HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

System Analysis and Design

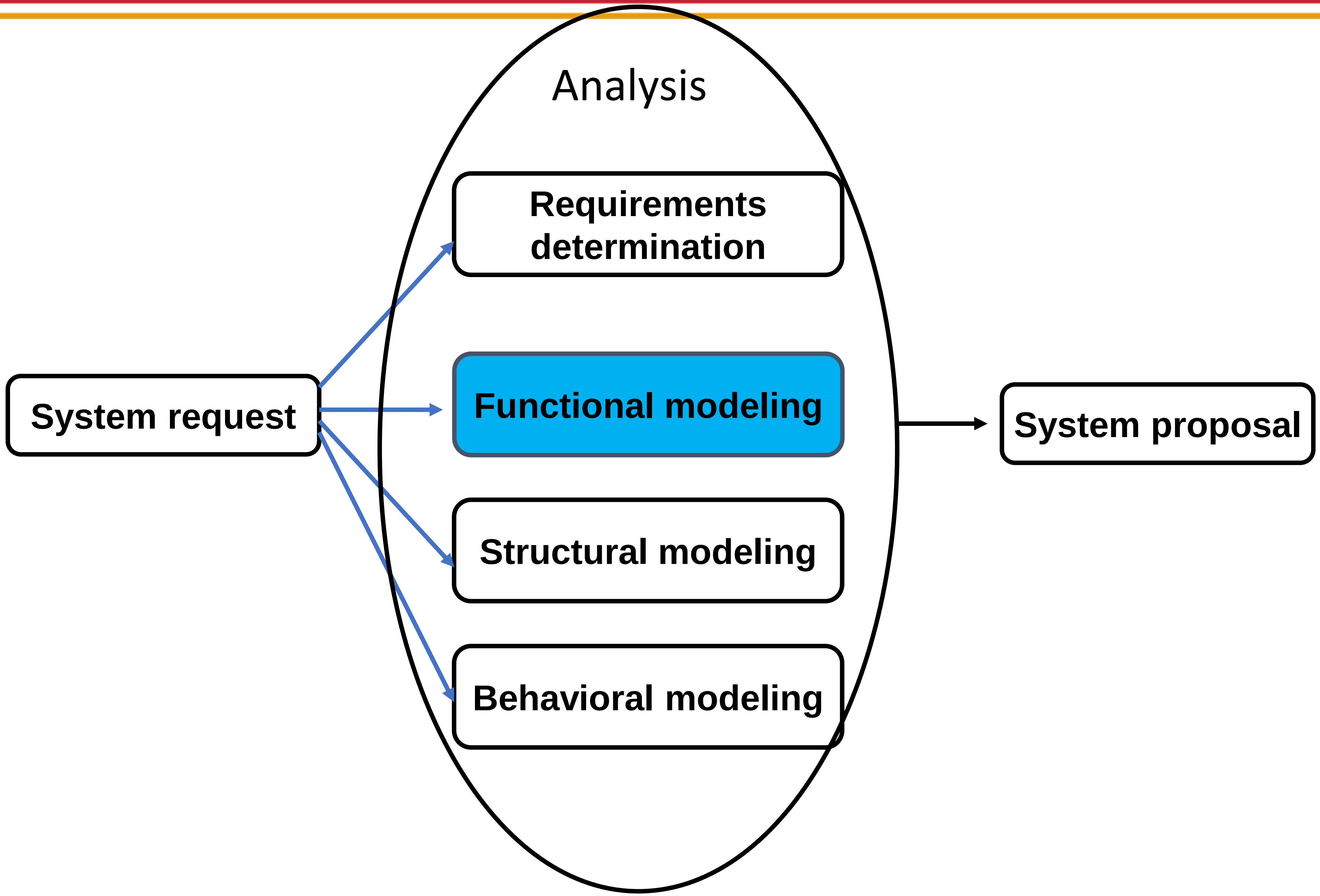
IT3120E

ONE LOVE. ONE FUTURE.

Part 2: System analysis

Chapter 5: *Functional modeling*

Part 2: System analysis



Analysis

System request

Requirements determination

Functional modeling

Structural modeling

Behavioral modeling

System proposal

Chapter syllabus

- Introduction
- Domain modeling
- Use-Case diagrams
- Use-Case description
- Activity diagrams
- Functional Model Verification and Validation

5.1. Introduction

- Functional models describe business processes and the interaction of an information system with its environment
- Business processes are sequential steps that a business follows to deliver a service
- Functional modeling defines
  - what the system should be capable of doing
  - how the user and the system will interact

5.1. Introduction

- Functional Modeling Steps
  - From system requirements
  - *Project team* identifies *business processes* and their *environment* → use-case diagrams
  - *Users* + *project team* document the business processes → use-case description for each use case
  - *Project team* draws the business processes of each use case → activity diagrams
  - *Project team* verifies and validates the business processes to ensure that all models (use-case diagrams, use-case descriptions, activity diagrams) agree with each other

5.1. Introduction

Use-Case and Activity Diagrams

- Are *logical models*:
  - describe the business domain’s activities without suggesting how they are conducted
- Focus on
  - How the business should run

5.1. Introduction

- Input
  - *Requirements definition report*

- Purpose
  - How the information in *requirements definition report* is organized and presented in the form of use-case diagrams, use-case descriptions and activity diagrams.

- Output
  - Use-case diagram
  - Use-case description
  - Activity diagram

→ Functional Modeling

5.2 Domain modeling

5.2.1 Concepts

- Domain modeling:
  - A project glossary, or a dictionary of terms used in the project
  - Provides a common vocabulary to enable *clear communication* between members of a project team

    → defines the scope on which to build the use cases

5.2.2. Domain Modeling Steps

- Identify objects from High-Level Requirements
  - Scan these requirements
  - Extracting the nouns and noun phrases.
- Refine the Domain Model
  - Eliminate the unnecessary terms/nonsense terms
  - Eliminate the duplicate terms
  - Modify to valid concept
  - Identify further domain objects that weren’t in the requirements, but are important to the problem
  - When in doubt, talk to the customer. Ask questions until getting a clear, unambiguous answer.
  - Review the relationships
- Draw the domain model to describe a clear relationship between objects

Refer to ICONIX Process: Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

CASE STUDY: Internet Bookstore

High-level requirements

1. The bookstore will be web based initially, but it must have a sufficiently flexible architecture that alternative front-ends may be developed (Swing/applets, web services, etc.).

2. The bookstore must be able to sell books, with orders accepted over the Internet.

3. The user must be able to add books into an online shopping cart, prior to checkout.

a. Similarly, the user must be able to remove items from the shopping cart.

4. The user must be able to maintain wish lists of books that he or she wants to purchase later.

5. The user must be able to cancel orders before they’ve shipped.

6. The user must be able to pay by credit card or purchase order.

7. It must be possible for the user to return books.

8. The bookstore must be embeddable into associate partners’ websites using mini-catalogs, which are derived from an overall master catalog stored in a central database.

a. The mini-catalogs must be defined in XML, as they will be transferred between this and (later to be defined) external systems.

b. The shipping fulfillment system shall be carried out via Amazon Web Services.

9. The user must be able to create a customer account, so that the system remembers the user’s details (name, address, credit card details) at login.

a. The system shall maintain a list of accounts in its central database.

b. When a user logs in, his or her password must always be matched against the passwords in the master account list.

CASE STUDY: Internet Bookstore

High-level requirements

1. The **bookstore** will be web based initially, but it must have a sufficiently flexible architecture that alternative front-ends may be developed (Swing/applets, web services, etc.).

2. The bookstore must be able to sell **books**, with **orders** accepted over the **Internet**.

3. The user must be able to add books into an online **shopping cart**, prior to **checkout**.

a. Similarly, the user must be able to remove **items** from the shopping cart.

4. The user must be able to maintain **wish lists** of books that he or she wants to purchase later.

5. The user must be able to cancel orders before they've shipped.

6. The user must be able to pay by **credit card** or **purchase order**.

7. It must be possible for the user to return books.

8. The bookstore must be embeddable into **associate partners**' websites using **mini-catalogs**, which are derived from an overall **master catalog** stored in a central **database**.

a. The mini-catalogs must be defined in XML, as they will be transferred between this and (later to be defined) external systems.

b. The **shipping fulfillment system** shall be carried out via Amazon Web Services.

9. The user must be able to create a **customer account**, so that the system remembers the user's details (name, address, credit card details) at login.

a. The system shall maintain a **list of accounts** in its central database.

b. When a user logs in, his or her **password** must always be matched against the passwords in the **master account list**.

CASE STUDY: Internet Bookstore

High-level requirements

10. The user must be able to search for books by various search methods—title, author, keyword, or category—and then view the books' details.

11. It must be possible for the user to post reviews of favorite books; the review comments should appear on the book details screen. The review should include a customer rating (1–5), which is usually shown along with the book title in book lists.

a. Book reviews must be moderated—that is, checked and "OK'd" by a member of staff before they're published on the website.

b. Longer reviews should be truncated on the book details screen; the customer may click to view the full review on a separate page.

12. It must be possible for staff to post editorial reviews of books. These should also appear on the book details screen.

13. The bookstore shall allow third-party sellers (e.g., second-hand bookstores) to add their own individual book catalogs. These are added into the overall master book catalog so that sellers' books are included in search results.

14. The bookstore must be scalable, with the following specific requirements:

a. The bookstore must be capable of maintaining user accounts for up to 100,000 customers in its first six months, and then a further 1,000,000 after that.

b. The bookstore must be capable of serving up to 1,000 simultaneous users (10,000 after six months).

c. The bookstore must be able to accommodate up to 100 search requests per minute (1,000/minute after six months).

d. The bookstore must be able to accommodate up to 100 purchases per hour (1,000/hour after six months).

CASE STUDY: Internet Bookstore
High-level requirements

10. The user must be able to search for books by various **search methods—title**, **author**, **keyword**, or **category—and** then view the **books' details**.

11. It must be possible for the user to post reviews of favorite books; the **review comments** should appear on the book details screen. The review should include a **customer rating** (1–5), which is usually shown along with the book title in **book lists.**

a. **Book reviews** must be moderated—that is, checked and "OK'd" by a member of staff before they're published on the website.

b. Longer reviews should be truncated on the book details screen; the **customer** may click to view the full review on a separate page.

12. It must be possible for staff to post **editorial reviews** of books. These should also appear on the book details screen.

13. The bookstore shall allow third-party **sellers** (e.g., second-hand bookstores) to add their own individual **book catalogs**. These are added into the overall **master book catalog** so that sellers' books are included in search results.

14. The bookstore must be scalable, with the following specific requirements:

a. The bookstore must be capable of maintaining **user accounts** for up to 100,000 customers in its first six months, and then a further 1,000,000 after that.

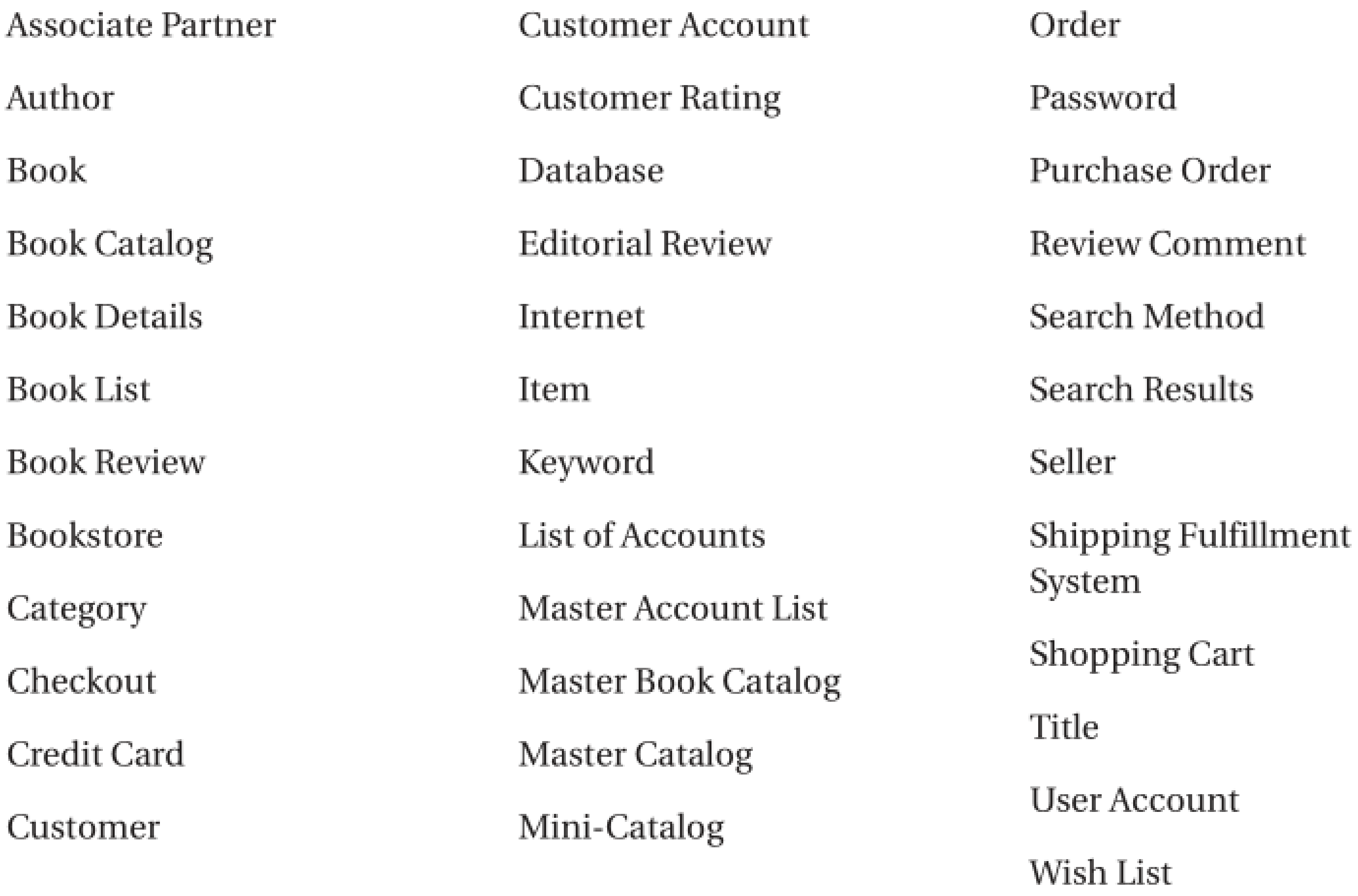
b. The bookstore must be capable of serving up to 1,000 simultaneous users (10,000 after six months).

c. The bookstore must be able to accommodate up to 100 search requests per minute (1,000/minute after six months).

d. The bookstore must be able to accommodate up to 100 purchases per hour (1,000/hour after six months).

