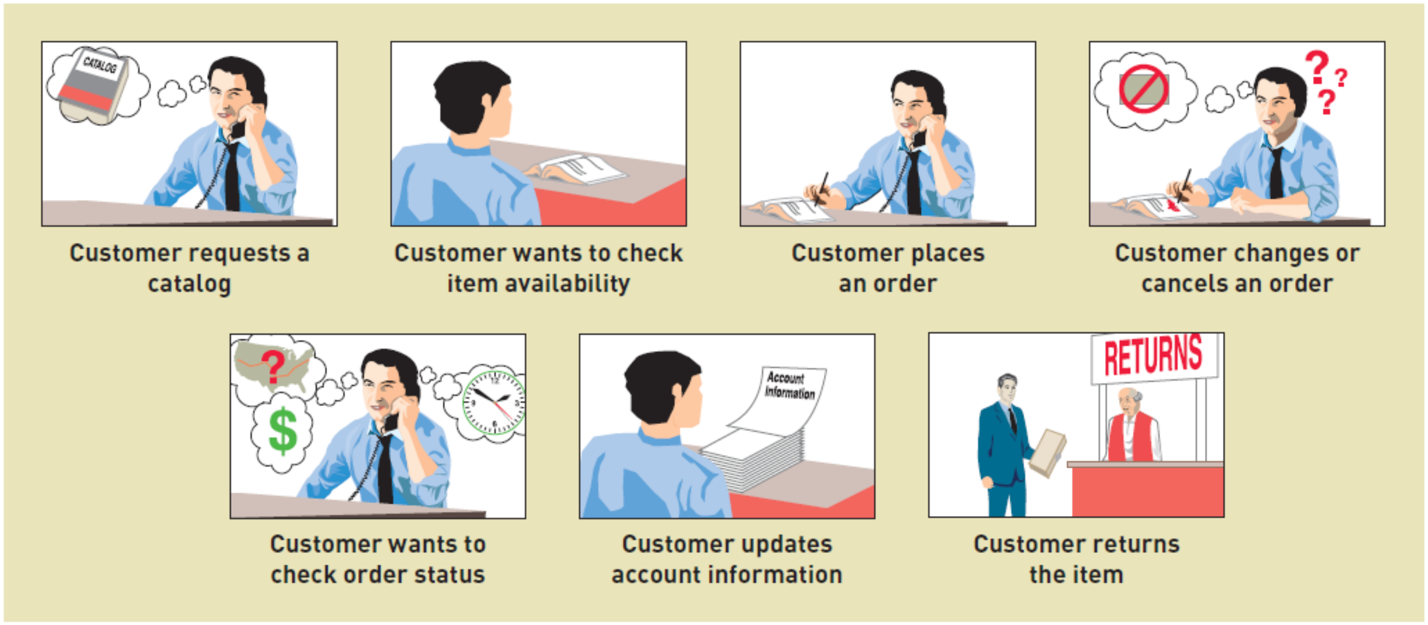
Customer tries on a shirt at Walmart



Customer buys a shirt (*the event that directly affects the system!*)

Identifying Events

- The Sequence of Events: Tracing a Transaction's Life Cycle
  - Useful to trace the sequence of events that might occur for a specific external agent or actor
  - E.g. all the possible transactions that might result from one new customer

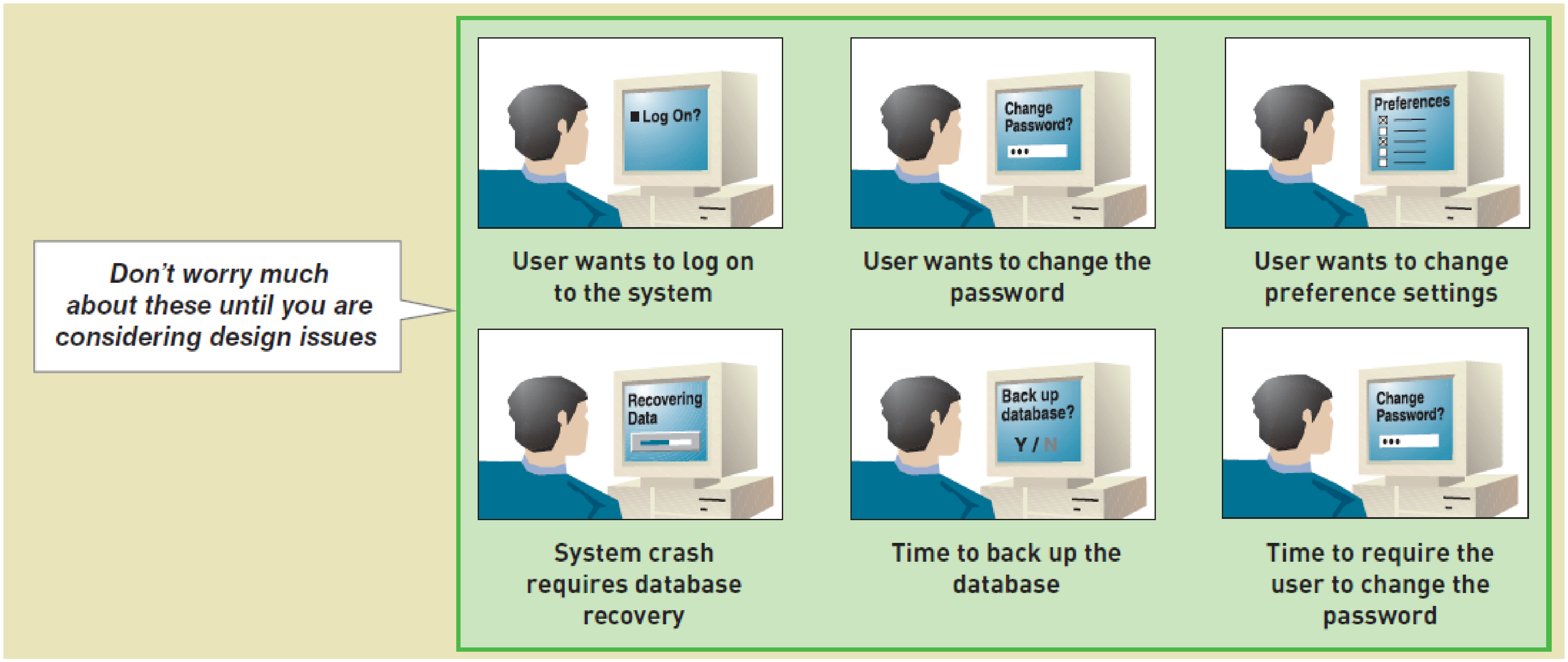

CATALOG
Customer requests a catalog
Customer wants to check item availability
Customer places an order
Customer changes or cancels an order
Customer wants to check order status
Account information
Customer updates account information
RETURNS
Customer returns the item

Identifying Events

- Technology-Dependent Events and System Controls
  - Events that are important to the system but do not directly concern users or transactions
  - Involve design choices or system controls – should be ignored in the analysis, but important for design
  - System controls: Checks or safety procedures put in place to protect the integrity of the system.
  - Perfect technology assumption
    - Decide which events apply to controls: events should be included during analysis only if the system would be required to respond under perfect condition (obviously not the case)

Perfect Technology Assumption

- Don't worry about functions built into system because of limits in technology and people. Wait until design.


Don't worry much about these until you are considering design issues
Log On?
User wants to log on to the system
Change Password?
User wants to change the password
Preferences
User wants to change preference settings
Recovering Data
System crash requires database recovery
Back up database? Y / N
Time to back up the database
Change Password?
Time to require the user to change the password

Event Decomposition Technique: Specific Steps

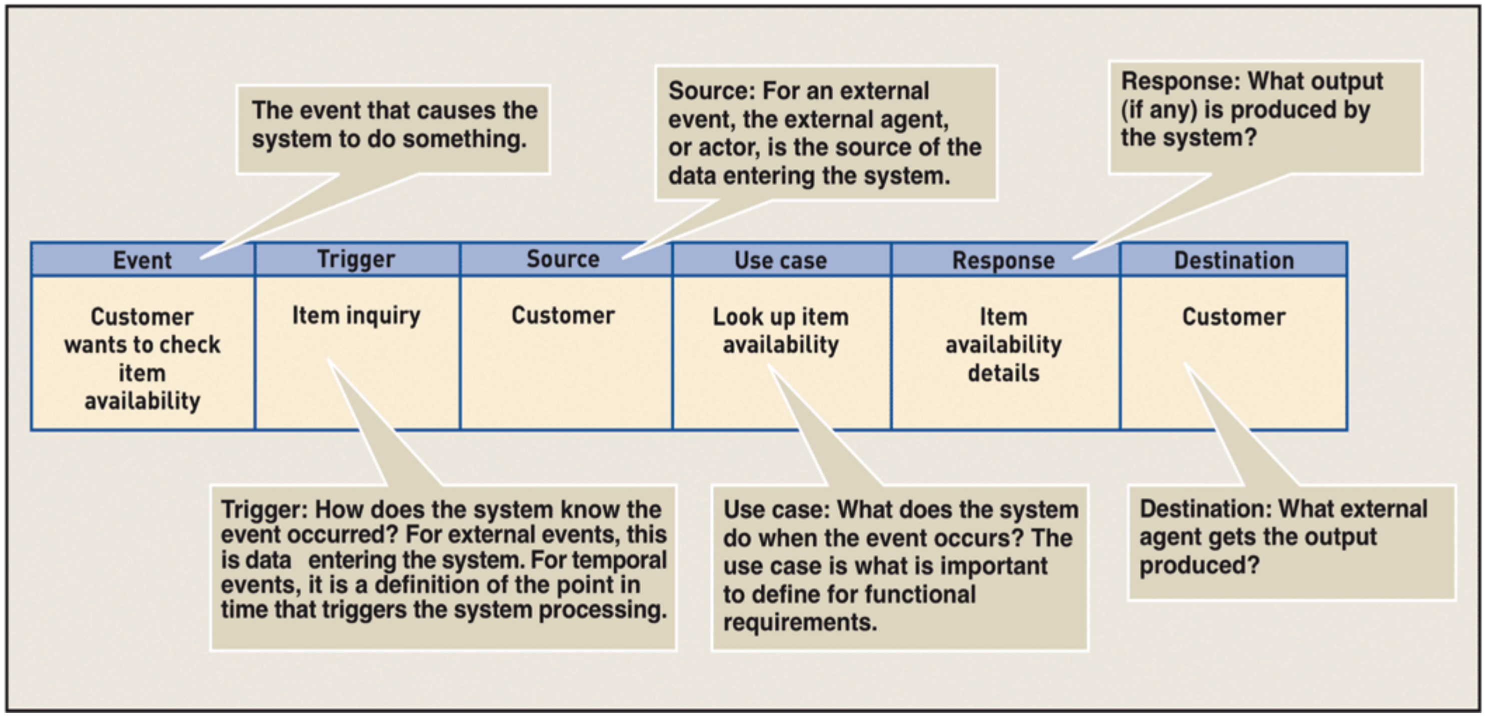
1. Consider the external events in the system environment that require a response from the system by using the checklist
2. For each external event, identify and name the use case that the system requires
3. Consider the temporal events that require a response from the system by using the checklist
4. For each temporal event, identify and name the use case that the system requires and then establish the point of time that will trigger the use case

