UML for analysis

11/3/24

Outline

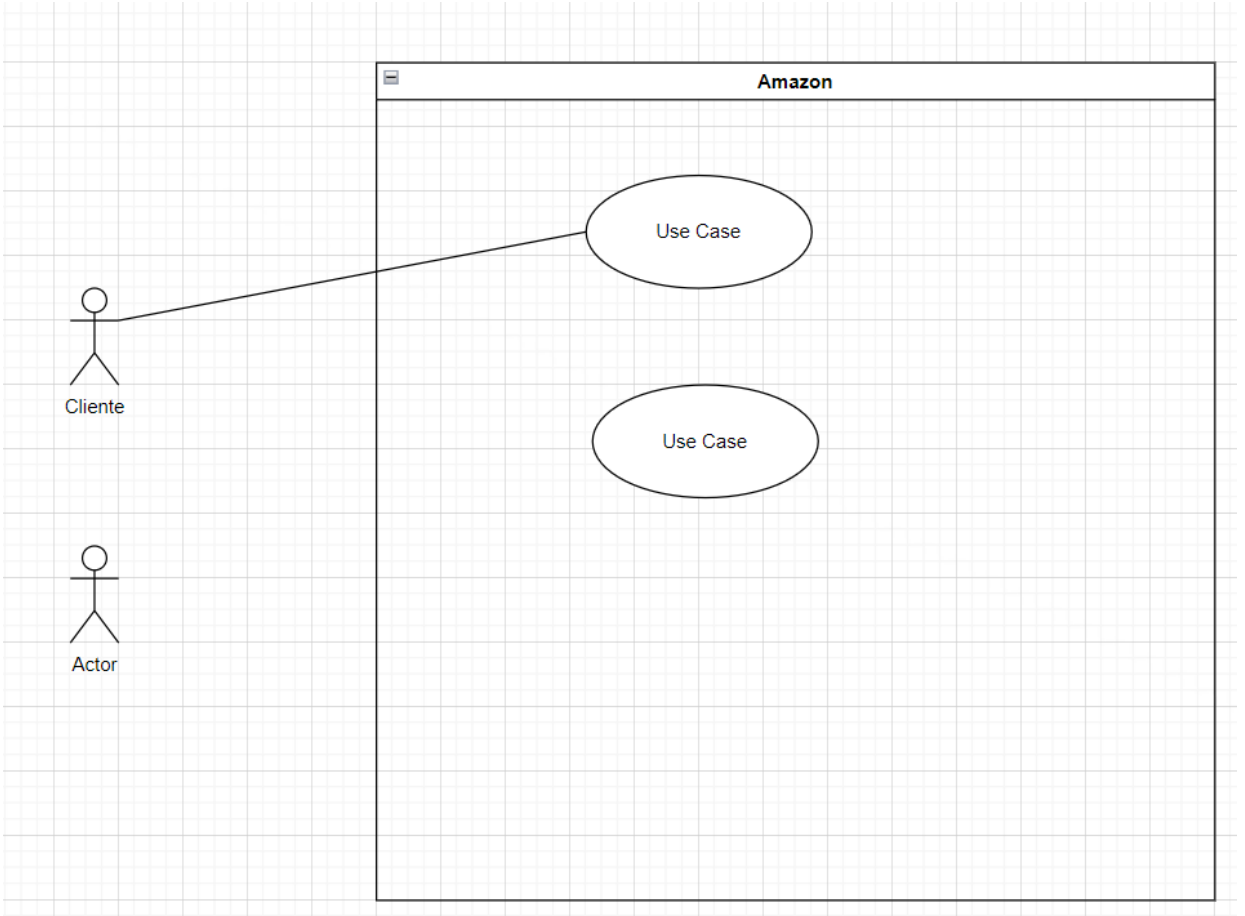
- UML for analysis
  - Use case diagrams
  - System Sequence Diagrams
  - Domain models
  - State diagrams
  - Activity diagrams

Use case diagrams

Use case diagrams

- Basic notation
- Relationship notation
- Advanced notation
  - Extension points in extends relationships
  - CRUD