CASE STUDY: Internet Bookstore

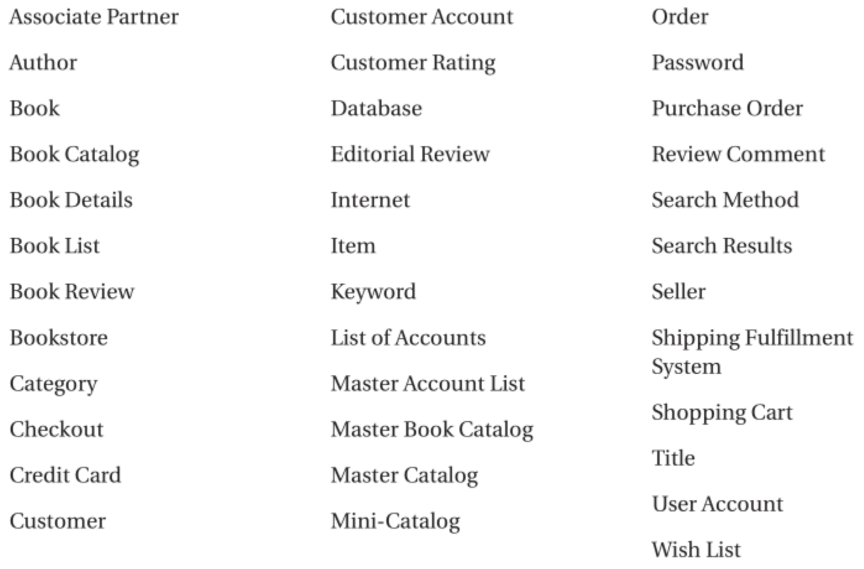
- Put all the highlighted nouns and noun phrases into a list



Associate Partner
Author
Book
Book Catalog
Book Details
Book List
Book Review
Bookstore
Category
Checkout
Credit Card
Customer
Customer Account
Customer Rating
Database
Editorial Review
Internet
Item
Keyword
List of Accounts
Master Account List
Master Book Catalog
Master Catalog
Mini-Catalog
Order
Password
Purchase Order
Review Comment
Search Method
Search Results
Seller
Shipping Fulfillment System
Shopping Cart
Title
User Account
Wish List

CASE STUDY: Internet Bookstore

- Eliminate the unnecessary terms
  - *Internet* is too generic
  - *Title*, *Keyword*, *Password* is a too small to be an object
- Identify and eliminate the duplicate terms
  - *User Account* and *Customer Account* → *Customer Account*
  - *List of Accounts* and *Master Account List* → *Master Account List*
  - *Book Review* and *Review Comment* → *Book Review*
  - *Book Catalog* and *Master Catalog* *Book Catalog*
  - etc.
- Modify the valid concept
  - *Item* → *Line Item*
- Need two additional domain objects: *Order History* and *Order Dispatch*
- Remove *Checkout, Author*



Associate Partner
Author
Book
Book Catalog
Book Details
Book List
Book Review
Bookstore
Category
Checkout
Credit Card
Customer

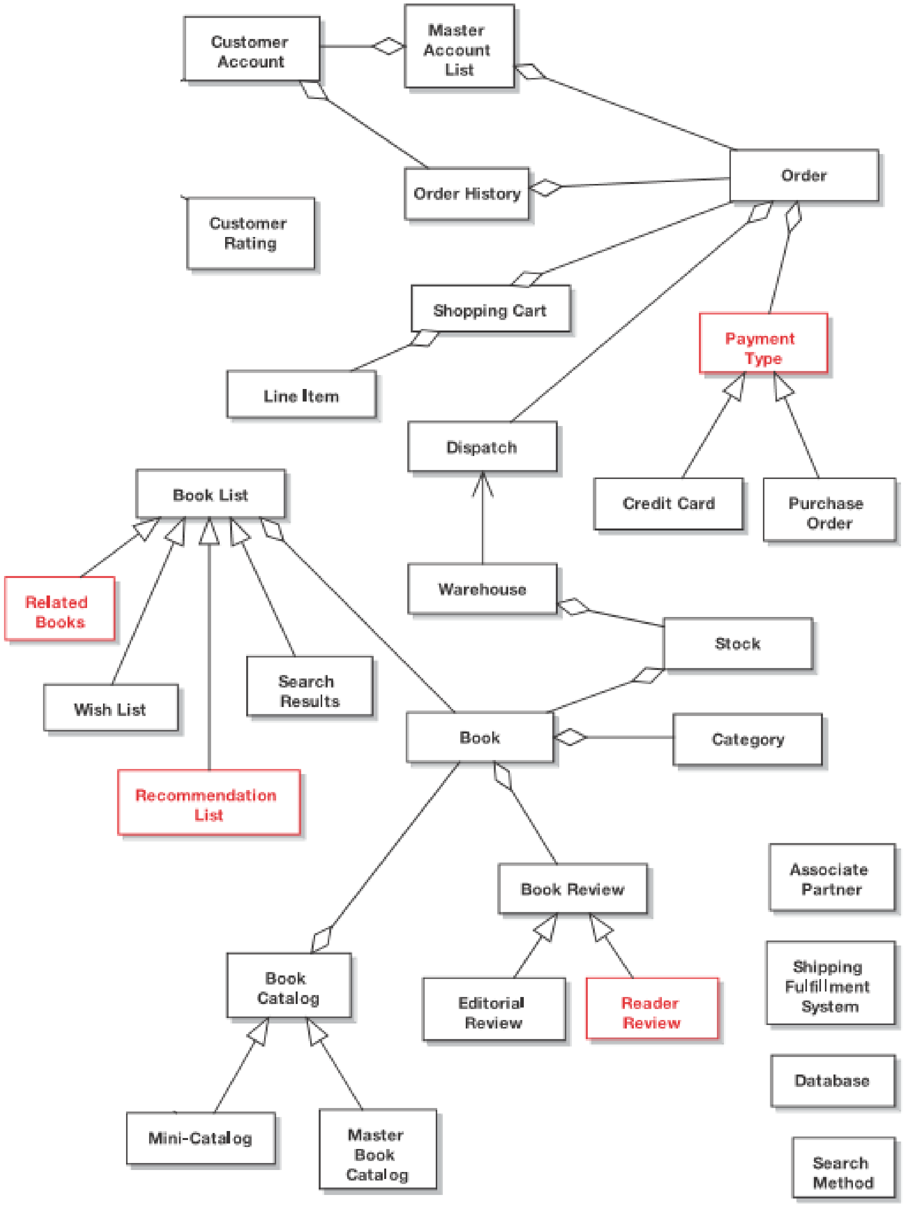
Customer Account
Customer Rating
Database
Editorial Review
Internet
Item
Keyword
List of Accounts
Master Account List
Master Book Catalog
Master Catalog
Mini-Catalog

Order
Password
Purchase Order
Review Comment
Search Method
Search Results
Seller
Shipping Fulfillment System
Shopping Cart
Title
User Account
Wish List

CASE STUDY: Internet Bookstore

Review the relationships:

- Building generalization relationships (is-a
- Building aggregation relationships (has-a



Domain Modeling guidelines

- Focus on real-world (problem domain) objects.
- Show how the objects relate to each other (use generalization (is-a) and aggregation (has-a) relationships)
- Limit your initial domain modeling efforts to a couple of hours
- Don’t mistake your domain model for a data model
- Don’t confuse an object (which represents a single instance) with a database table (which contains a collection of things)
- Do your initial domain model before you write your use cases, to avoid name ambiguity

5.3 Use-Case diagrams

5.3.1 Concepts

- Use-case (UC): What is it?
  - Describe the basic functions of the system from the user's point of view
  - Model the interaction between the user, the system, and the environment
  - Give the overview of the business processes without specifying how those operations are implemented.
  - Foundation for testing and user-interface design
- Use case diagrams: describing the basic ***functions of the system***.

- Start with the Domain Model then Use Cases
  - Use case text should be grounded in reality, very close to the system.
  - Use case should be written in the context of the object model
  - Domain model ~ simplified class diagram

5.3.2 Use Case diagrams

- From the High-Level Requirements, identify the functions of the system (~verb phase)
  - add the other functions if needed
- One use case describes only one function
- Organize use cases into packages
  - a way of grouping related elements
- Organize use cases with actors

The Internet Bookstore

High-level requirements

1. The bookstore will be web based initially, but it must have a sufficiently flexible architecture that alternative front-ends may be developed (Swing/applets, web services, etc.).

2. The bookstore must be able to sell books, with orders accepted over the Internet.

3. The user must be able to add books into an online shopping cart, prior to checkout.

a. Similarly, the user must be able to remove items from the shopping cart.

4. The user must be able to maintain wish lists of books that he or she wants to purchase later.

5. The user must be able to cancel orders before they've shipped.

6. The user must be able to pay by credit card or purchase order.

7. It must be possible for the user to return books.

8. The bookstore must be embeddable into associate partners' websites using mini-catalogs, which are derived from an overall master catalog stored in a central database.

a. The mini-catalogs must be defined in XML, as they will be transferred between this and (later to be defined) external systems.

b. The shipping fulfillment system shall be carried out via Amazon Web Services.

9. The user must be able to create a customer account, so that the system remembers the user's details (name, address, credit card details) at login.

a. The system shall maintain a list of accounts in its central database.

b. When a user logs in, his or her password must always be matched against the passwords in the master account list.

Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

The Internet Bookstore
High-level requirements

1. The bookstore will be web based initially, but it must have a sufficiently flexible architecture that alternative front-ends may be developed (Swing/applets, web services, etc.).

2. The bookstore must be able to **sell books**, with orders accepted over the Internet.

3. The user must be able to **add books into an online shopping cart**, prior to **checkout**.

a. Similarly, the user must be able to **remove items from the shopping cart**.

4. The user must be able to **maintain wish lists** of books that he or she wants to purchase later.

5. The user must be able to **cancel orders** before they've shipped.

6. The user must be able to **pay by credit card or purchase order**.

7. It must be possible for the user to **return books**.

8. The bookstore must be embeddable into associate partners' websites using mini-catalogs, which are derived from an overall master catalog stored in a central database.

a. The mini-catalogs must be defined in XML, as they will be transferred between this and (later to be defined) external systems.

b. The shipping fulfillment system shall be carried out via Amazon Web Services.

9. The user must be able to **create a customer account**, so that the system remembers the user's details (name, address, credit card details) at login.

a. The system shall maintain a list of accounts in its central database.

b. When a user **logs in**, his or her password must always be matched against the passwords in the master account list.

The Internet Bookstore
High-level requirements

10. The user must be able to search for books by various search methods—title, author, keyword, or category—and then view the books' details.

11. It must be possible for the user to post reviews of favorite books; the review comments should appear on the book details screen. The review should include a customer rating (1–5), which is usually shown along with the book title in book lists.

a. Book reviews must be moderated—that is, checked and "OK'd" by a member of staff before they're published on the website.

b. Longer reviews should be truncated on the book details screen; the customer may click to view the full review on a separate page.

12. It must be possible for staff to post editorial reviews of books. These should also appear on the book details screen.

13. The bookstore shall allow third-party sellers (e.g., second-hand bookstores) to add their own individual book catalogs. These are added into the overall master book catalog so that sellers' books are included in search results.

14. The bookstore must be scalable, with the following specific requirements:

a. The bookstore must be capable of maintaining user accounts for up to 100,000 customers in its first six months, and then a further 1,000,000 after that.

b. The bookstore must be capable of serving up to 1,000 simultaneous users (10,000 after six months).

c. The bookstore must be able to accommodate up to 100 search requests per minute (1,000/minute after six months).

d. The bookstore must be able to accommodate up to 100 purchases per hour (1,000/hour after six months).

The Internet Bookstore

High-level requirements

10. The user must be able to **search for books** by various search methods—title, author, keyword, or category—and then view the books' details.

11. It must be possible for the user to **post reviews** of favorite books; the review comments should appear on the book details screen. The review should include a customer rating (1–5), which is usually shown along with the book title in book lists.

a. Book **reviews must be moderated**—that is, checked and "OK'd" by a member of staff before they're published on the website.

b. Longer reviews should be truncated on the book details screen; the customer may click to **view the full review** on a separate page.

12. It must be possible for staff to **post editorial reviews** of books. These should also appear on the book details screen.

13. The bookstore shall allow third-party sellers (e.g., second-hand bookstores) to **add their own individual book catalogs**. These are added into the overall master book catalog so that sellers' books are included in search results.

14. The bookstore must be scalable, with the following specific requirements:

a. The bookstore must be capable of maintaining user accounts for up to 100,000 customers in its first six months, and then a further 1,000,000 after that.

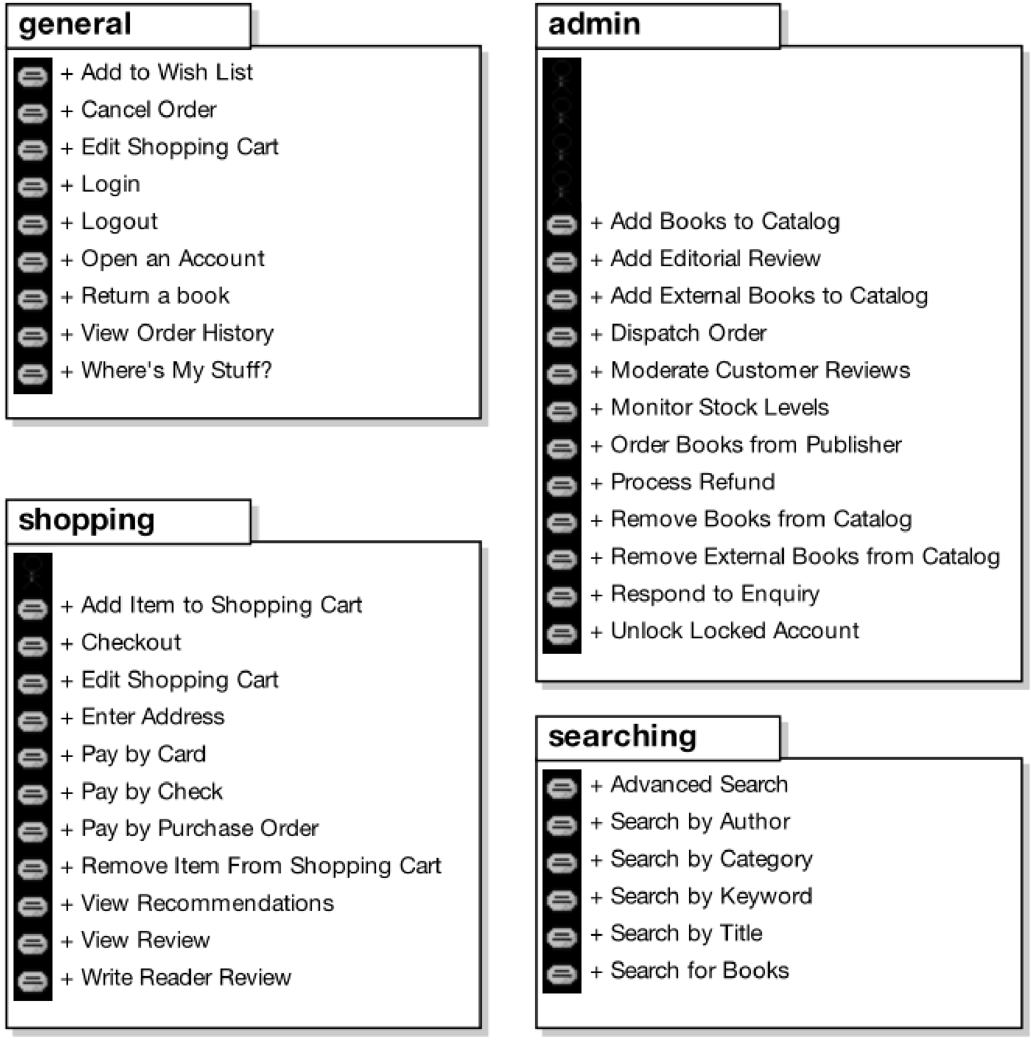
b. The bookstore must be capable of serving up to 1,000 simultaneous users (10,000 after six months).

c. The bookstore must be able to accommodate up to 100 search requests per minute (1,000/minute after six months).

d. The bookstore must be able to accommodate up to 100 purchases per hour (1,000/hour after six months).

CASE STUDY: Internet Bookstore

- Packages of use cases
- Actors: Customer, Customer Service, Seller, Shipping Clerk, and Webmaster



general

- + Add to Wish List
- + Cancel Order
- + Edit Shopping Cart
- + Login
- + Logout
- + Open an Account
- + Return a book
- + View Order History
- + Where's My Stuff?

shopping

- + Add Item to Shopping Cart
- + Checkout
- + Edit Shopping Cart
- + Enter Address
- + Pay by Card
- + Pay by Check
- + Pay by Purchase Order
- + Remove Item From Shopping Cart
- + View Recommendations
- + View Review
- + Write Reader Review

admin

- + Add Books to Catalog
- + Add Editorial Review
- + Add External Books to Catalog
- + Dispatch Order
- + Moderate Customer Reviews
- + Monitor Stock Levels
- + Order Books from Publisher
- + Process Refund
- + Remove Books from Catalog
- + Remove External Books from Catalog
- + Respond to Enquiry
- + Unlock Locked Account

searching

- + Advanced Search
- + Search by Author
- + Search by Category
- + Search by Keyword
- + Search by Title
- + Search for Books

5.3.2 Use Case diagrams

- Draw each use case
- Components:
  - System boundary (Subject): rectangle
    - Subject name: in the top left corner
  - Use case: oval
    - Use case name: Start with a verb, should be clear, short and concise
    - Inside the system
  - Actor: a “role” that users/external systems can play
    - External to the system
  - An association from actor to a use case
    - Actor is the one who carries out that use case.
    - A straight line without direction
    - The arrow head is optional but it’s commonly used to denote who initiates communication

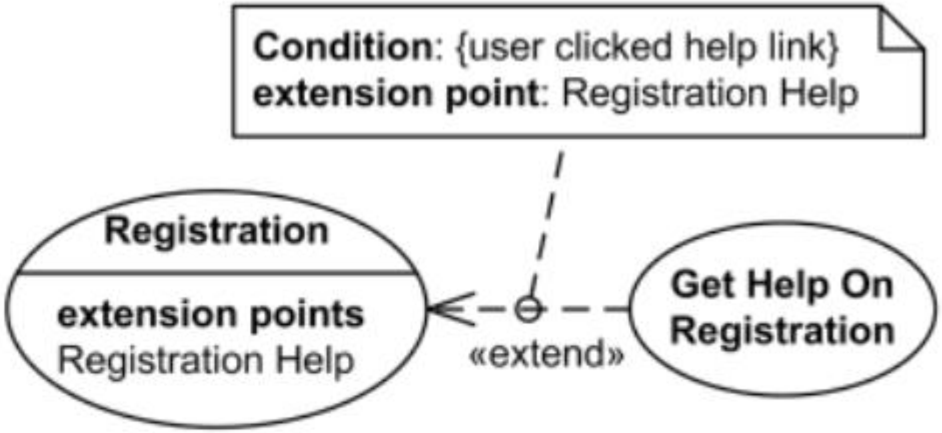
    - An actor must be associated with at least one use case.
    - An actor can be associated with multiple use cases.
    - Multiple actors can be associated with a single use case.