Event Decomposition Technique: Specific Steps (continued)

5. Consider the state events that the system might respond to, particularly if it is a real-time system in which devices or internal state changes trigger use cases.
6. For each state event, identify and name the use case that the system requires and then define the state change.
7. When events and use cases are defined, check to see if they are required by using the perfect technology assumption. Do not include events that involve such system controls as login, logout, change password, and backup or restore the database, as these are put in later.

Documenting the events – event table

- Row contains information about one event
- Column represents a key piece of information about the event



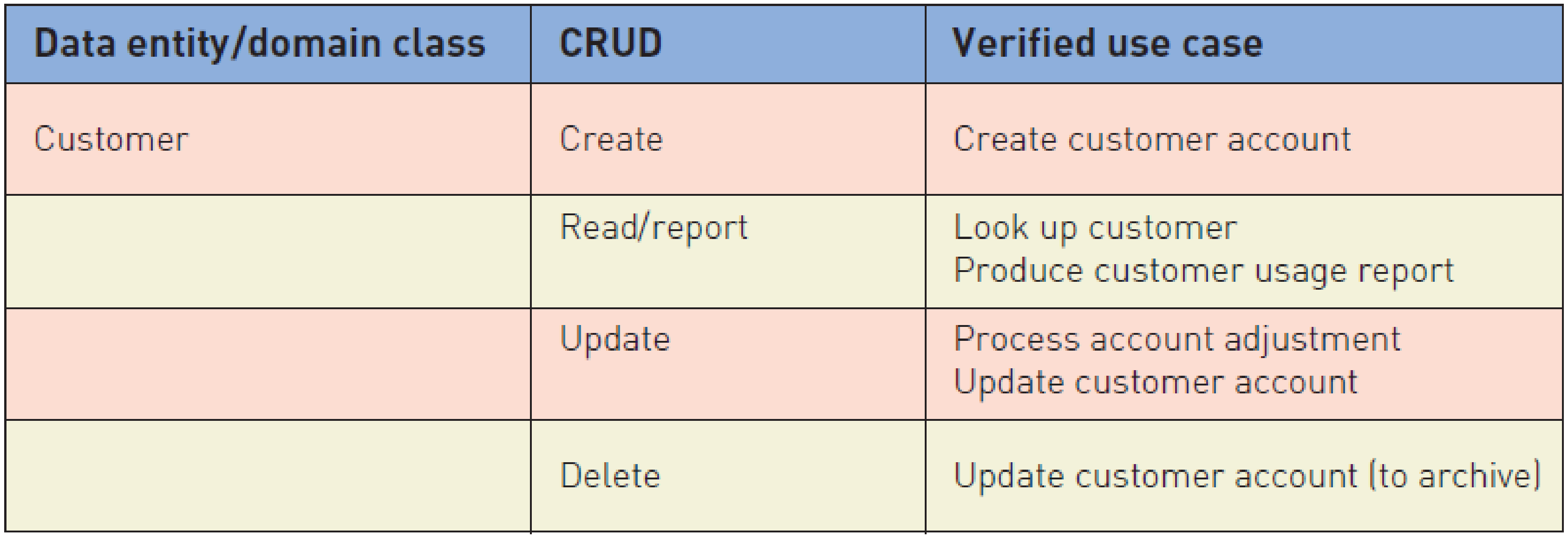
Event	Trigger	Source	Use case	Response	Destination
Customer wants to check item availability	Item inquiry	Customer	Look up item availability	Item availability details	Customer

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?

Use Cases and CRUD Technique

- CRUD: Create, Read, Update, and Delete
  - Often introduced in database context
- Technique to validate, refine or cross-check use cases
  - Looks at each type of data and verifies that use cases have been identified that crate the data, read or report on the data, update the data, and delete (or archive) the data
  - NOT for primarily identifying use cases

Verifying Use Cases with the CRUD Technique



Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

CRUD Technique Steps

1. Identify all the data entities or domain classes involved in the new system.
2. For each type of data (data entity or domain class), verify that a use case has been identified that creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.
3. If a needed use case has been overlooked, add a new use case and then identify the stakeholders.
4. With integrated applications, make sure it is clear which application is responsible for adding and maintaining the data and which system merely uses the data.

Use Case Descriptions

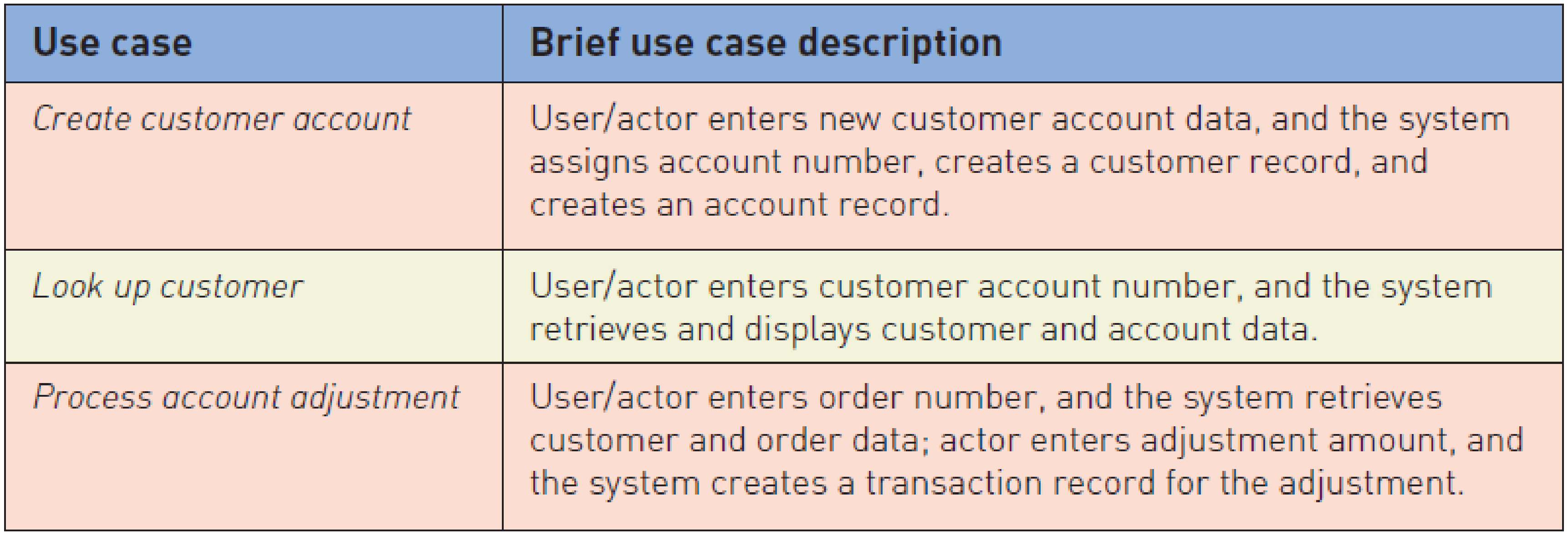
- Use case name
  - Verb-noun
- Scenario (if needed)
  - A use case can have more than one scenario (special case or more specific path)
- Triggering event
  - Based on event decomposition technique
- Brief description
  - Written previously when use case was identified
- Actors
  - One or more users from use case diagrams

Use Case Descriptions

- Related use cases <<includes>>
    - If one use case invokes or includes another
- Stakeholders
    - Anyone with an interest in the use case
- Preconditions
    - What must be true before the use case begins
- Post conditions
    - What must be true when the use case is completed
    - Use for planning test case expected results
- Flow of activities
    - The activities that go on between actor and the system
- Exception conditions
    - Where and what can go wrong