Basic notation

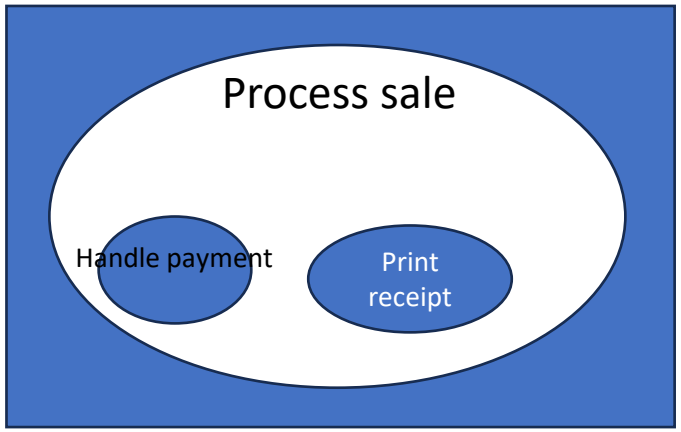


Relationship notation

Relationships

- Include
- Extend
- Generalization-specialization

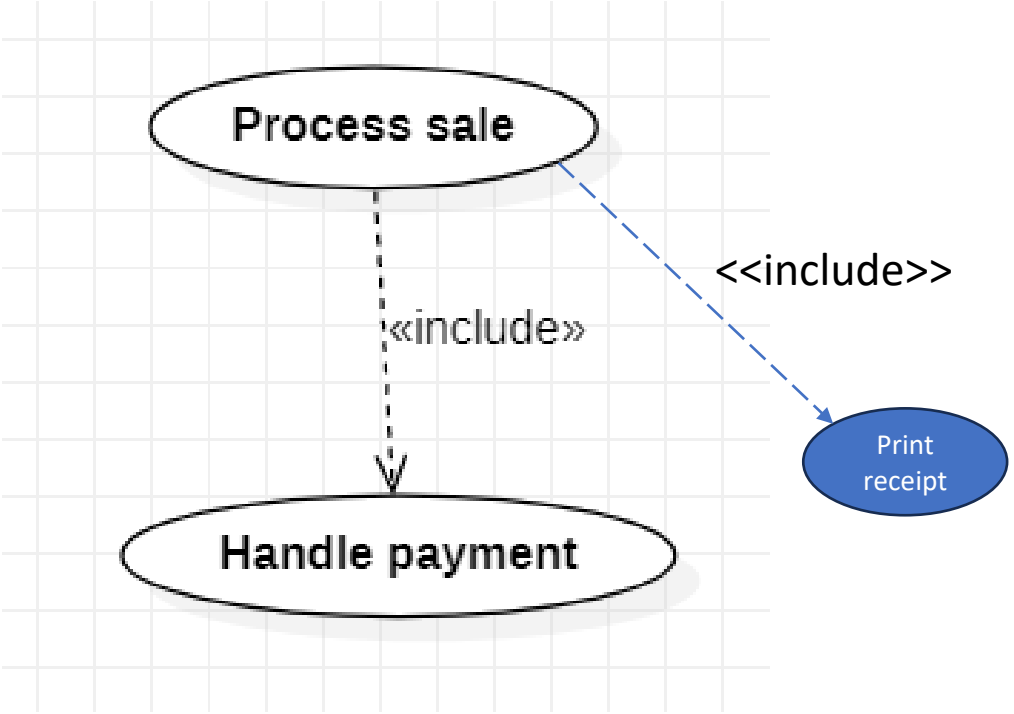
“*include*” relationship



- It is used to represent composition:
  - A use case *includes* of one or more sub use cases

Base use case (super process)

Addition use case (sub process)



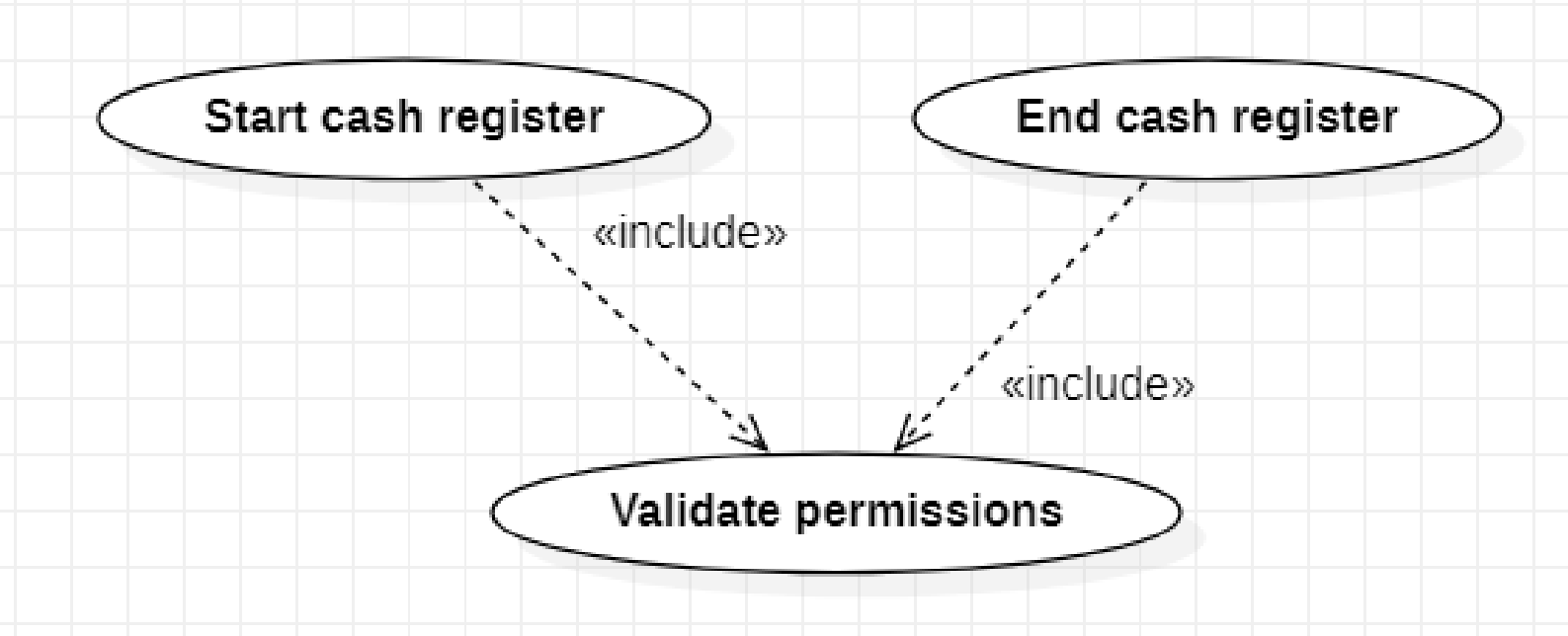
El caso de uso base relacionado con include es dependiente de sus sub casos de uso