5.3.2 Use Case diagrams

- Components (cont.):
  - Relationship between actors
    - Generalization
  - Relationship between user cases
    - A <<precedes>> B: Use case A must take place in its entirety before use case B begins
    - A <<invokes>> B: Use case B (can) happens during the lifespan of use case A.
      - A <<includes>> B: Halfway through use case A, use case B is called (at a location specified in A). When B finishes, A carries on from where it left off
      - A <<extends>> B: at some extension points, adds additional parent's functionality (Optional)
    - Generalization relationship: overrides or totally replaces steps of the parent use case

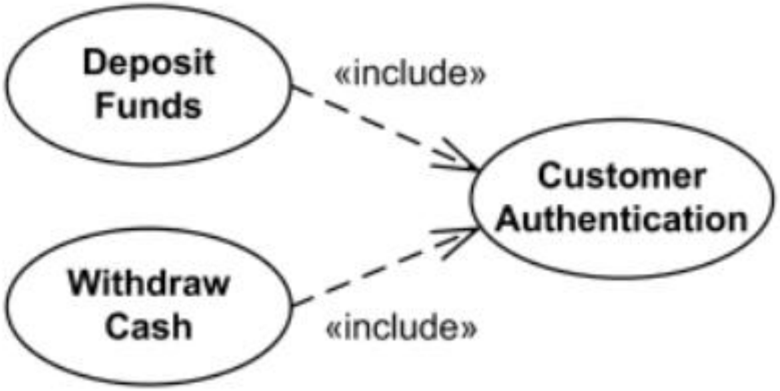
5.3.2 Use Case diagrams

- Use Case Extend
  - **Extended** use case is meaningful on its own, it is **independent** of the extending use case
  - **Extending** use case typically defines **optional** behavior that is not necessarily meaningful by itself.
  - The extension takes place at one or more extension points (location + condition) defined in the extended use case.
  - The extension points are *optionally* shown



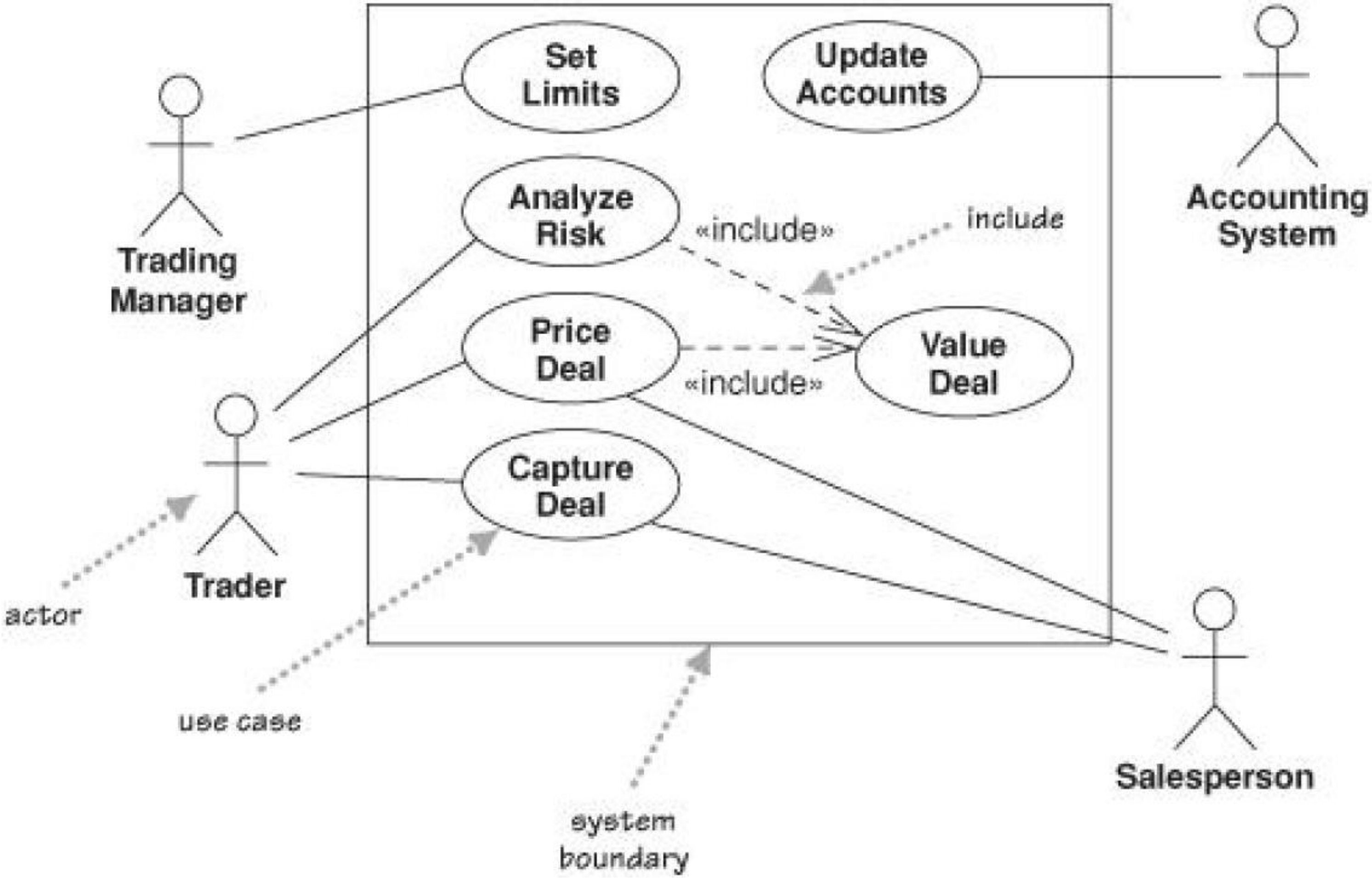
5.3.2 Use Case diagrams

- Use Case Include
  - The **included** use case (the addition) is inserted into the behavior of the **including** (the base) use case.
    - To simplify large use case by splitting it into several use cases
    - To extract **common parts** of the behaviors of two or more use cases.
  - Base use case is incomplete (abstract use case).
  - There are no explicit "inclusion points location” (but implicit)



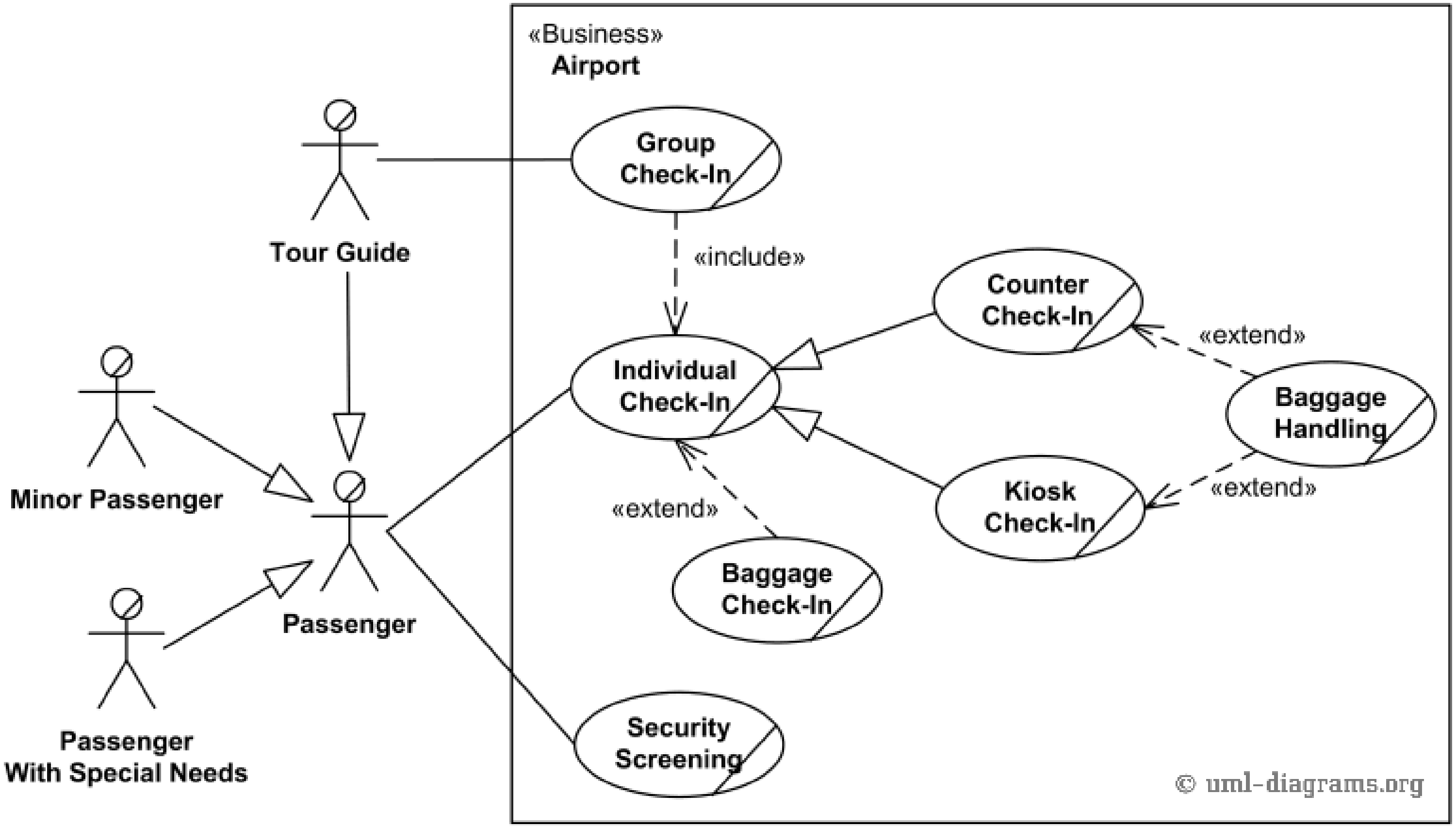
5.3.2 Use Case diagrams

- Example

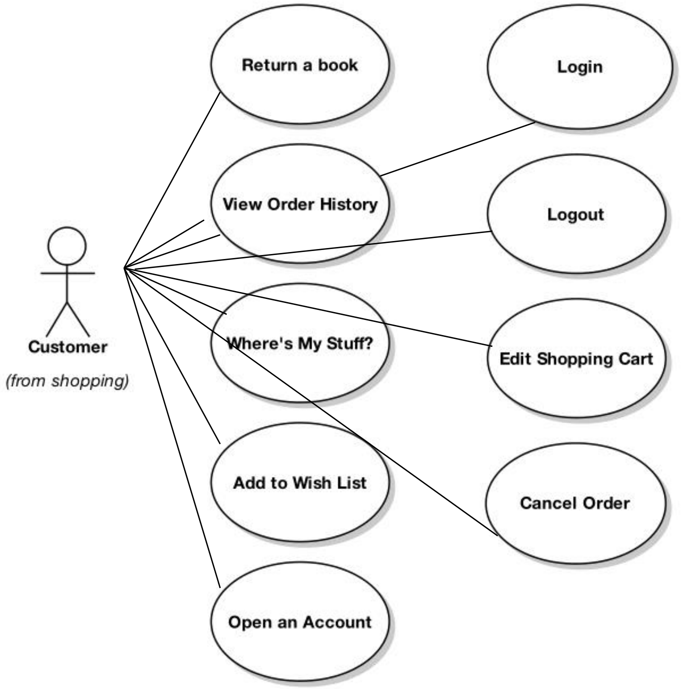


5.3.2 Use Case diagrams

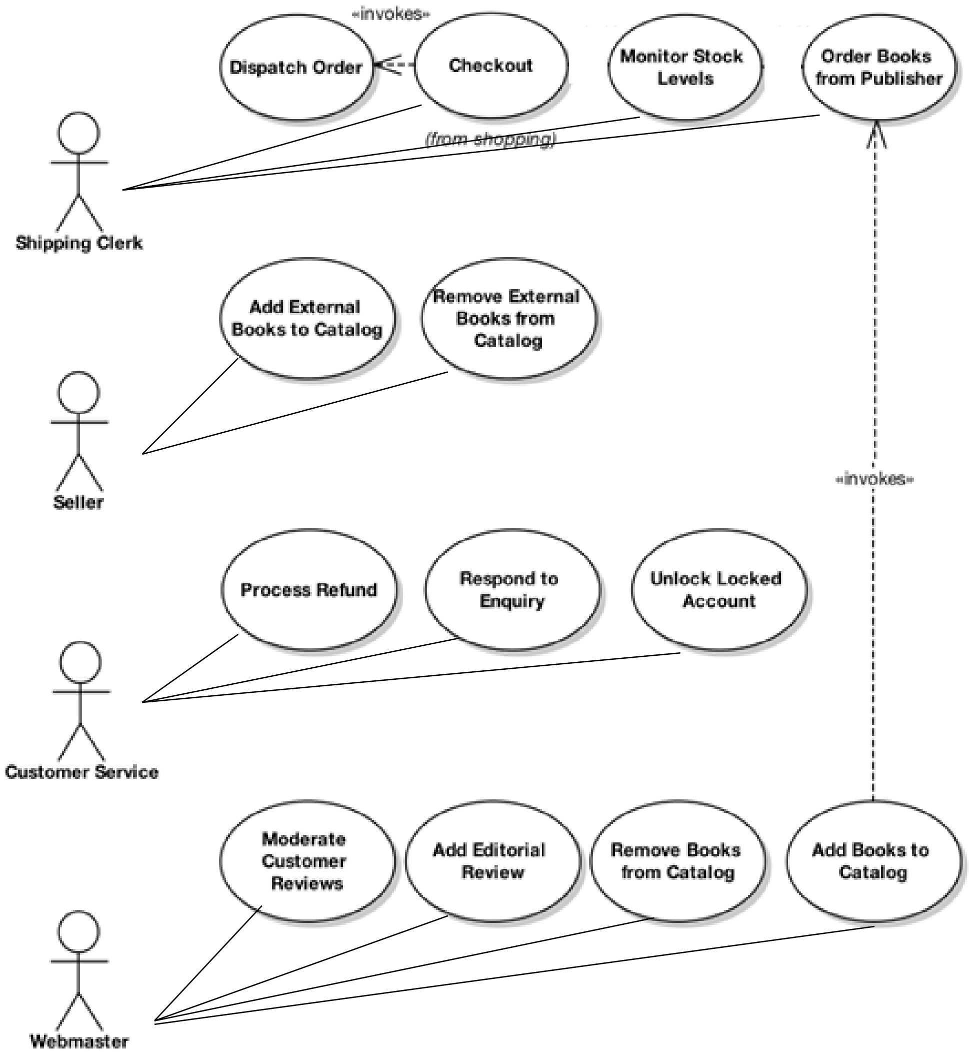
- Example



CASE STUDY: Internet Bookstore



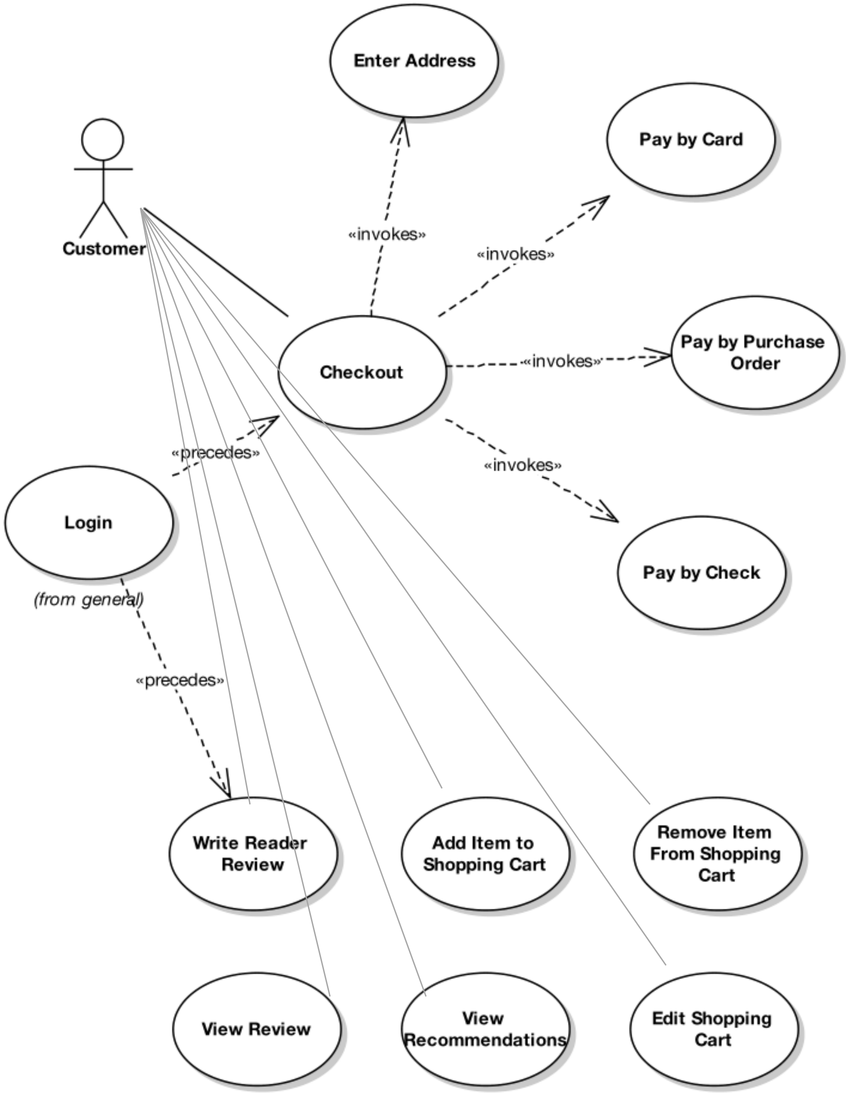
Use case diagram for the “general” package