Brief Use Case Descriptions

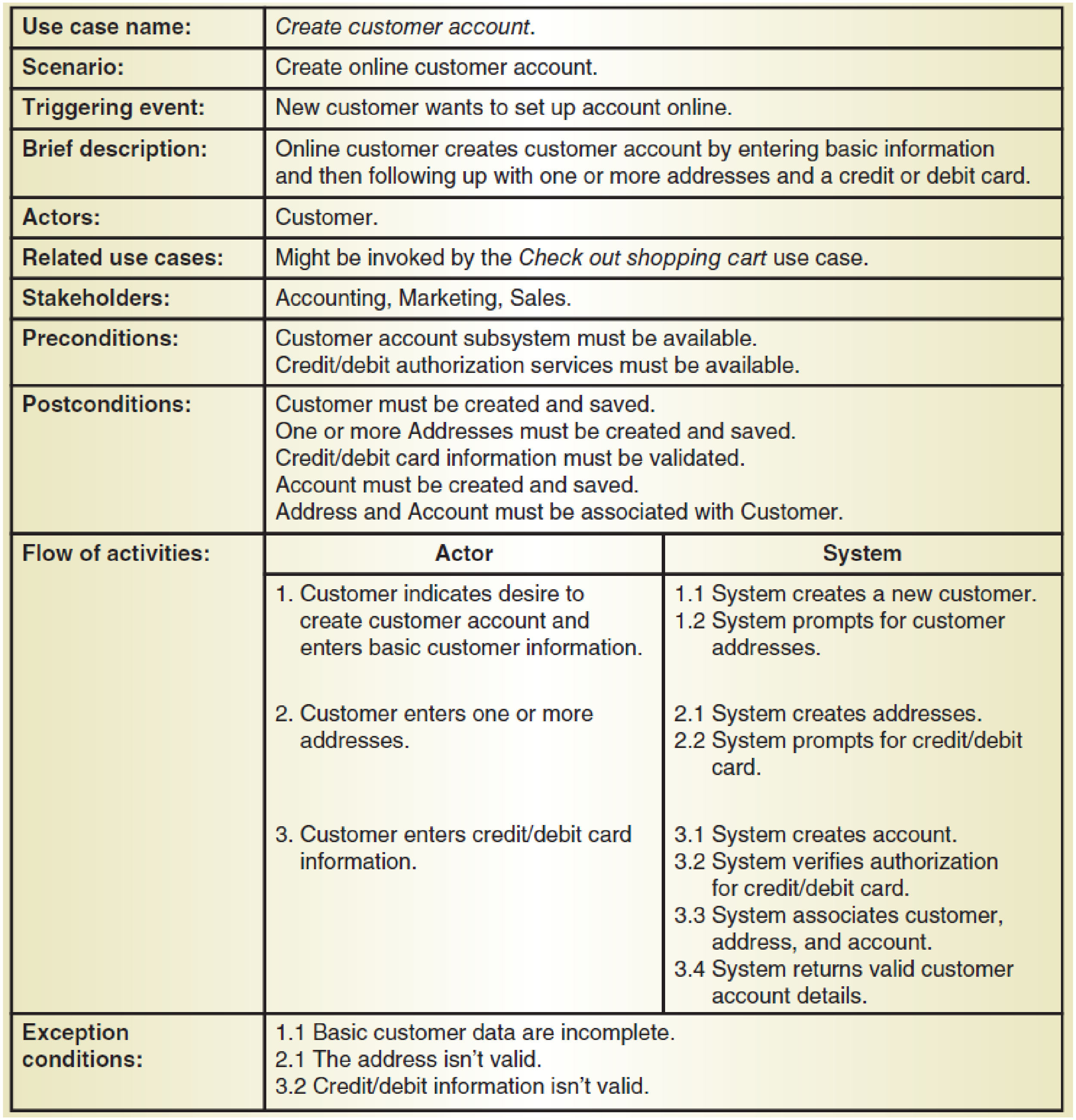
- Brief use case description is often a one sentence description showing the main steps in a use case



Use case	Brief use case description
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer account number, and the system retrieves and displays customer and account data.
Process account adjustment	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

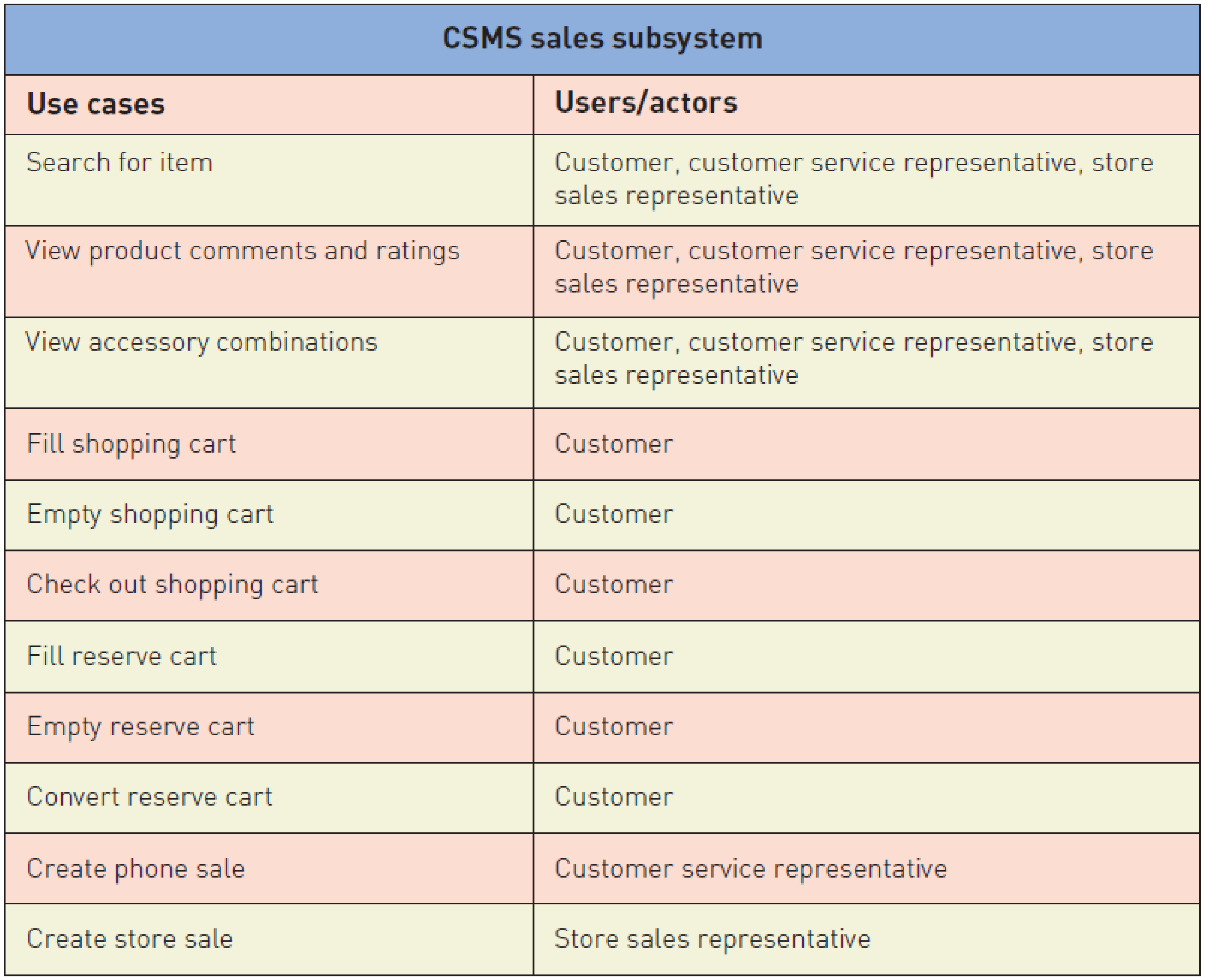
Fully Developed Use Case Description

Use case: *Create customer account*



Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

A List of Use Cases: RMO CSMS Project

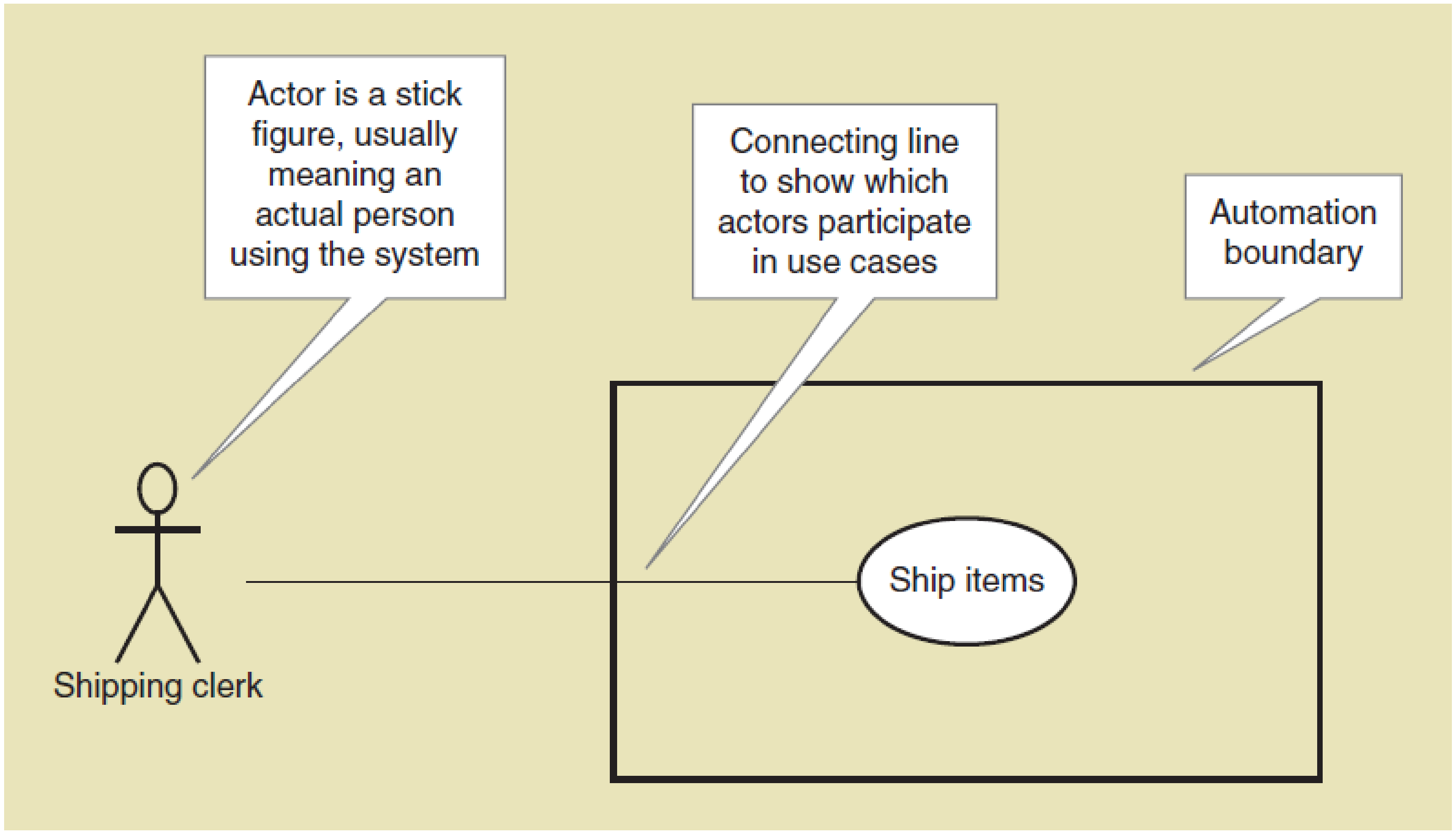


CSMS sales subsystem	
Use cases	**Users/actors**
Search for item	Customer, customer service representative, store sales representative
View product comments and ratings	Customer, customer service representative, store sales representative
View accessory combinations	Customer, customer service representative, store sales representative
Fill shopping cart	Customer
Empty shopping cart	Customer
Check out shopping cart	Customer
Fill reserve cart	Customer
Empty reserve cart	Customer
Convert reserve cart	Customer
Create phone sale	Customer service representative
Create store sale	Store sales representative