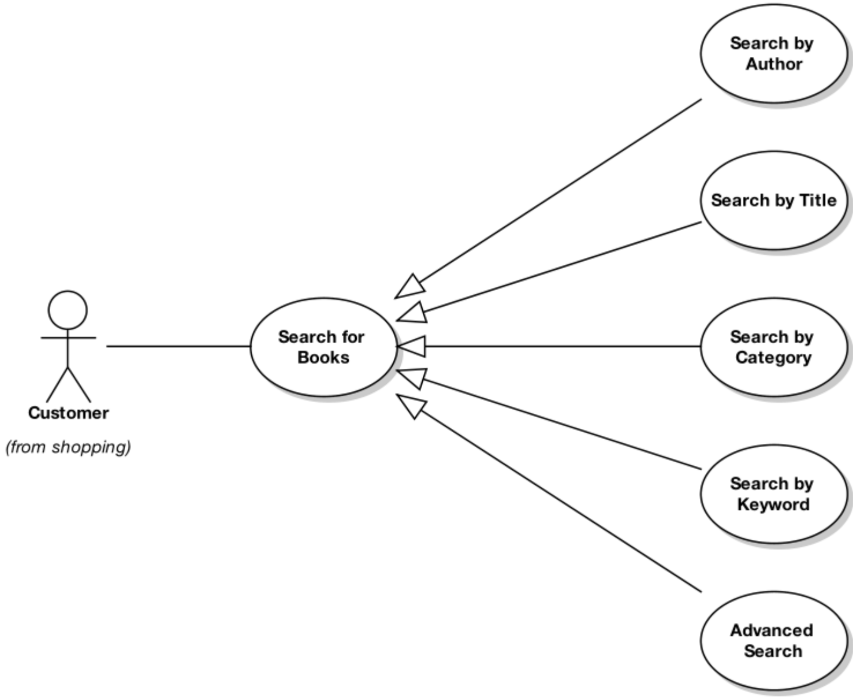


Use case diagram for the “admin” package

Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

CASE STUDY: Internet Bookstore





Use case diagram for the “searching” package

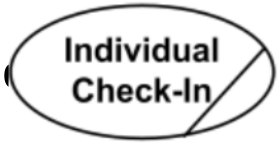
Use case diagram for the “shopping” package

Guidelines for Use-Cases

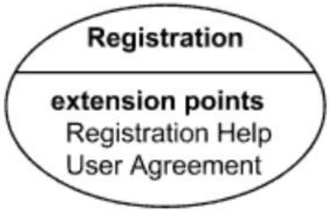
- Use case: functions of the system from the user's point of view
  - For internal functions/programs/algorithms => do not create a use case
- Should not use too many relationships between use case
  - Only use the relationship if necessary
- In the Use case diagram, the order of use cases are not specified
- Each actor involved with at least one use case
- Have all functional requirements been met?

Other Use Case Diagrams Reference

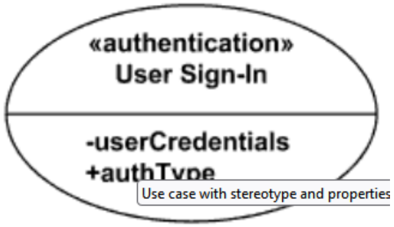
- Business use case
  - to support Business Modeling - to represent business functi process, or activity performed in the modeled **business**.



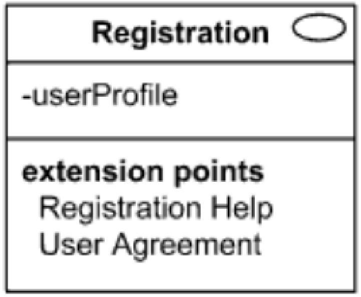
- Use Case with extension points



- Use Case with optional stereotype keyword in «...» and a list of operatior and attributes



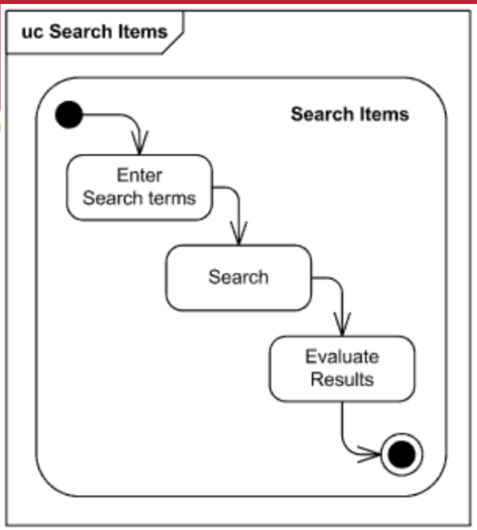
- Use Case shown using the standard rectangle notatio classifiers.



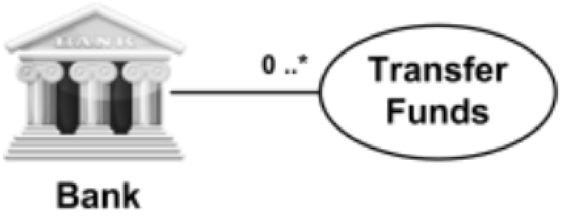
https://www.uml-diagrams.org

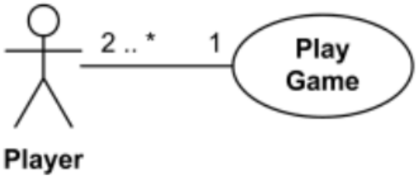
Other Use Case Diagrams Reference

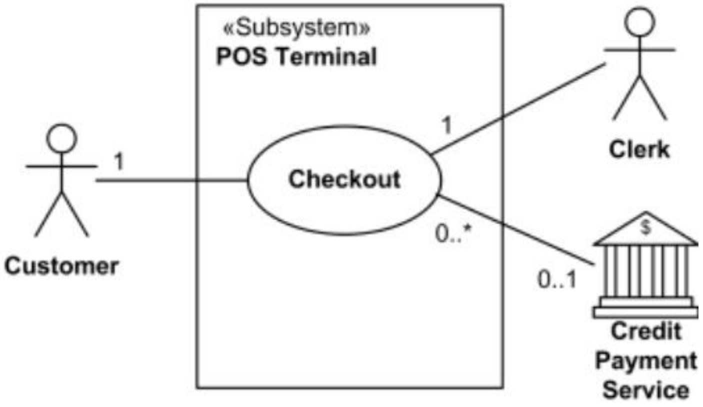
- Use case rendered as frame with associated activity diagram



- Use case with a multiplicity







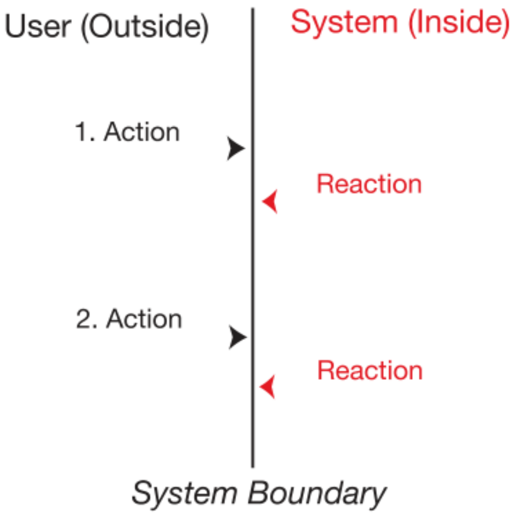
Multiplicity **0..1** of actor means that the actor is not required by any of the associated use cases.

https://www.uml-diagrams.org

5.4 Use-Case description

5.4.1. Use Case Description

- Use case behaviors may be described in a natural language text.
- Use case Description describes
  - What user can do + How the system respond in this use case
  - *basic scenario*: scenarios that assumes everything goes correctly
  - *alternate scenarios*: what happens if something goes wrong ?
- Describe the steps involved in both sides of the ***user/system dialogue***



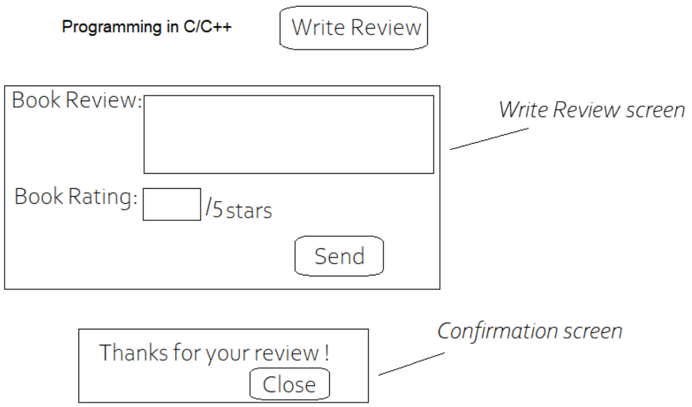
- Note: Keep on updating the domain model as analyzing the use cases

5.4.1. Use Case Description

- Write the use case in the context of the **object model**.
  - need to reference domain model in the previous step
- Should start with "The system displays..."
  - The interaction between user and system takes place via screens, windows, or pages ?
  - Relate to a visual aid: sequence of screens
    - include all the buttons and menus the user can touch to generate events within the use case
    - especially valuable for business system applications and Websites: commonly involve extensive user interactions

CASE STUDY: Internet Bookstore

- *Write Customer Review* use case
- Short description: The user types in a review of the selected item, gives it a score, and sends it. The review is sent to a moderator.
- Visualization by storyboards



Thanks for your review !

- → BASIC SCENARIO: The **Customer** clicks the *Write Review* button for the book currently being viewed, and the system shows the *Write Review screen*. The Customer types in a **Book Review**, gives it a **Book Rating** out of five stars, and clicks the *Send* button. The system ensures that the **Book Review** isn't too long or short, and that the **Book Rating** is within one and five stars. The system then displays a *confirmation screen*, and the review is saved to a **Moderator**, ready to be added**.**

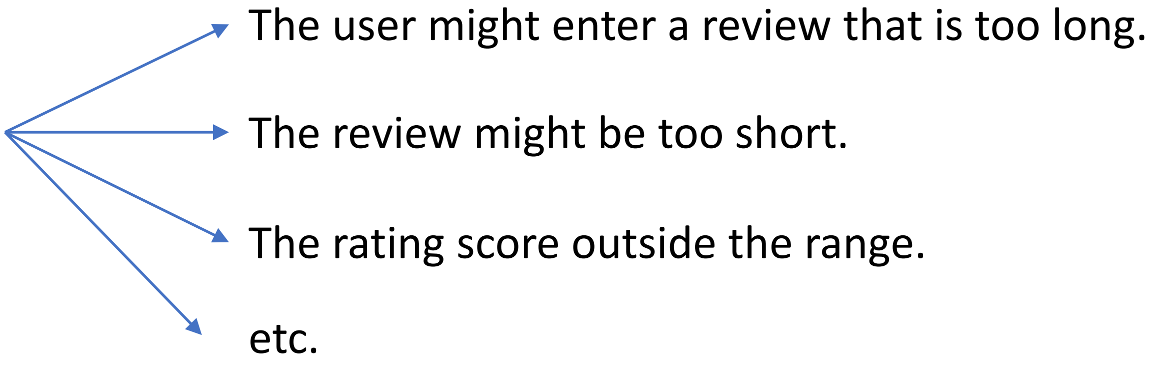
Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

CASE STUDY: Internet Bookstore

- *Write Customer Review* use case
- Extension

  Think as user !

  What else happens?



The user might enter a review that is too long.

The review might be too short.

The rating score outside the range.

etc.

ALTERNATE SCENARIOS :

- User not logged in: The user is taken to the Login screen.
- The user enters a review that is too long (text > 1MB): The system rejects the review and responds with a message explaining why the review was rejected.
- The review is too short (< 10 characters): The system rejects the review and responds with a message explaining why the review was rejected.
- The rating is out side of (0-5): The system rejects the review and responds with a message explaining why the review was rejected.

Use Case Description Guidelines

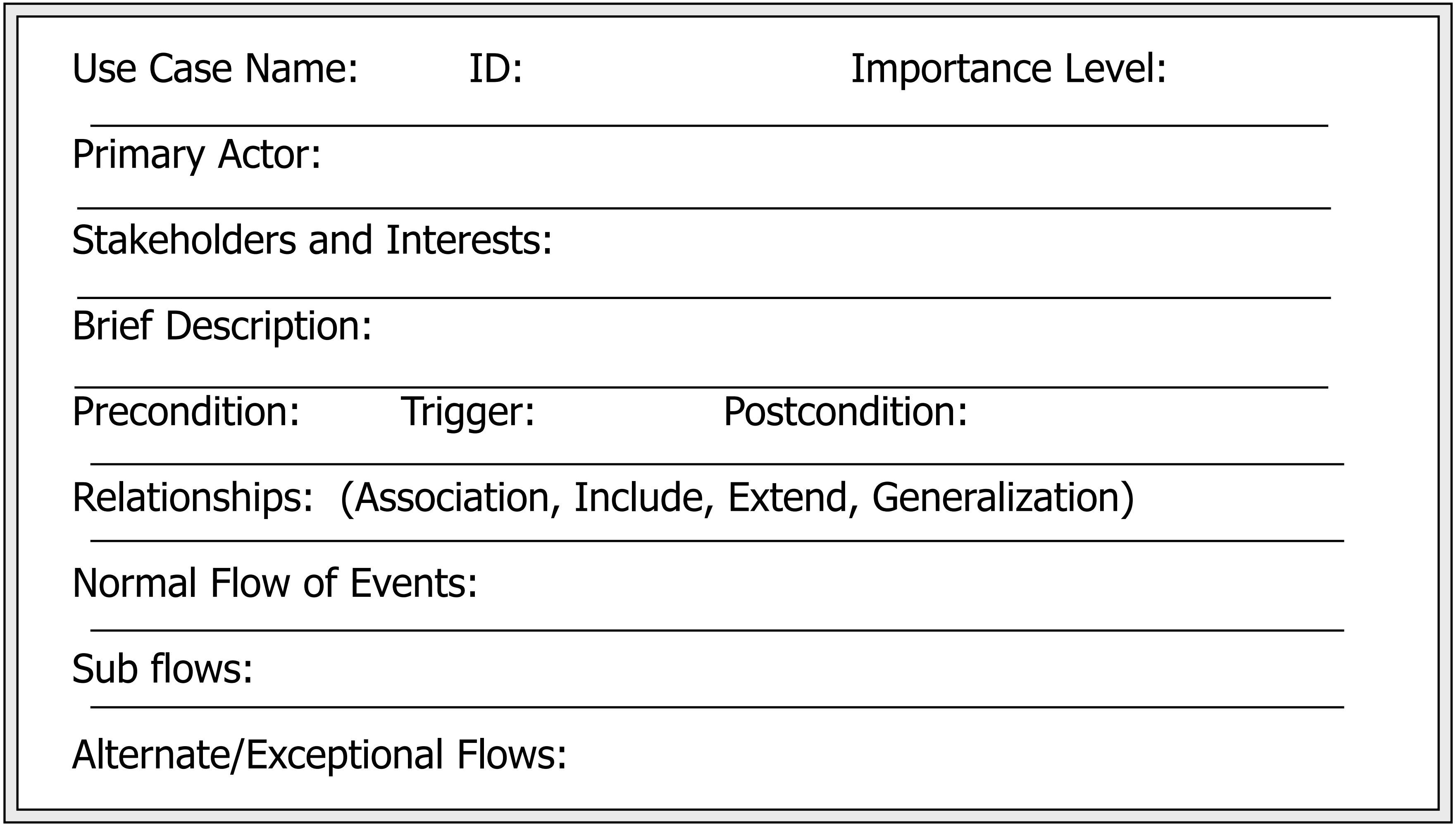
- Write use cases in active voice, using a noun-verb-noun sentence structure.
- Write use case using an event/response flow, describing both sides of the user/system dialogue.
- Write basic scenario and alternate scenarios
- Should not include long lists of functional requirements, just write about how the users will be using the system and what the system will do in response

What happens? ... And then what happens? ... What else might happen?
[If] .... then......... else ....

More important part than draw the diagrams!