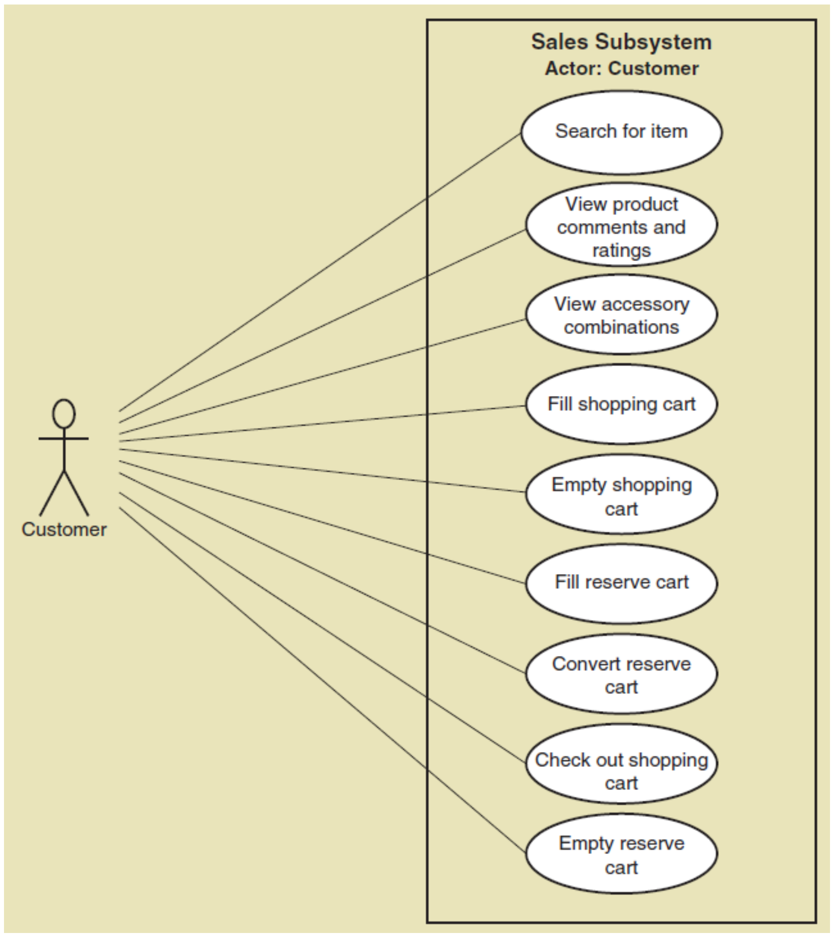
Use Case Diagrams

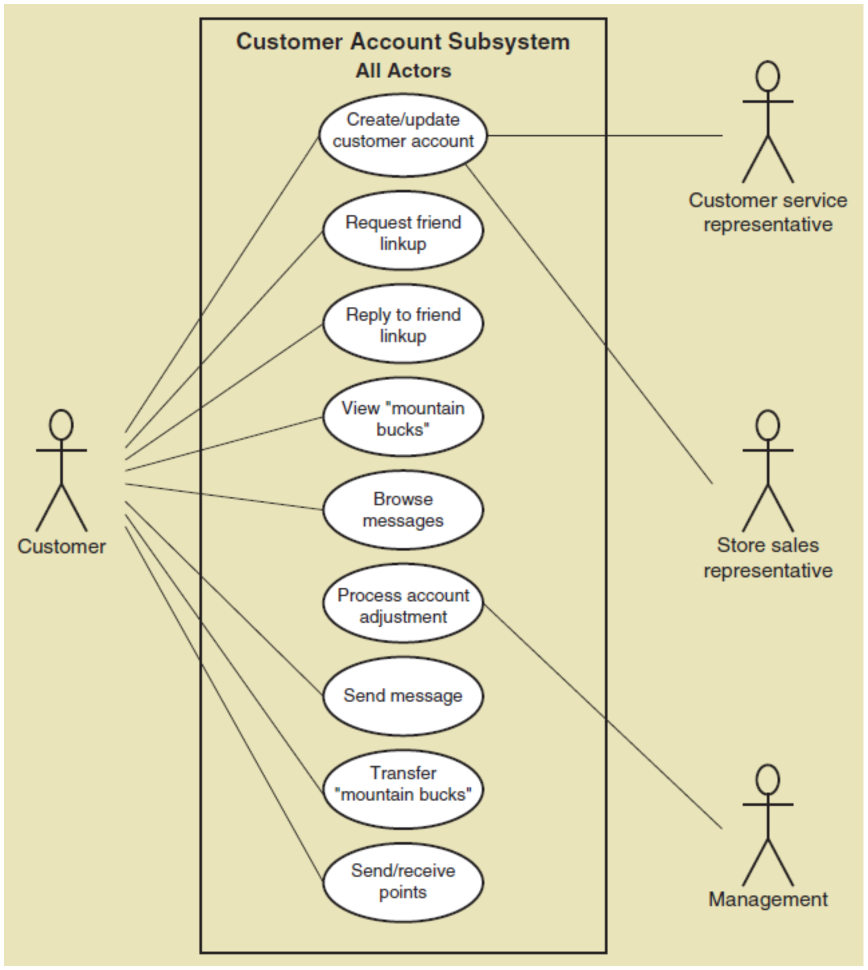
- Use case diagram— a UML model used to graphically show uses cases and their relationships to actors
- UML is Unified Modeling Language, the standard for diagrams and terminology for developing information systems
- Actor is the UML name for a end user
- Automation boundary— the boundary between the computerized portion of the application and the users who operate the application

Use Case Diagrams Symbols


Actor is a stick figure, usually meaning an actual person using the system
Connecting line to show which actors participate in use cases
Automation boundary
Shipping clerk
Ship items

Use Case Diagram Examples

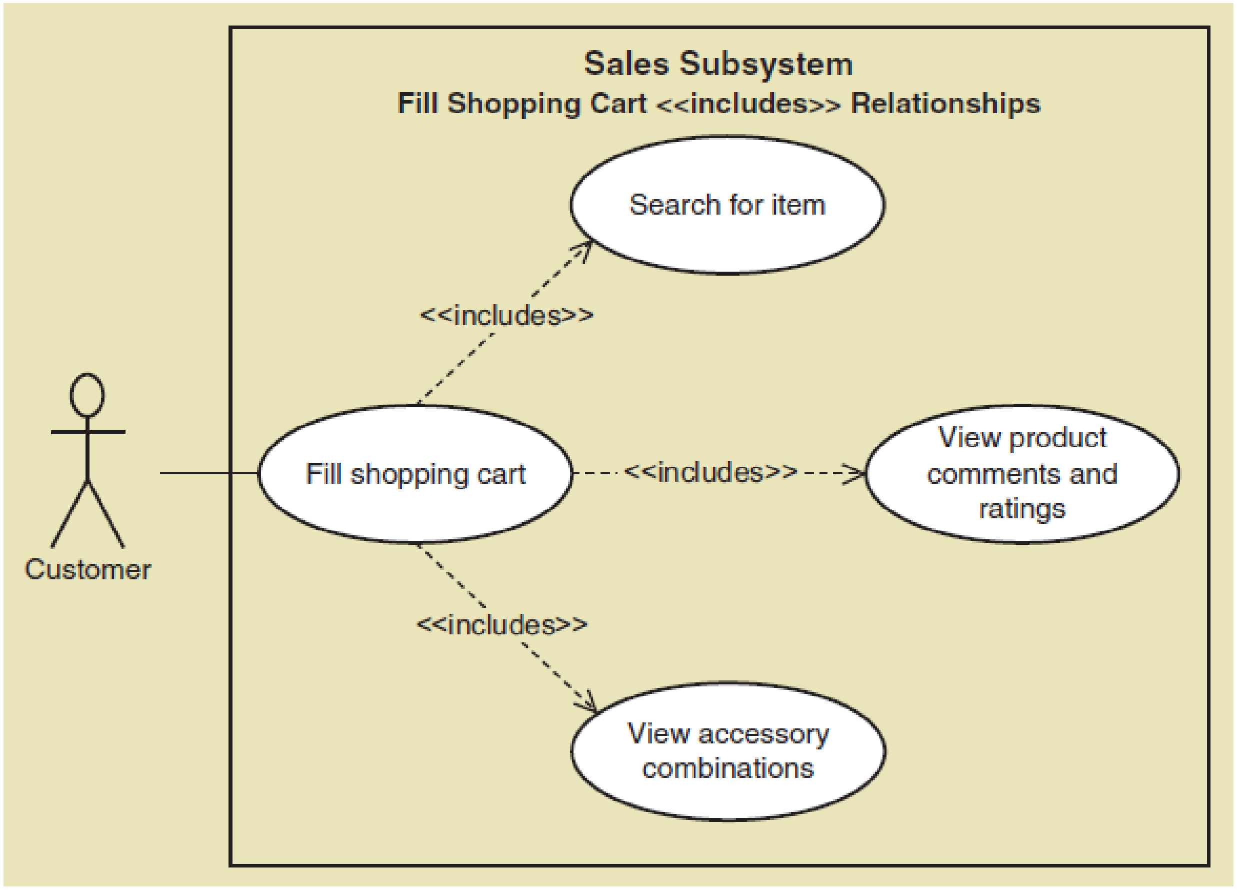



Customer Account Subsystem
All Actors
Create/update customer account
Request friend linkup
Reply to friend linkup
View "mountain bucks"
Browse messages
Process account adjustment
Send message
Transfer "mountain bucks"
Send/receive points
Customer
Customer service representative
Store sales representative
Management

Use Case Diagrams

The <<Includes>> relationship

- A relationship between use cases where one use case is stereotypically included within the other use case— like a called subroutine. Arrow points to subroutine


Sales Subsystem
Fill Shopping Cart <<includes>> Relationships
Search for item
<<includes>>
Fill shopping cart
<<includes>>
View product comments and ratings
<<includes>>
View accessory combinations
Customer

Things in the Problem Domain

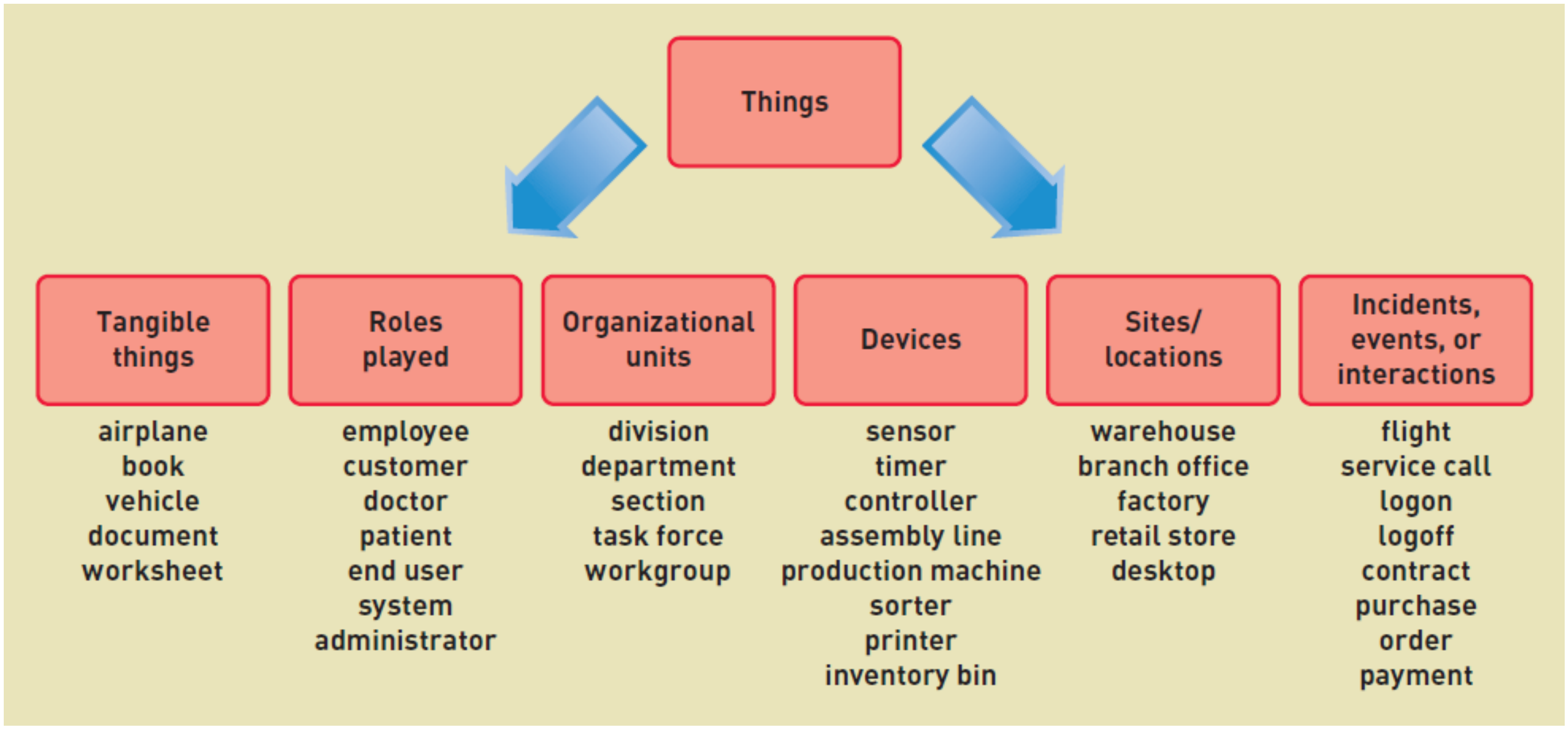
- Problem domain—the specific area (or domain) of the users' business need that is within the scope of the new system.
- "Things" are those items users work with when accomplishing tasks that need to be remembered (stored)
  - Modeled as domain classes or data entities
  - E.g. products, sales, shippers, customers, invoices, payments, etc.
  - OOA: similar to objects

Things in the Problem Domain
Two Techniques for Identifying them

- Brainstorming Technique
  - Use a checklist of all of the usual types of things typically found and brainstorm to identify domain classes of each type
- Noun Technique
  - Identify all of the nouns that come up when the system is described and determine if each is a domain class, an attribute, or not something we need to remember

Brainstorming Technique

- Are there any tangible things? Are there any organizational units? Sites/locations? Are there incidents or events that need to be recorded?



Brainstorming Technique: Steps

1. Identify a user and a set of use cases
2. Brainstorm with the user to identify things involved when carrying out the use case—that is, things about which information should be captured by the system.
3. Use the types of things (categories) to systematically ask questions about potential things, such as the following: Are there any tangible things you store information about? Are there any locations involved? Are there roles played by people that you need to remember?
4. Continue to work with all types of users and stakeholders to expand the brainstorming list
5. Merge the results, eliminate any duplicates, and compile an initial list

The Noun Technique

- A technique to identify problem domain classes (things) by finding, classifying, and refining a list of nouns that come up in in discussions or documents
- Popular technique. Systematic.
- Does end up with long lists and many nouns that are not things that need to be stored by the system
- Difficulty identifying synonyms and things that are really attributes
- Good place to start when there are no users available to help brainstorm

The Noun Technique: Steps

1. Using the use cases, actors, and other information about the system— including inputs and outputs— identify all nouns.
   - For the RMO CSMS, the nouns might include customer, product item, sale, confirmation, transaction, shipping, bank, change request, summary report, management, transaction report, accounting, back order, back order notification, return, return confirmation…

2. Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed.
   - For the RMO CSMS, these might include price, size, color, style, season, inventory quantity, payment method, and shipping address.

The Noun Technique: Steps (continued)

3. As this list of nouns builds, refine it. Ask these questions about each noun to help you decide whether you should include it:
    - Is it a unique thing the system needs to know about?
    - Is it inside the scope of the system I am working on?
    - Does the system need to remember more than one of these items?

   Ask these questions to decide to exclude it:
    - Is it really a synonym for some other thing I have identified?
    - Is it really just an output of the system produced from other information I have identified?
    - Is it really just an input that results in recording some other information I have identified?

   Ask these questions to research it:
    - Is it likely to be a specific piece of information (attribute) about some other thing I have identified?
    - Is it something I might need if assumptions change?

The Noun Technique: Steps (continued)

4. Create a master list of all nouns identified and then note whether each one should be included, excluded, or researched further.

5. Review the list with users, stakeholders, and team members and then define the list of things in the problem domain.

Partial List of Nouns for RMO

With notes on whether to include as domain class

Identified noun	Notes on including noun as a thing to store
Accounting	We know who they are. No need to store it.
Back order	A special type of order? Or a value of order status? Research.
Back-order information	An output that can be produced from other information.
Bank	Only one of them. No need to store.
Catalog	Yes, need to recall them, for different seasons and years. Include.
Catalog activity reports	An output that can be produced from other information. Not stored.
Catalog details	Same as catalog? Or the same as product items in the catalog? Research.
Change request	An input resulting in remembering changes to an order.
Charge adjustment	An input resulting in a transaction.
Color	One piece of information about a product item.
Confirmation	An output produced from other information. Not stored.
Credit card information	Part of an order? Or part of customer information? Research.
Customer	Yes, a key thing with lots of details required. Include.
Customer account	Possibly required if an RMO payment plan is included. Research.
Fulfillment reports	An output produced from information about shipments. Not stored.
Inventory quantity	One piece of information about a product item. Research.
Management	We know who they are. No need to store.
Marketing	We know who they are. No need to store.
Merchandising	We know who they are. No need to store.

Details about Domain Classes

- Attribute— describes one piece of information about each instance of the class
  - Customer has first name, last name, phone number
- Identifier or key
  - One attribute uniquely identifies an instance of the class. Required for data entities, optional for domain classes. Customer ID identifies a customer
- Compound attribute
  - Two or more attributes combined into one structure to simplify the model. (E.g., address rather than including number, street, city, state, zip separately). Sometimes an identifier or key is a compound attribute.