5.4.1. Use Case Description

- In Table format



Use Case Name: ID: Importance Level:

Primary Actor:

Stakeholders and Interests:

Brief Description:

Precondition: Trigger: Postcondition:

Relationships: (Association, Include, Extend, Generalization)

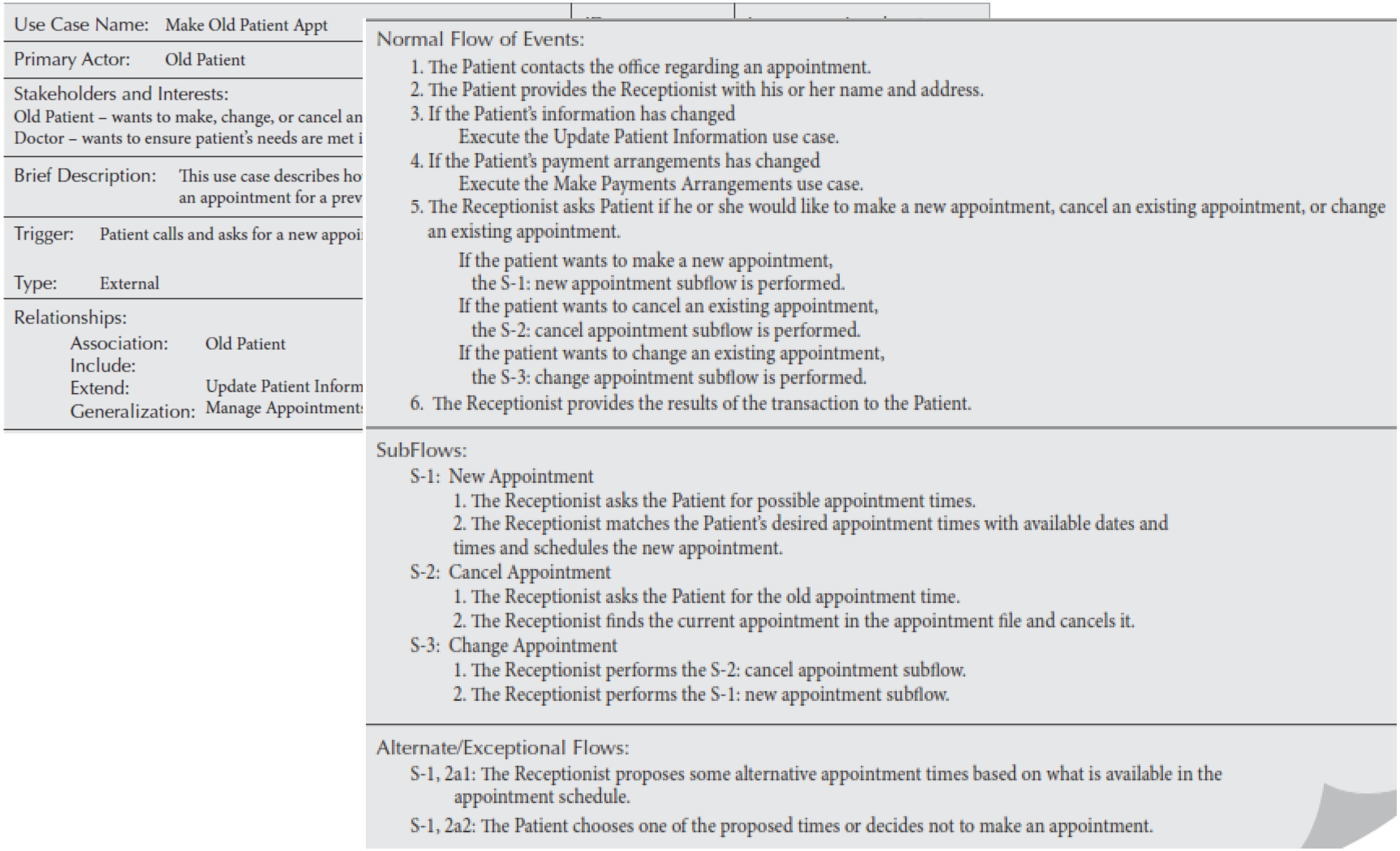
Normal Flow of Events:

Sub flows:

Alternate/Exceptional Flows:

5.4.1. Use Case Description

- Example



Use Case Name: Make Old Patient Appt

Primary Actor: Old Patient

Stakeholders and Interests:
Old Patient – wants to make, change, or cancel an
Doctor – wants to ensure patient's needs are met i

Brief Description: This use case describes ho
an appointment for a prev

Trigger: Patient calls and asks for a new appoi

Type: External

Relationships:
Association: Old Patient
Include:
Extend: Update Patient Inform
Generalization: Manage Appointment

Normal Flow of Events:

1. The Patient contacts the office regarding an appointment.
2. The Patient provides the Receptionist with his or her name and address.
3. If the Patient's information has changed
   Execute the Update Patient Information use case.
4. If the Patient's payment arrangements has changed
   Execute the Make Payments Arrangements use case.
5. The Receptionist asks Patient if he or she would like to make a new appointment, cancel an existing appointment, or change an existing appointment.
   If the patient wants to make a new appointment,
   the S-1: new appointment subflow is performed.
   If the patient wants to cancel an existing appointment,
   the S-2: cancel appointment subflow is performed.
   If the patient wants to change an existing appointment,
   the S-3: change appointment subflow is performed.
6. The Receptionist provides the results of the transaction to the Patient.

SubFlows:

S-1: New Appointment
1. The Receptionist asks the Patient for possible appointment times.
2. The Receptionist matches the Patient's desired appointment times with available dates and times and schedules the new appointment.

S-2: Cancel Appointment
1. The Receptionist asks the Patient for the old appointment time.
2. The Receptionist finds the current appointment in the appointment file and cancels it.

S-3: Change Appointment
1. The Receptionist performs the S-2: cancel appointment subflow.
2. The Receptionist performs the S-1: new appointment subflow.

Alternate/Exceptional Flows:

S-1, 2a1: The Receptionist proposes some alternative appointment times based on what is available in the appointment schedule.

S-1, 2a2: The Patient chooses one of the proposed times or decides not to make an appointment.

Dennis, Wixom, Tegarden, "Systems Analysis & Design: An Object-Oriented Approach with UML", 5th edition, Wiley

Guidelines for Use-Case Description

- Make clear in each step
  - Who/What is the initiator of the action,
  - Who/What is the receiver of the action
  - Is it clear how and when the use case's flow of events starts and ends?
  - Are the actor interactions and exchanged information clear?

- Ensure that the use case contains a sensible set of actions
  1. Primary actor initiates the execution of the UC by *sending a request* to the system
  2. System *ensures* that the request is *valid*
  3. System *processes the request* and possibly *changes its own internal state*
  4. System *sends* the primary actor *the result of the processing*

- If the UC becomes too complex and/or too long, the UC should be decomposed into a set of UCs

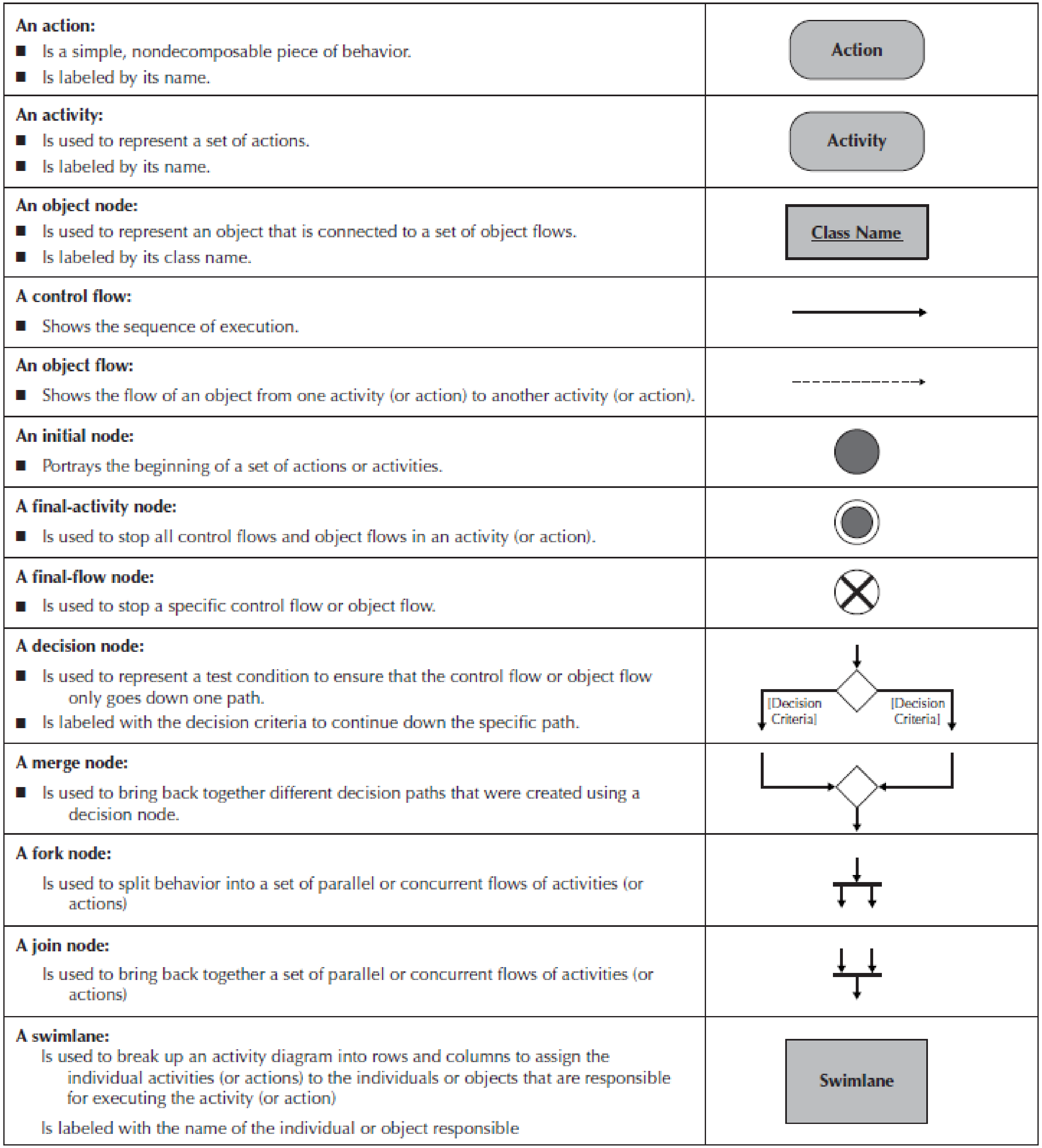
5.5 Activity diagrams

5.5 Activity diagrams

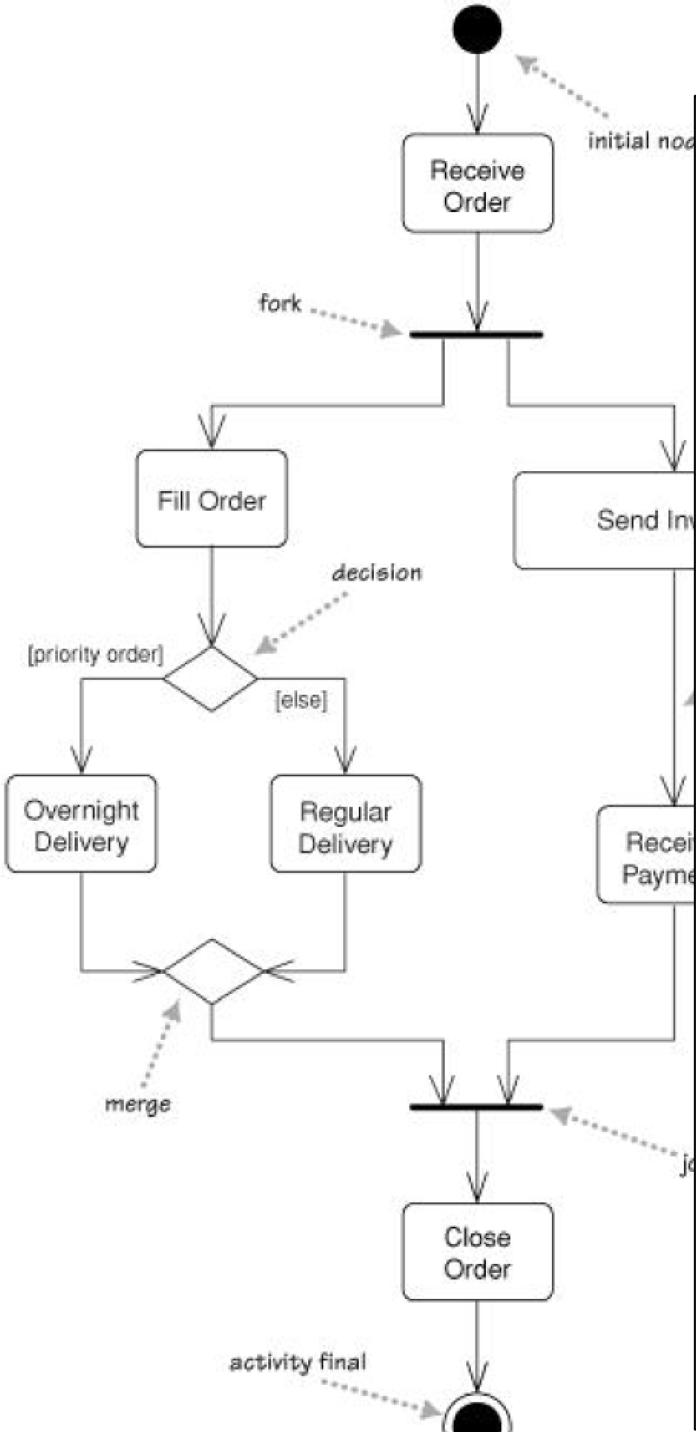
- Use case behaviors may be described using activity diagram
- In basic form, an activity diagram is a simple and intuitive illustration of *what happens in a workflow*
- Activity diagrams vs. Use Case diagrams:
  - Use case Diagram: from user perspective, shows relationship between actors and use cases with *no specific time order*
  - Activity Diagram: define workflow of the system, subsystem or a use case
    - ~ Use case description in diagram format
- Use activity diagrams
  - To illustrate how a business processes / use-case operates
  - To illustrate the flow of events in a use-case model.
    - **flows of controls** or **objects** with emphasis on the **sequence** and **conditions** of the flow