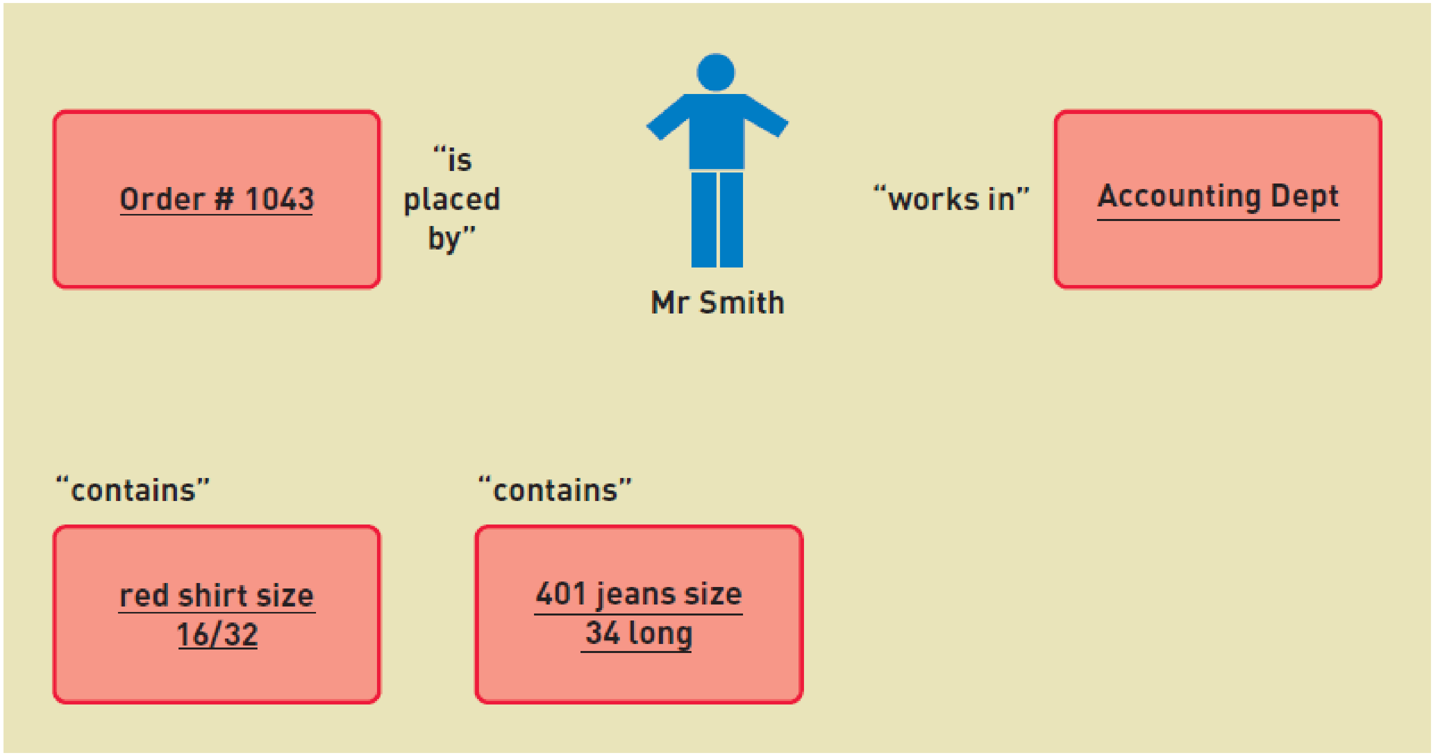
Attributes and Values

- Class is a type of thing. Object is a specific instance of the class. Each instance has its own values for an attribute

All customers have these attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First name	John	Mary	Bill
Last name	Smith	Jones	Casper
Home phone	555-9182	423-1298	874-1297
Work phone	555-3425	423-3419	874-8546

Associations Among Things

- Association— a naturally occurring relationship between classes

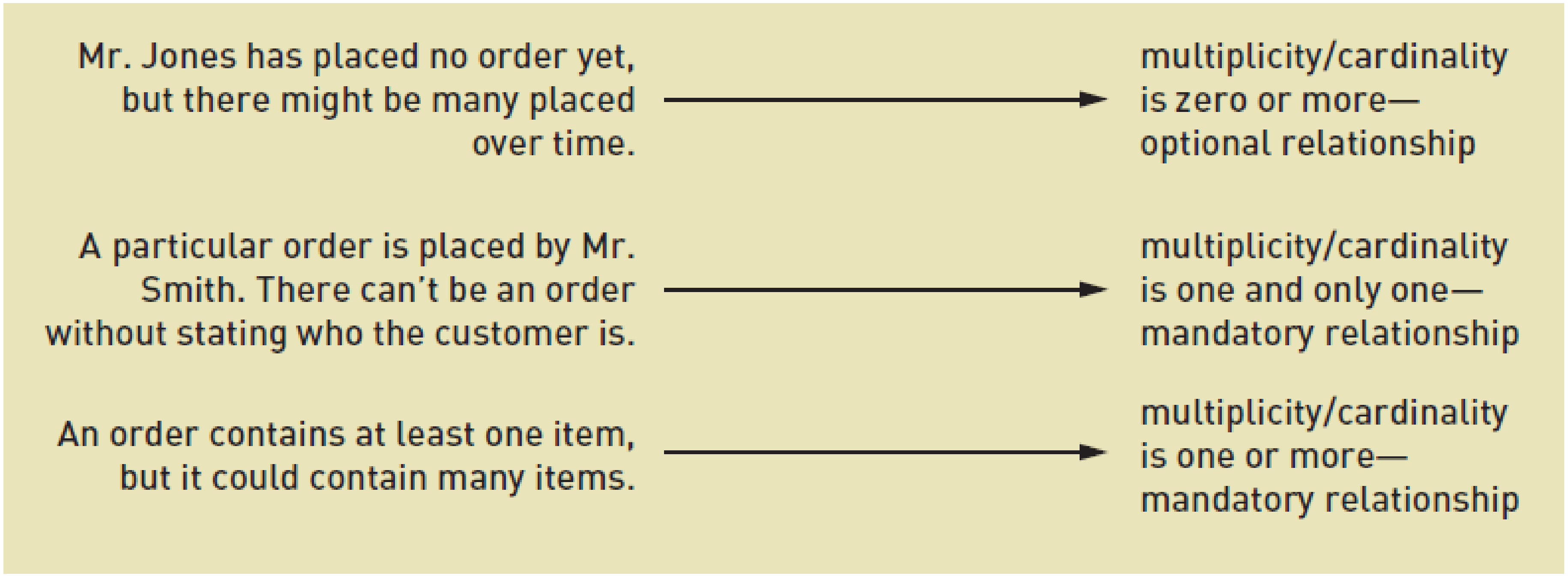


Just to Clarify…

- Called ***association*** on class diagram in UML
  - Multiplicity is term for the number of associations between classes: 1 to 1 or 1 to many
- Called ***relationship*** on ERD
  - Cardinality is term for number of relationships in entity relationship diagrams: 1 to 1 or 1 to many
- Associations and Relationships apply in two directions
  - Read them separately each way
  - A customer places an order
  - An order is placed by a customer

Minimum and Maximum Multiplicity

- Associations have minimum and maximum constraints
  - minimum is zero, the association is optional
  - If minimum is at least one, the association is mandatory



Mr. Jones has placed no order yet, but there might be many placed over time.	$\rightarrow$	multiplicity/cardinality 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.	$\rightarrow$	multiplicity/cardinality is one and only one—mandatory relationship
An order contains at least one item, but it could contain many items.	$\rightarrow$	multiplicity/cardinality is one or more—mandatory relationship

Types of Associations

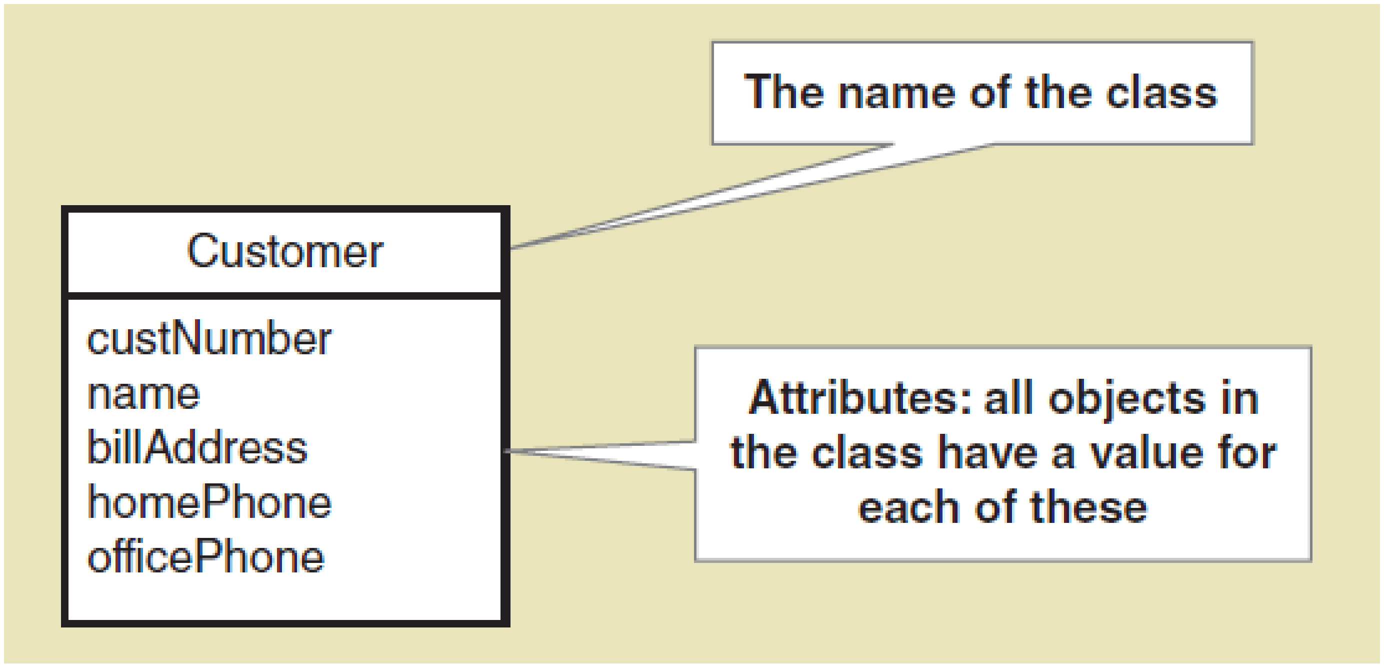
- Binary Association
  - Associations between exactly two different classes
    - “Course Section includes Students”
    - “Members join Club”
- Unary Association (recursive)
  - Associations between two instances of the same class
    - “Person married to person”
    - “Part is made using parts”
- Ternary Association (three)
  - Associations between three classes
    - “one order associated with a specific customer plus a specific sales representative”
- N-ary Association (between n)

The Domain Model Class Diagram

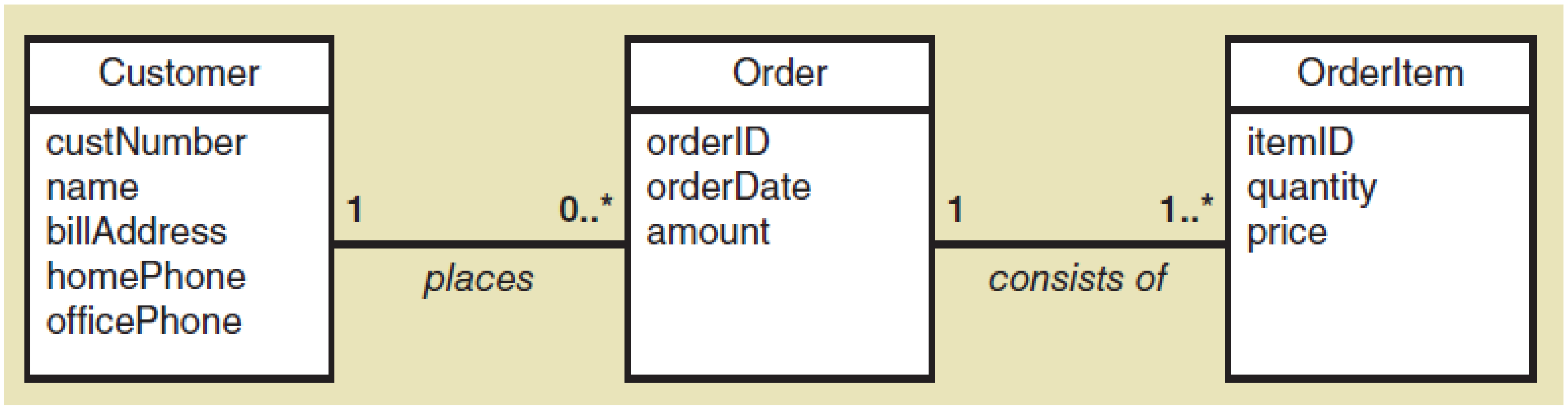
- Class
  - A category of classification used to describe a collection of objects
- Domain Class
  - Classes that describe objects in the problem domain
- Class Diagram
  - A UML diagram that shows classes with attributes and associations (plus methods if it models software classes)
- Domain Model Class Diagram
  - A class diagram that only includes classes from the problem domain, not software classes so no methods