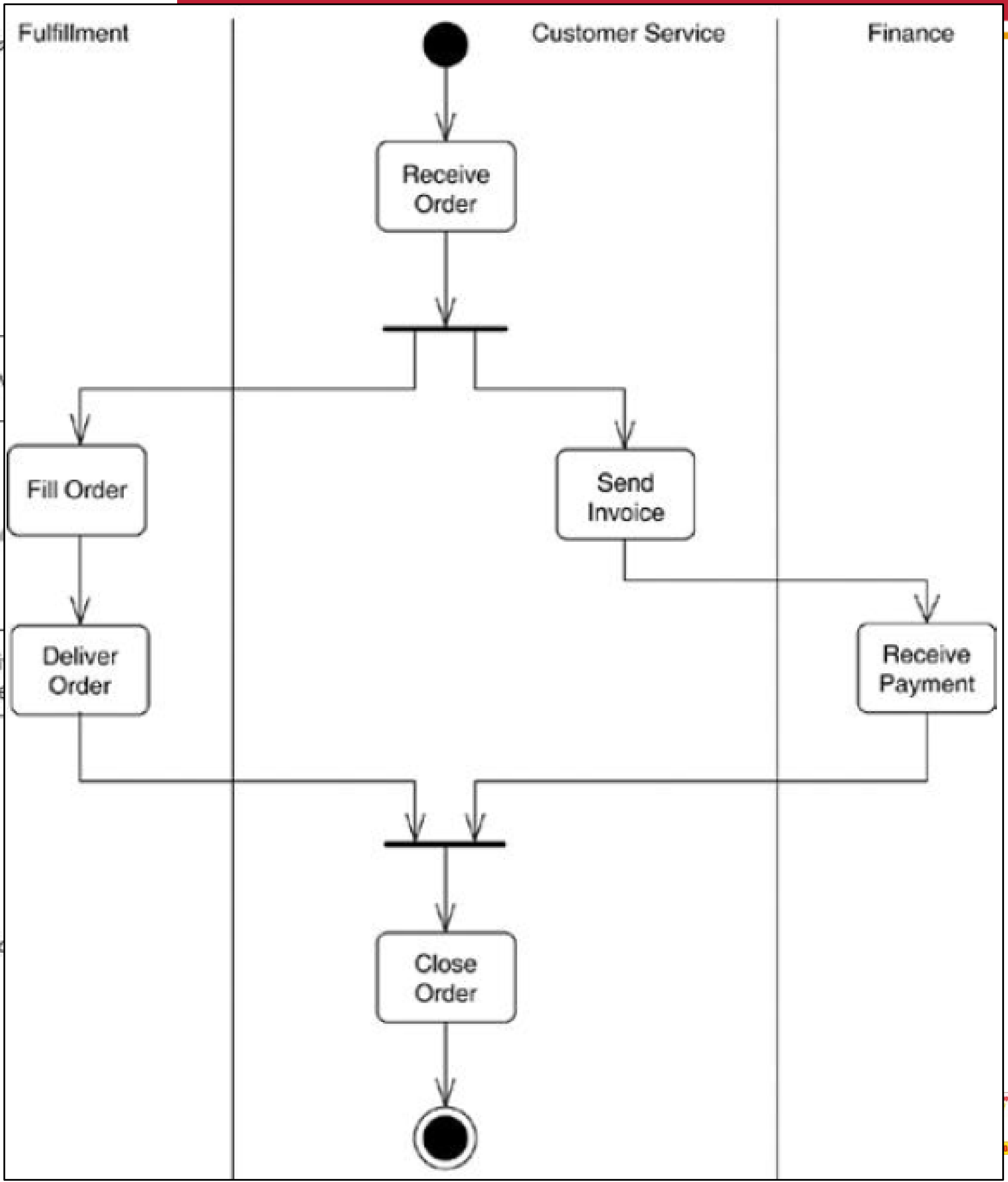
5.5 Activity diagrams

- Elements

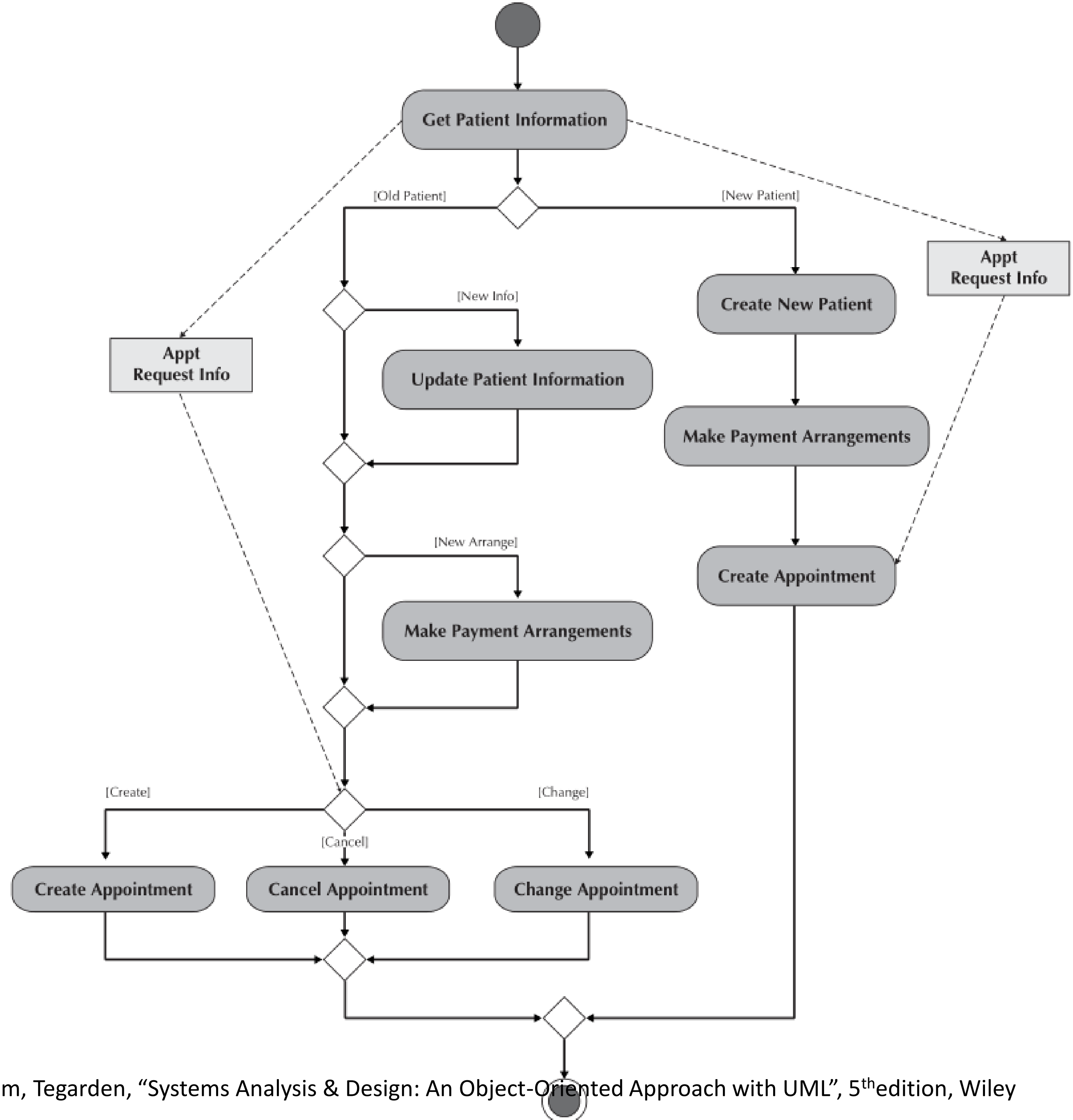


An action: ■ Is a simple, nondecomposable piece of behavior. ■ Is labeled by its name.	Action
An activity: ■ Is used to represent a set of actions. ■ Is labeled by its name.	Activity
An object node: ■ Is used to represent an object that is connected to a set of object flows. ■ Is labeled by its class name.	Class Name
A control flow: ■ Shows the sequence of execution.	⟶
An object flow: ■ Shows the flow of an object from one activity (or action) to another activity (or action).	⇢
An initial node: ■ Portrays the beginning of a set of actions or activities.	●
A final-activity node: ■ Is used to stop all control flows and object flows in an activity (or action).	◉
A final-flow node: ■ Is used to stop a specific control flow or object flow.	⊗
A decision node: ■ Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. ■ Is labeled with the decision criteria to continue down the specific path.	[Decision Criteria] ◇ [Decision Criteria]
A merge node: ■ Is used to bring back together different decision paths that were created using a decision node.	◇
A fork node: Is used to split behavior into a set of parallel or concurrent flows of activities (or actions)	
A join node: Is used to bring back together a set of parallel or concurrent flows of activities (or actions)	
A swimlane: Is used to break up an activity diagram into rows and columns to assign the individual activities (or actions) to the individuals or objects that are responsible for executing the activity (or action) Is labeled with the name of the individual or object responsible	Swimlane


initial nod
Receive Order
fork
Fill Order
Send In
decision
[priority order]
[else]
Overnight Delivery
Regular Delivery
Recei Payme
merge
j
Close Order
activity final


Fulfillment
Customer Service
Finance
Receive Order
Fill Order
Send Invoice
Deliver Order
Receive Payment
Close Order

Activity



Dennis, Wixom, Tegarden, “Systems Analysis & Design: An Object-Oriented Approach with UML”, 5th edition, Wiley

Steps to create Activity Diagrams for each Use case

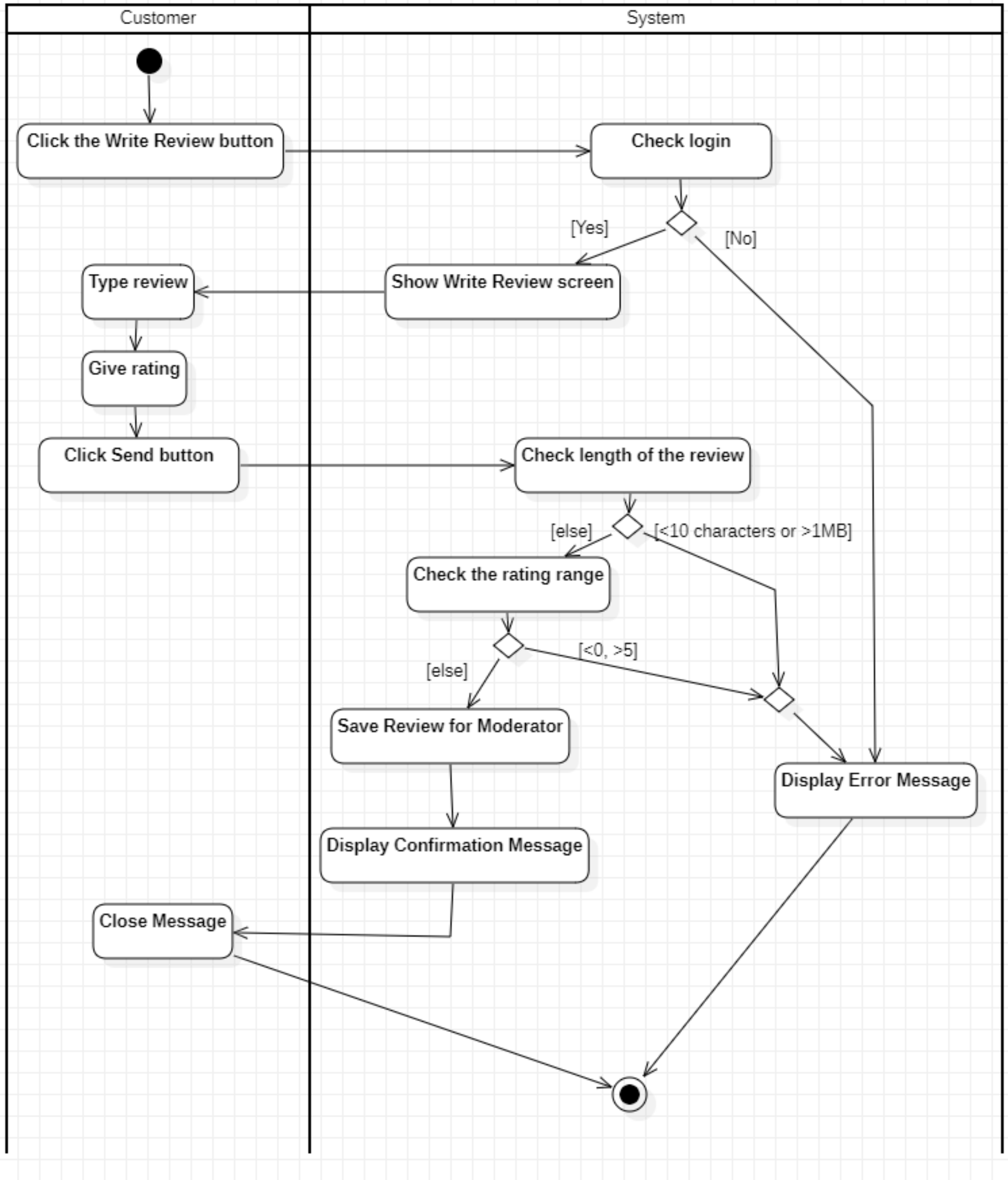
1. Choose a Business Process/Use case
2. Identify Activities
3. Identify Control Flows & Nodes
4. Identify Object Flows & Nodes
5. Lay Out & Draw Diagram

Activity Diagram Guidelines

- Should set the object of the activity being modeled in appropriate title in the domain model
- Must identify the activities, control flows, object flows that occur between the activities
- Should identify any decisions that are part of the process being modeled (important decisions)
- Should attempt to identify any prospects for parallelism in the process
- Line crossings should be minimized
- Swim lanes: should be used only to simplify the understanding of an activity diagram

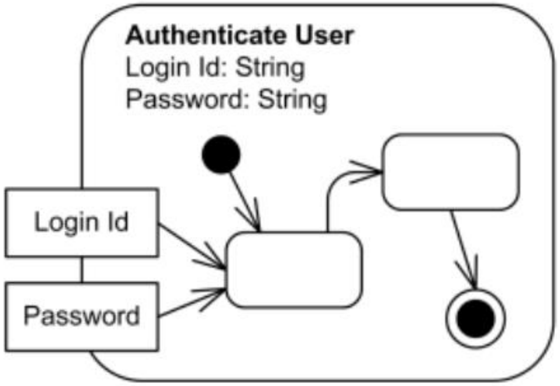
CASE STUDY: Internet Bookstore

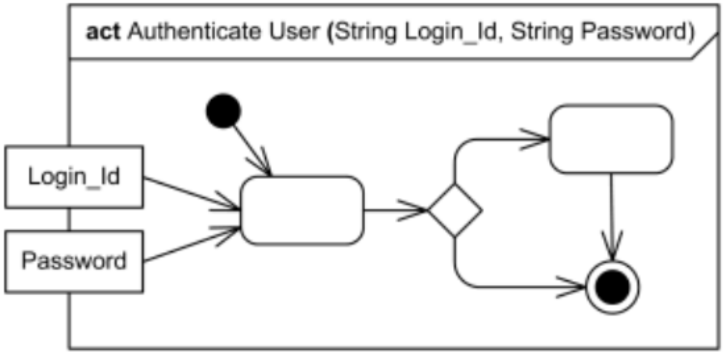
- *Write Customer Review*

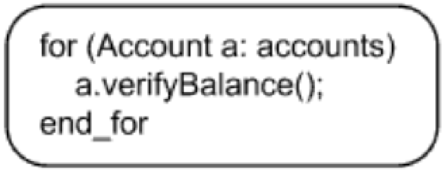


Other Activity Diagrams Reference

- Activity with parameters
  - parameter-name: parameter-type.
- Activity Frame with parameters



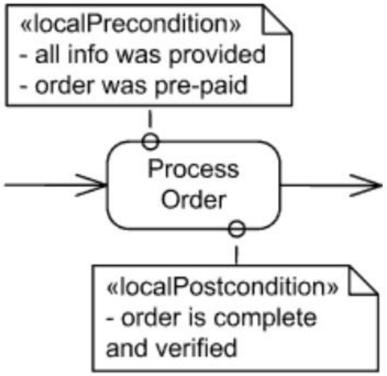




```
for (Account a: accounts)
   a.verifyBalance();
end_for
```

- Action expressed in some application-depend action language.

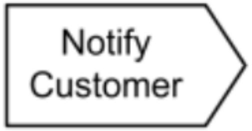
- Local pre- and post-conditions attached to acti



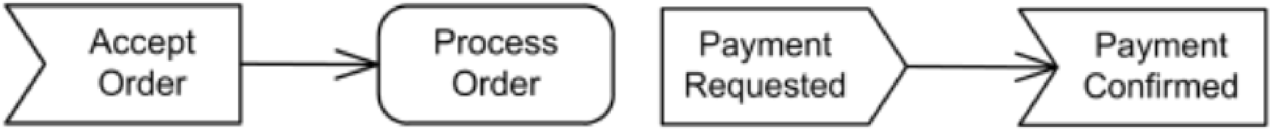
https://www.uml-diagrams.org

Other Activity Diagrams Reference

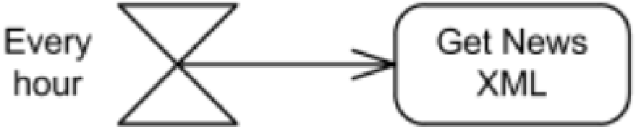
- Send Signal Action



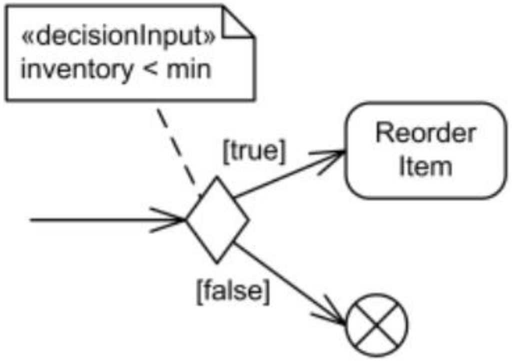
- Accept event action



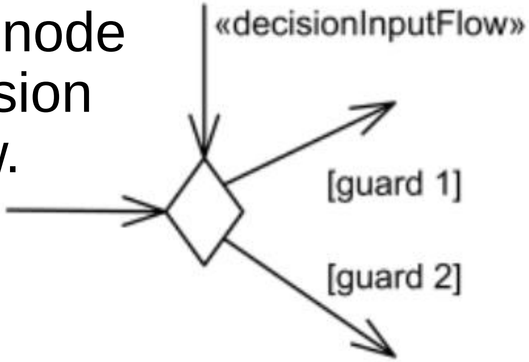
- Wait Time Action



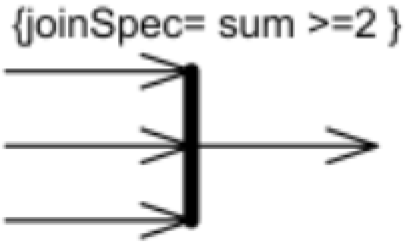
- Decision node with decision input behavior



- Decision node with decision input flow.

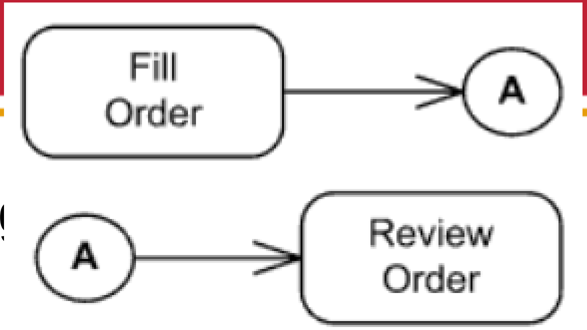


- Join node with join specification



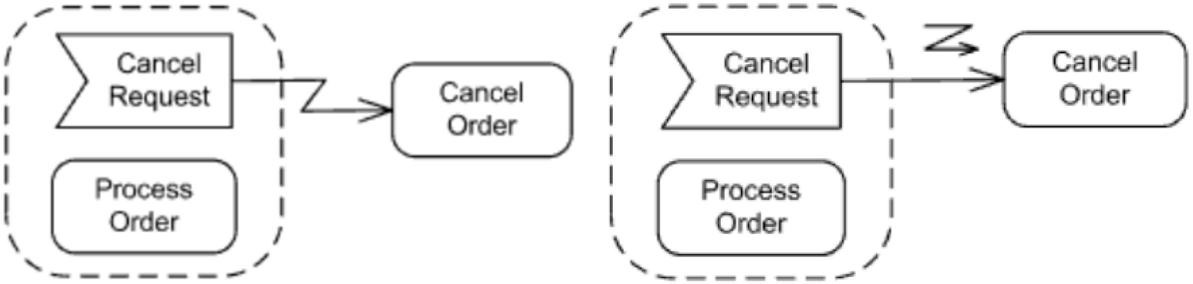
https://www.uml-diagrams.org

Other Activity Diagrams Reference

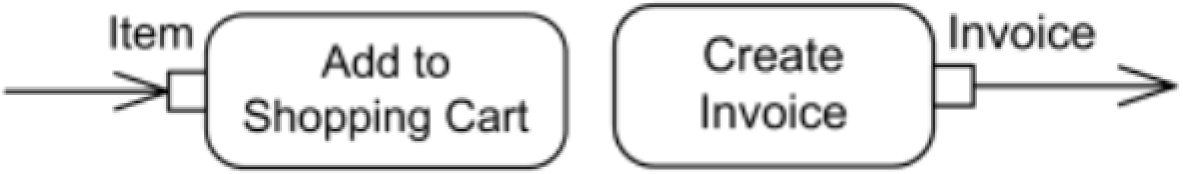


- Connectors are generally used to avoid drawing a long edge.