Domain Class Notation

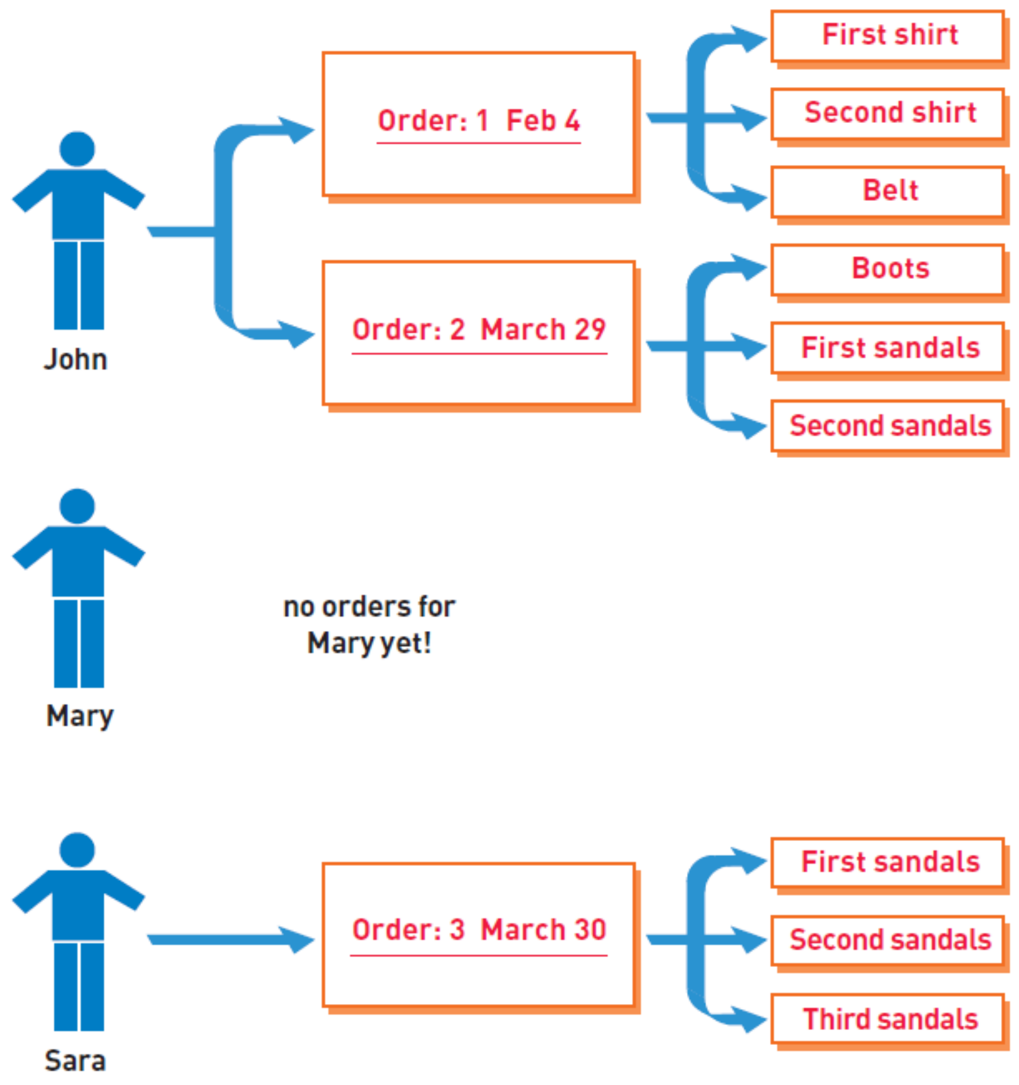
- Domain class has no methods



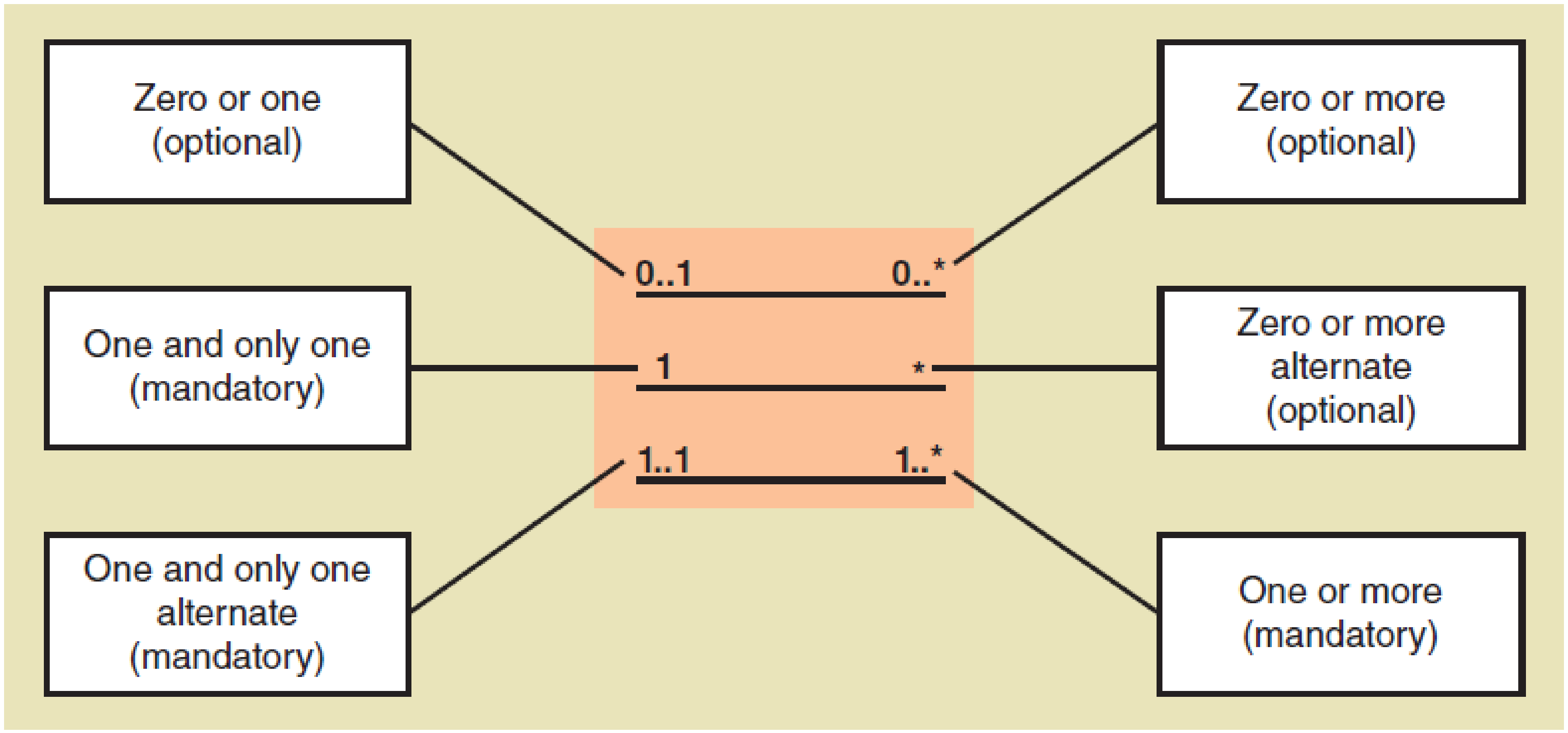
A Simple Domain Model Class Diagram



Transactions Example



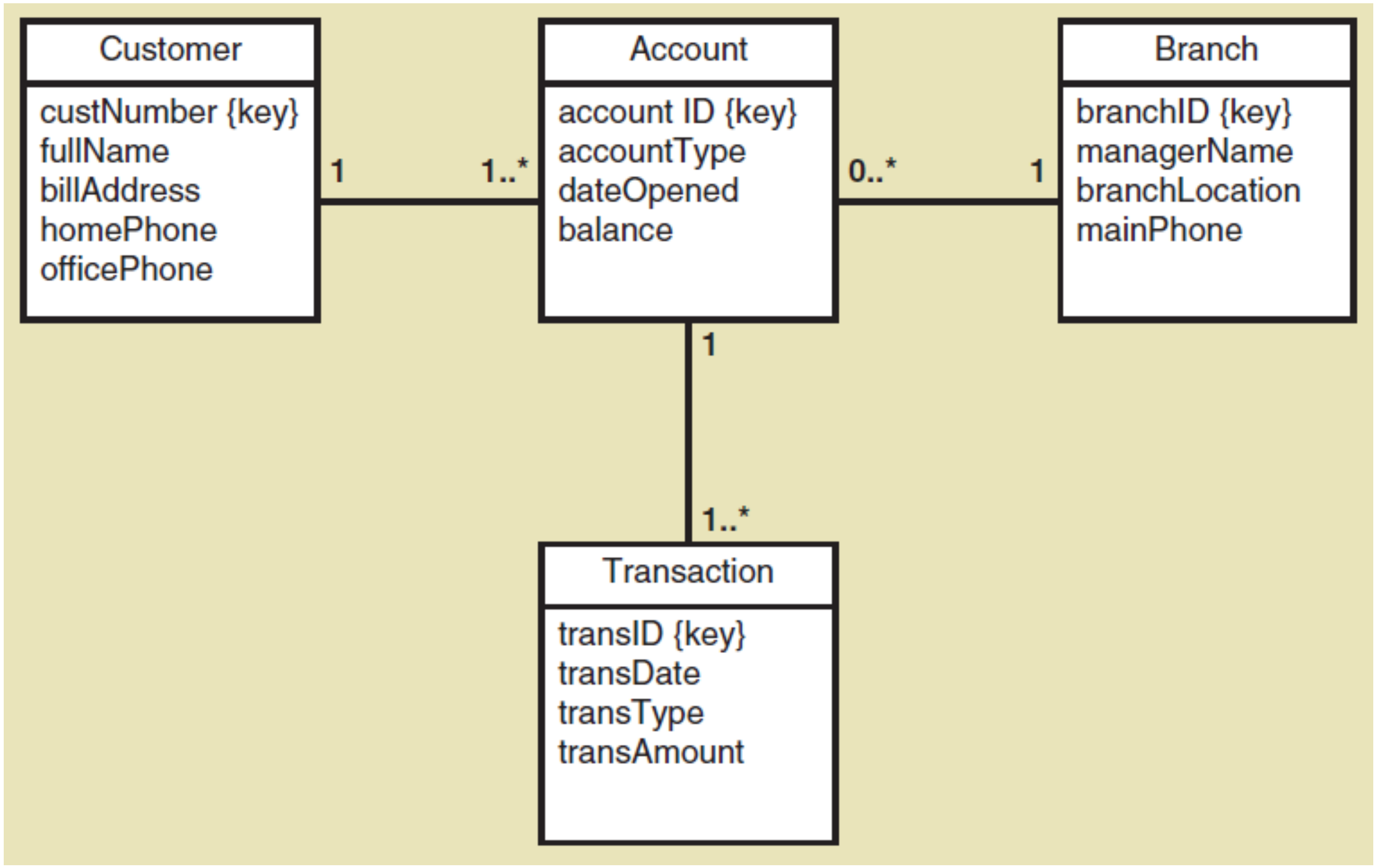
UML Notation for Multiplicity



Domain Model Class Diagram

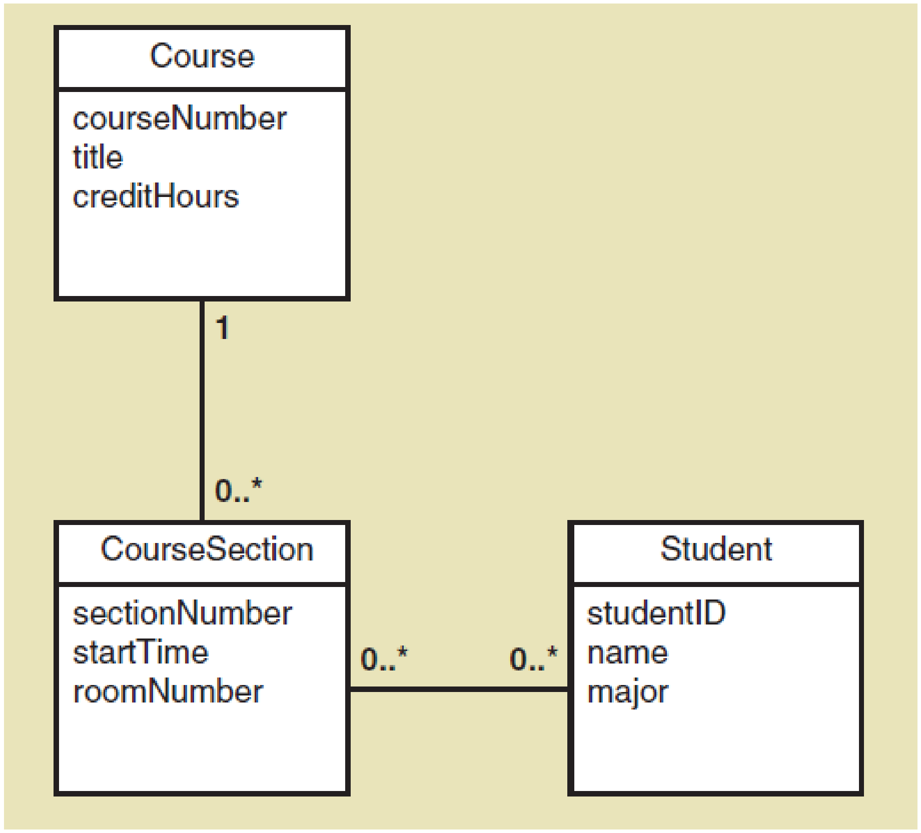
for a bank with many branches

A bank has branches where customers can open accounts and make transactions.



Domain Model Class Diagram
for course enrollment at a university

A university offers courses. Students can enroll in the course sections.

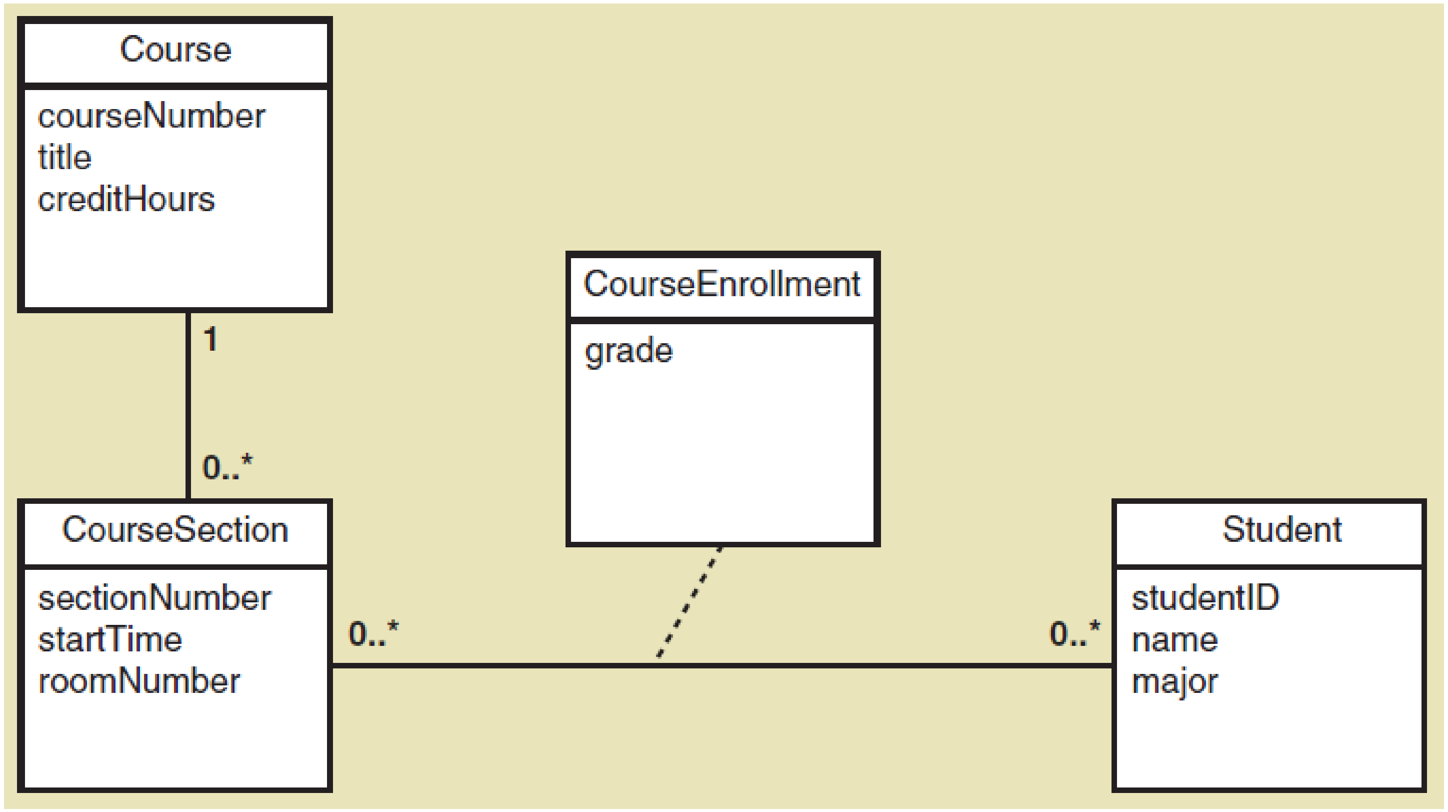


- Where is each student's grade remembered in this model?
  - Each section has many grades and each grade is association with a student
  - Each student has many grades and each grade is association with a section

Refined Course Enrollment Model

with an Association Class CourseEnrollment

- Association class— an association that is treated as a class in a many to many association because it has attributes that need to be remembered, such as grade


Course
courseNumber
title
creditHours
1
0..*
CourseSection
sectionNumber
startTime
roomNumber
CourseEnrollment
grade
0..*
0..*
Student
studentID
name
major