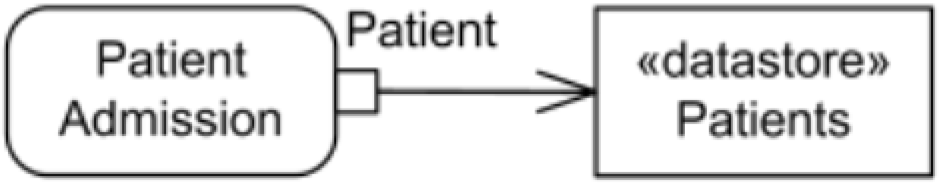
- Interrupting Edge



- A pin as an object node for inputs and outputs to actions.



- Data Store



https://www.uml-diagrams.org

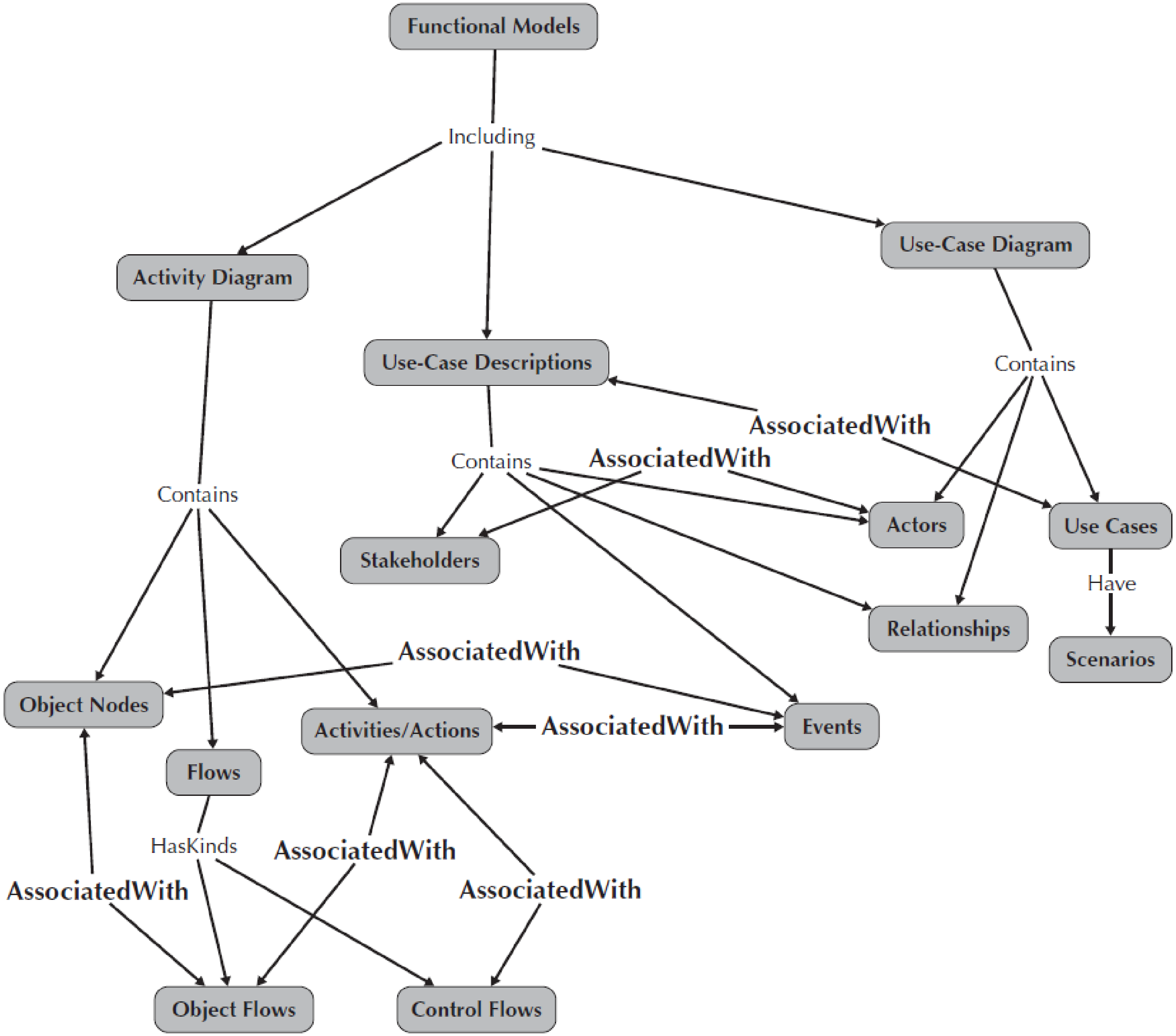
5.6 Functional Model Verification and Validatio

Functional Model Verification and Validation

1. Events recorded in the flows of the *use-case description* need to be included on *activity diagram*.
2. All objects portrayed as an object node in an *activity diagram* must be mentioned in an event in the flows of the *use-case description*.
3. Sequential order of the events in a *use-case description* should occur in the same sequential order of the *activities diagram*.
4. There must be one and only one use-case description for each *use case*, and vice-versa.
5. All actors listed in a use-case description must be portrayed on the use-case diagram
6. Each actor must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description.
7. Stakeholders listed in the use-case description as actors in the use-case diagram.
8. All relationships (include, extend, generalization) listed in a use-case description must be portrayed on a use-case diagram.
9. Diagram-specific requirements that must be enforced.
10. Make sure that the use case text matches the customer’s expectations.

Functional Model Verification and Validation

- Inter-relationship among Function Models

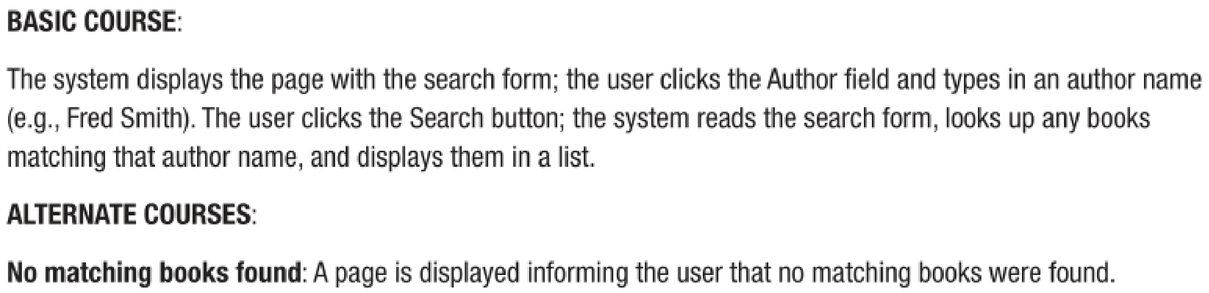


Dennis, Wixom, Tegarden, “Systems Analysis & Design: An Object-Oriented Approach with UML”, 5thedition, Wiley

Quiz

Find and Fix problem of the following use cases

1. Search by Author



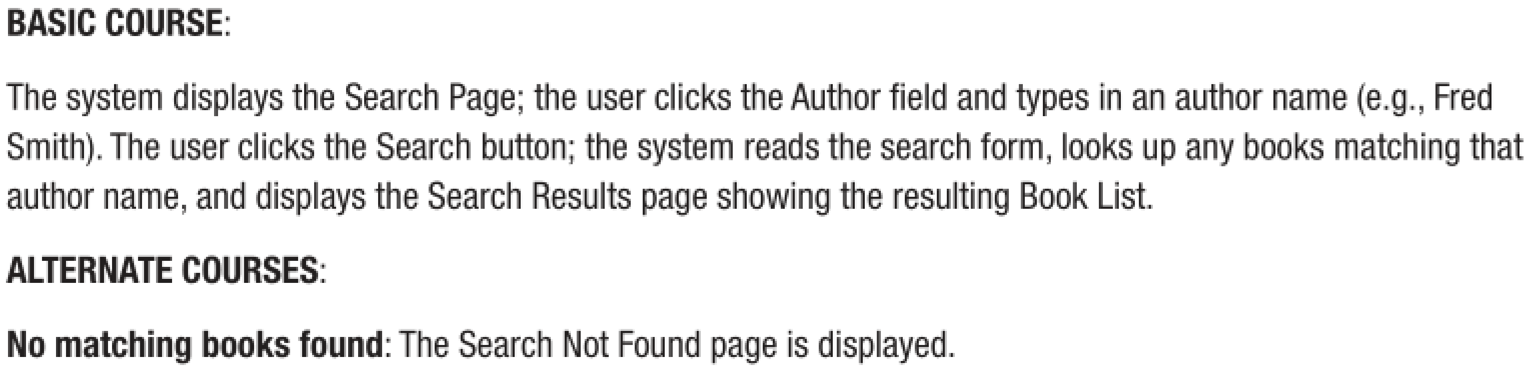
BASIC COURSE:

The system displays the page with the search form; the user clicks the Author field and types in an author name (e.g., Fred Smith). The user clicks the Search button; the system reads the search form, looks up any books matching that author name, and displays them in a list.

ALTERNATE COURSES:

No matching books found: A page is displayed informing the user that no matching books were found.

Need explicit boundary object names



BASIC COURSE:

The system displays the Search Page; the user clicks the Author field and types in an author name (e.g., Fred Smith). The user clicks the Search button; the system reads the search form, looks up any books matching that author name, and displays the Search Results page showing the resulting Book List.

ALTERNATE COURSES:

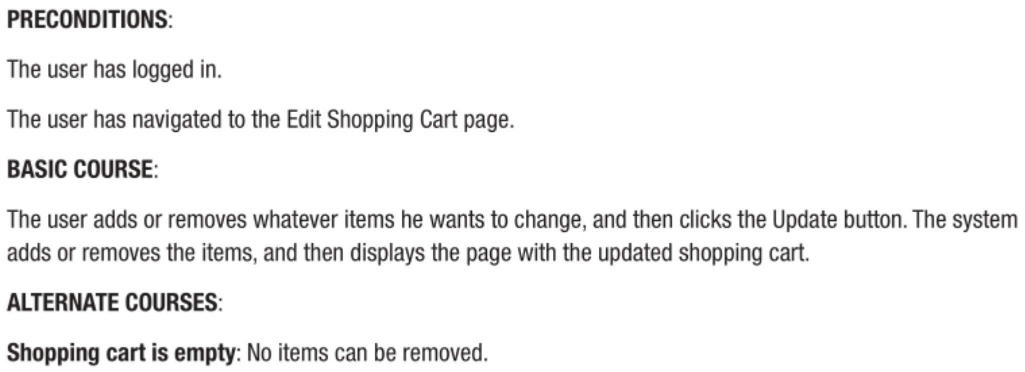
No matching books found: The Search Not Found page is displayed.

Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

Quiz

Find and Fix problem of the following use cases

2. Edit Shopping Cart



PRECONDITIONS:

The user has logged in.

The user has navigated to the Edit Shopping Cart page.

BASIC COURSE:

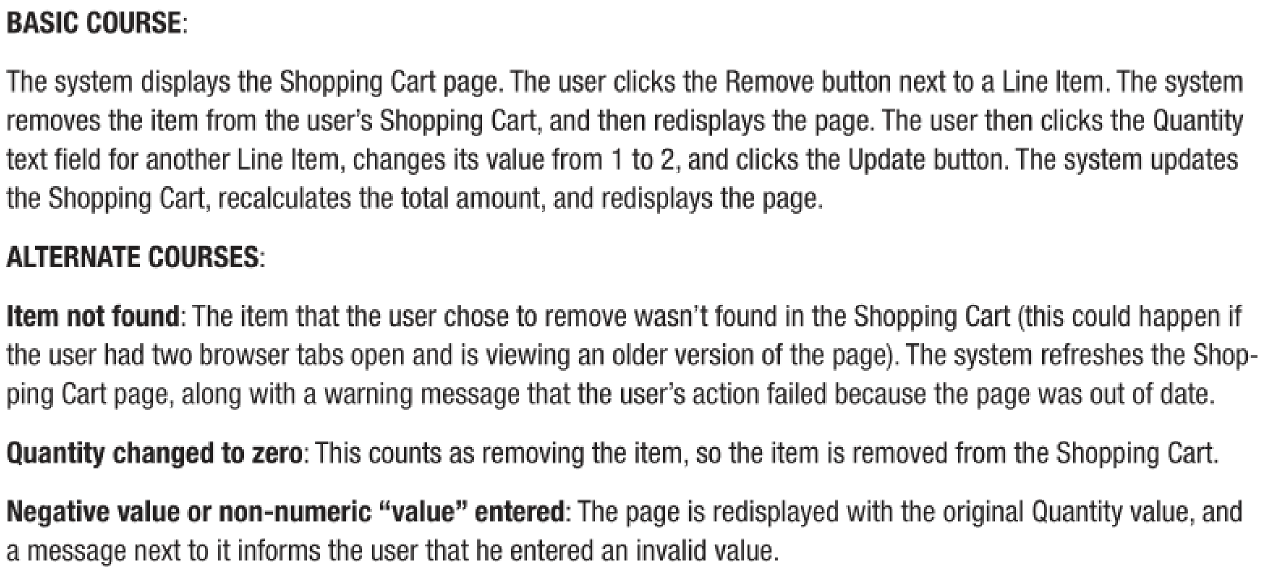
The user adds or removes whatever items he wants to change, and then clicks the Update button. The system adds or removes the items, and then displays the page with the updated shopping cart.

ALTERNATE COURSES:

Shopping cart is empty: No items can be removed.

Vague and Ambiguous

+ Initial “display” action is missed (instead of Preconditions)
+ In the basic course, no specific scenario, but instead tries to cover them all
+ Alternate course doesn’t tie into any particular action in the use case text.
+ Miss other alternate courses



BASIC COURSE:

The system displays the Shopping Cart page. The user clicks the Remove button next to a Line Item. The system removes the item from the user’s Shopping Cart, and then redisplays the page. The user then clicks the Quantity text field for another Line Item, changes its value from 1 to 2, and clicks the Update button. The system updates the Shopping Cart, recalculates the total amount, and redisplays the page.

ALTERNATE COURSES:

Item not found: The item that the user chose to remove wasn’t found in the Shopping Cart (this could happen if the user had two browser tabs open and is viewing an older version of the page). The system refreshes the Shopping Cart page, along with a warning message that the user’s action failed because the page was out of date.

Quantity changed to zero: This counts as removing the item, so the item is removed from the Shopping Cart.

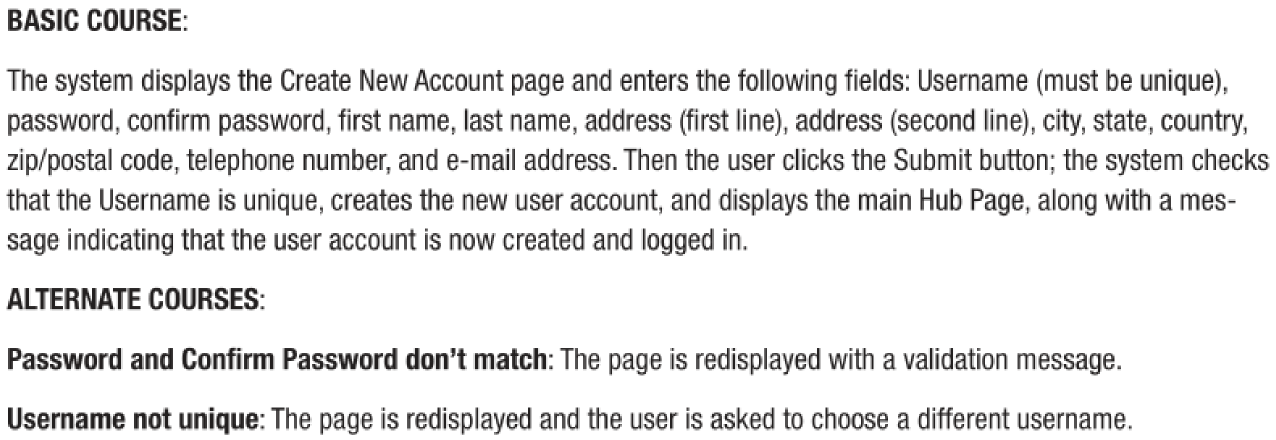
Negative value or non-numeric “value” entered: The page is redisplayed with the original Quantity value, and a message next to it informs the user that he entered an invalid value.

Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

Quiz

Find and Fix problem of the following use cases

3. Open an Account



BASIC COURSE:

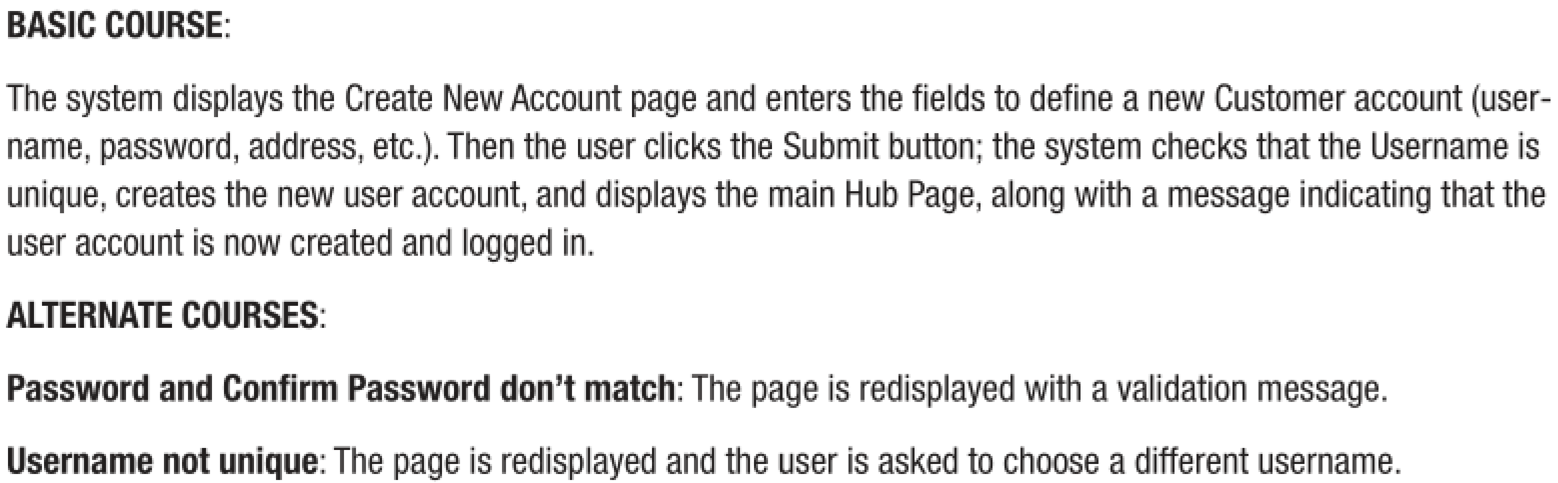
The system displays the Create New Account page and enters the following fields: Username (must be unique), password, confirm password, first name, last name, address (first line), address (second line), city, state, country, zip/postal code, telephone number, and e-mail address. Then the user clicks the Submit button; the system checks that the Username is unique, creates the new user account, and displays the main Hub Page, along with a message indicating that the user account is now created and logged in.

ALTERNATE COURSES:

Password and Confirm Password don't match: The page is redisplayed with a validation message.

Username not unique: The page is redisplayed and the user is asked to choose a different username.

Too Many Presentation Details



BASIC COURSE:

The system displays the Create New Account page and enters the fields to define a new Customer account (username, password, address, etc.). Then the user clicks the Submit button; the system checks that the Username is unique, creates the new user account, and displays the main Hub Page, along with a message indicating that the user account is now created and logged in.

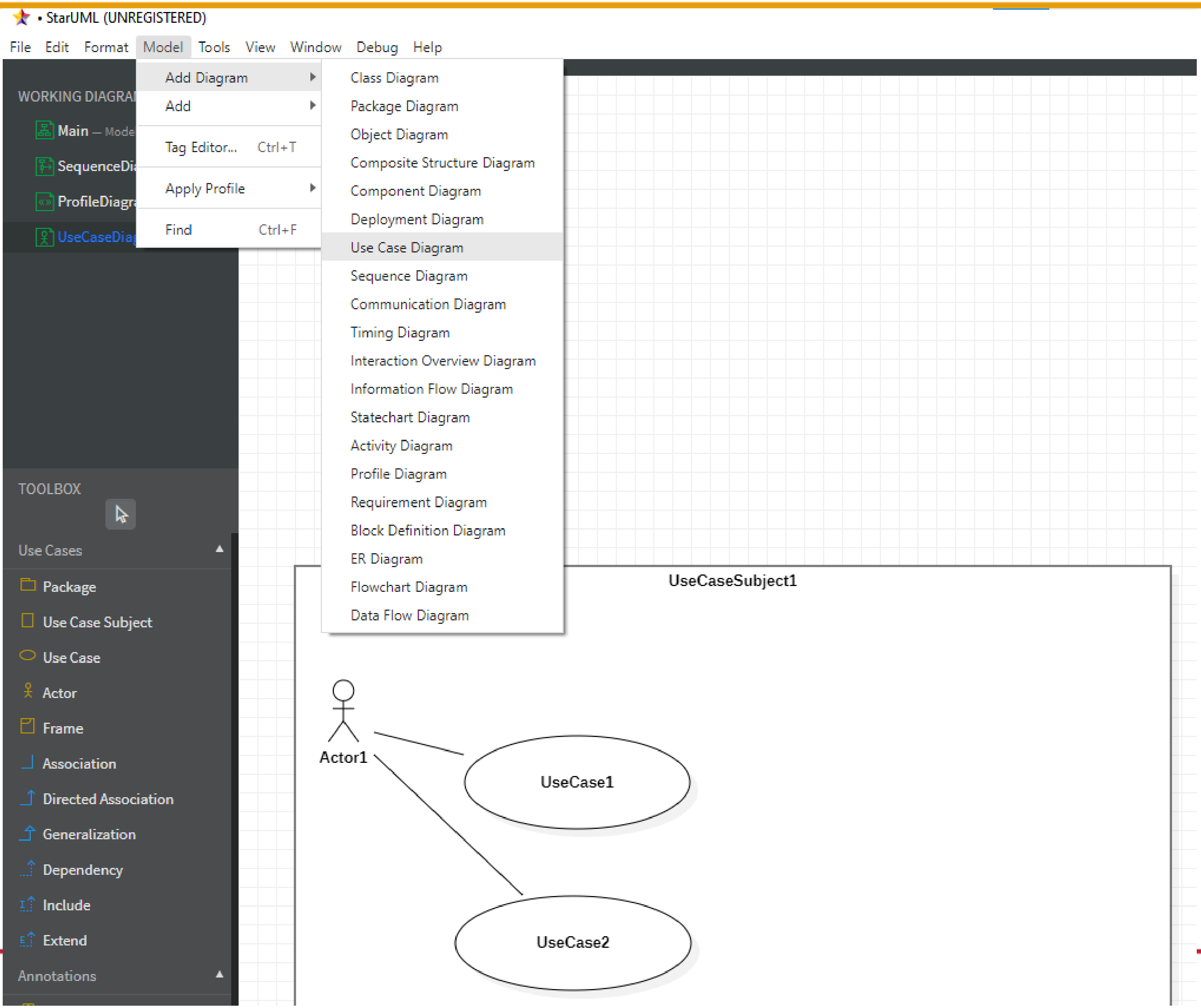
ALTERNATE COURSES:

Password and Confirm Password don't match: The page is redisplayed with a validation message.

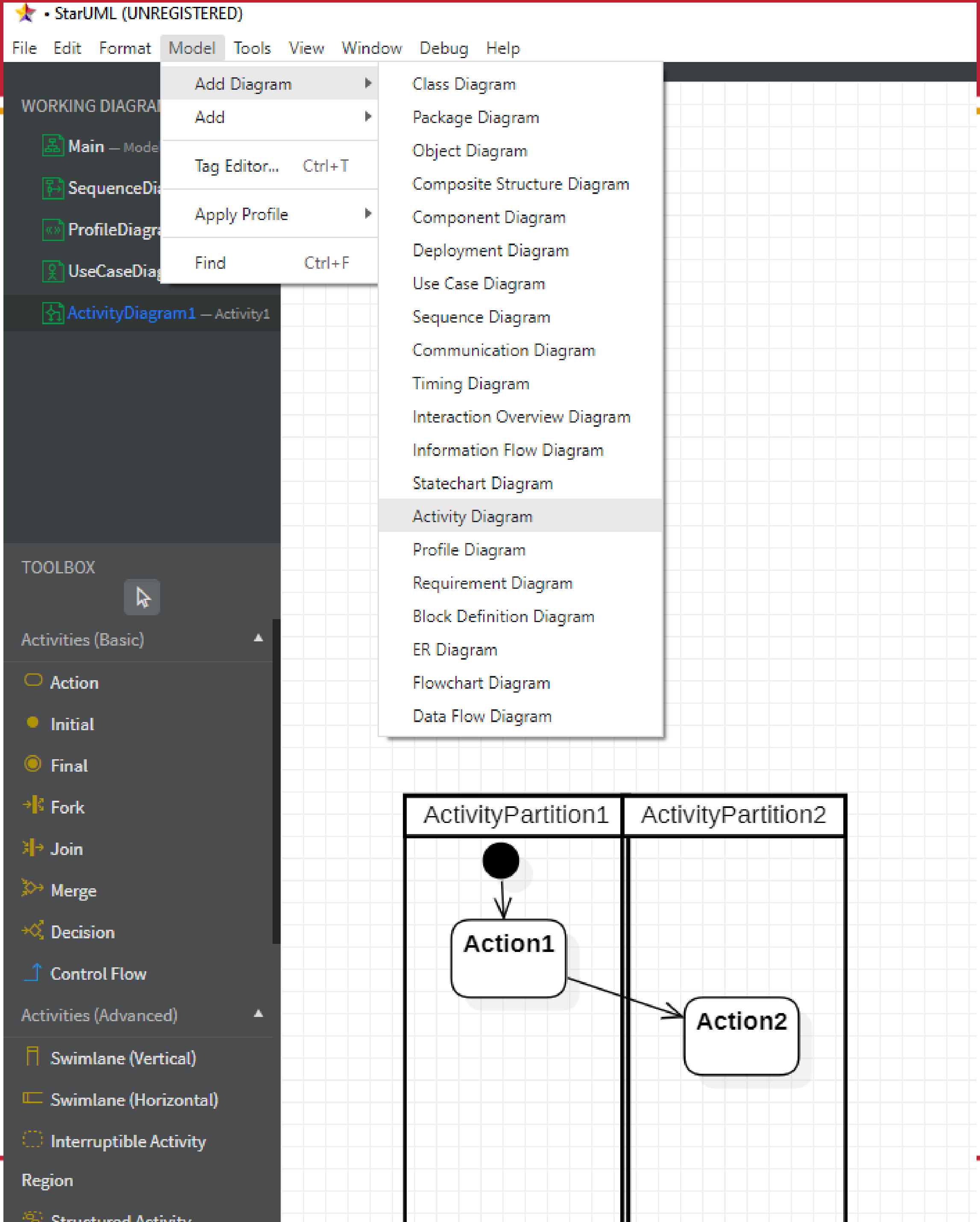
Username not unique: The page is redisplayed and the user is asked to choose a different username.

Doug Rosenberg and Matt Stephens, 2007, *Use Case Driven Object Modeling with UML*, Theory and Practice, Apress

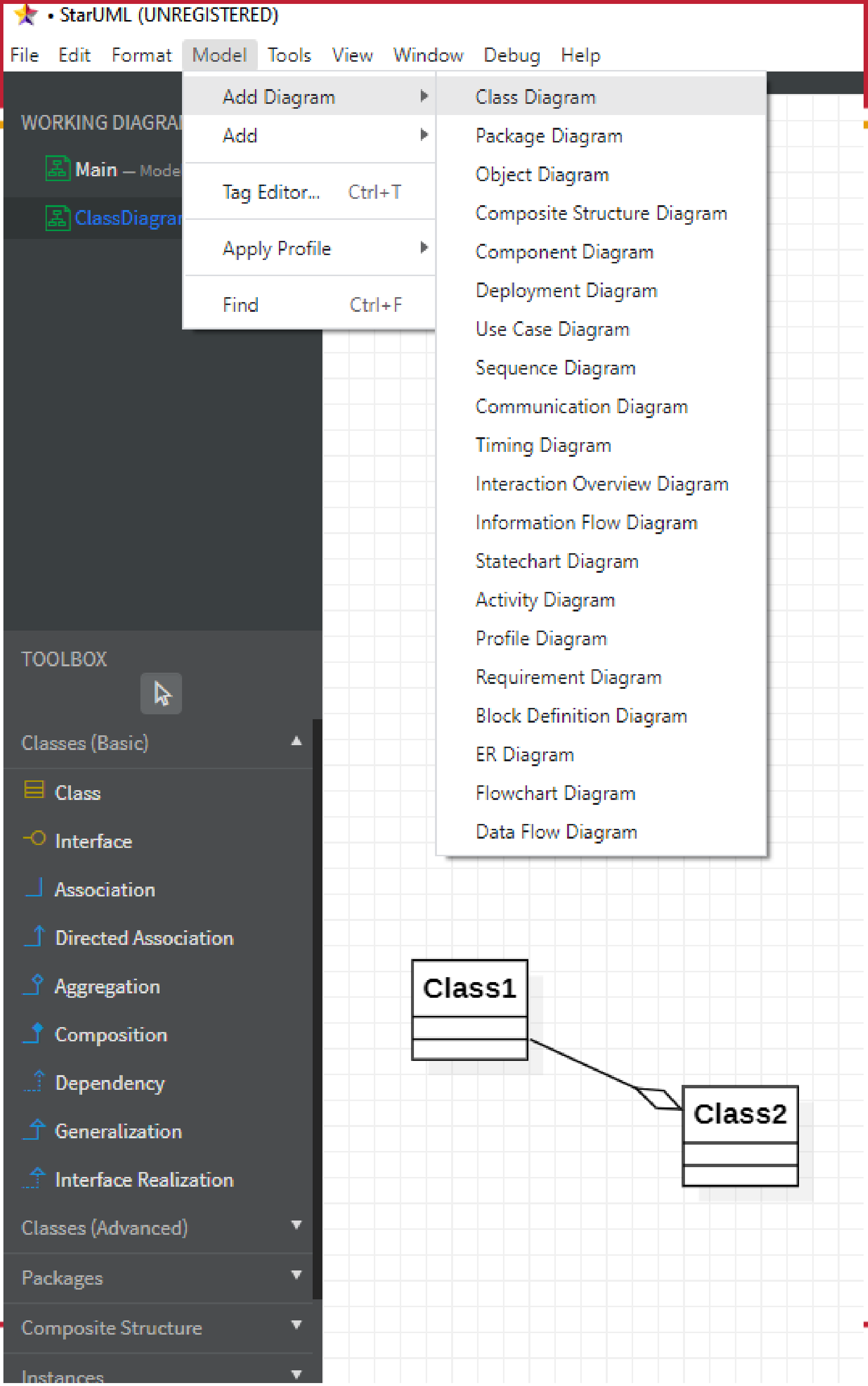
StarUML


• StarUML (UNREGISTERED)
File
Edit
Format
Model
Tools
View
Window
Debug
Help
WORKING DIAGRAI
Main — Mode
SequenceDi
ProfileDiagr
UseCaseDia
Add Diagram
Add
Tag Editor... Ctrl+T
Apply Profile
Find Ctrl+F
Class Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Component Diagram
Deployment Diagram
Use Case Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
Information Flow Diagram
Statechart Diagram
Activity Diagram
Profile Diagram
Requirement Diagram
Block Definition Diagram
ER Diagram
Flowchart Diagram
Data Flow Diagram
TOOLBOX
Use Cases
Package
Use Case Subject
Use Case
Actor
Frame
Association
Directed Association
Generalization
Dependency
Include
Extend
Annotations
UseCaseSubject1
Actor1
UseCase1
UseCase2

StarUML


• StarUML (UNREGISTERED)
File
Edit
Format
Model
Tools
View
Window
Debug
Help
WORKING DIAGRA
Main — Mode
SequenceDi
ProfileDiagr
UseCaseDia
ActivityDiagram1 — Activity1
Add Diagram
Add
Tag Editor... Ctrl+T
Apply Profile
Find Ctrl+F
Class Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Component Diagram
Deployment Diagram
Use Case Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
Information Flow Diagram
Statechart Diagram
Activity Diagram
Profile Diagram
Requirement Diagram
Block Definition Diagram
ER Diagram
Flowchart Diagram
Data Flow Diagram
TOOLBOX
Activities (Basic)
Action
Initial
Final
Fork
Join
Merge
Decision
Control Flow
Activities (Advanced)
Swimlane (Vertical)
Swimlane (Horizontal)
Interruptible Activity
Region
Structured Activity
ActivityPartition1
ActivityPartition2
Action1
Action2

StarUML


• StarUML (UNREGISTERED)
File
Edit
Format
Model
Tools
View
Window
Debug
Help
WORKING DIAGRA
Main — Mode
ClassDiagra
Add Diagram
Add
Tag Editor... Ctrl+T
Apply Profile
Find Ctrl+F
Class Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Component Diagram
Deployment Diagram
Use Case Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
Information Flow Diagram
Statechart Diagram
Activity Diagram
Profile Diagram
Requirement Diagram
Block Definition Diagram
ER Diagram
Flowchart Diagram
Data Flow Diagram
TOOLBOX
Classes (Basic)
Class
Interface
Association
Directed Association
Aggregation
Composition
Dependency
Generalization
Interface Realization
Classes (Advanced)
Packages
Composite Structure
Instances
Class1
Class2

Exercise

- Create Use case diagrams, Use case descriptions, Activity diagrams for your Project