When to use “*include*” relationship?

- 1. A set of steps appear in several use cases
  - You want to avoid repeating those steps
- 2. A use case is too complex

1. A shared sub use case

- Factor out the common steps into a sub process



2. Complex use cases

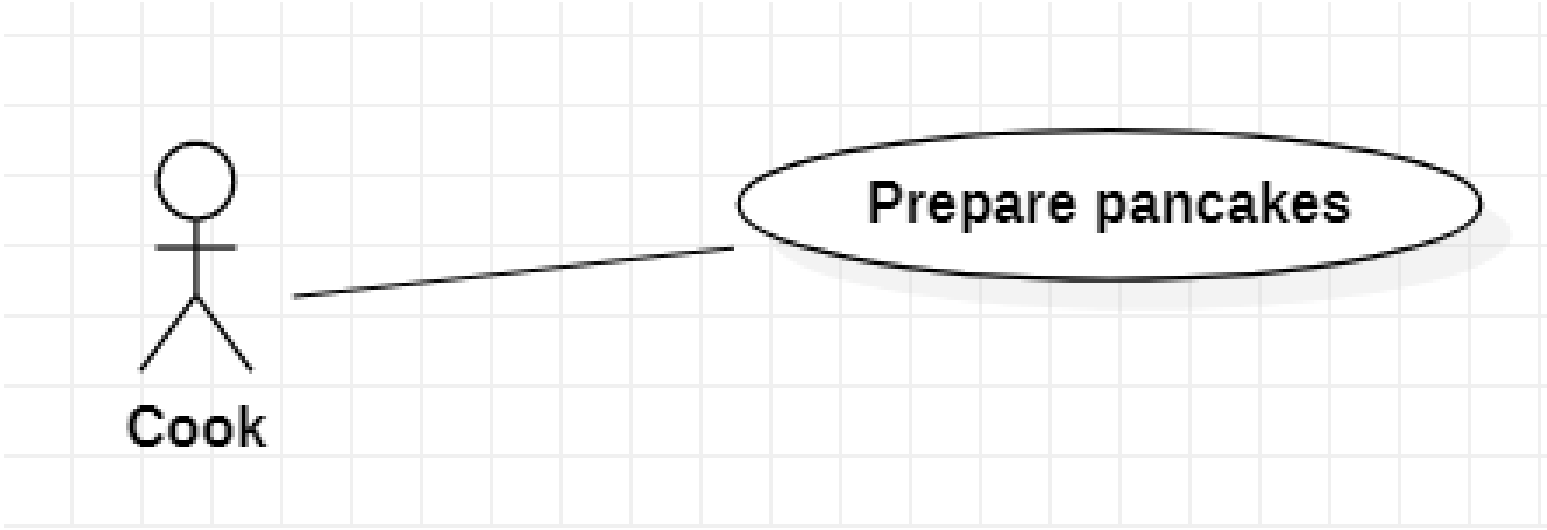
- i.e. It has many steps
- Long processes are harder to understand

Example: Prepare pancakes

- 1. Gather the ingredients for the batter:
  - 1 cup of all-purpose flour
  - 1 cup of milk
  - 1 tablespoon of melted butter or vegetable oil
  - 1 teaspoon of baking powder
  - 1 teaspoon of vanilla extract
  - 1 egg
  - 1/3 cup of sugar
- 2. Gather the extras:
- Extras (optional)
  - 1 cup of chocolate chips
  - 1 cup of blueberries
- 3. Gather the toppings
- Toppings (optional)
  - 1 cup of diced mango
  - 1 cup of diced peach
  - Whipping cream

- 4. Mix the wet ingredients (milk, melted butter or vegetable oil, vanilla extract and the egg) in a large bowl. The sugar is considered a “wet” ingredient, so it dissolves well.
- 5. Mix the dry ingredients (all-purpose flour and baking powder) in another bowl.
- 6. Pour the dry ingredients into the wet ones
- 7. Mix them together slowly until the batter is smooth
- 8. Heat a thick pan in high heat for 2 minutes
- 9. Turn the heat down to medium
- 10. Rub some butter on the pan
- 11. Pour some batter onto the pan
- 12. Put some extras on the batter when bubbles show
- 13. Flip the pan cake
- 14. Let it cook for a minute or so
- 15. Set aside
- 16. Repeat steps 11-16 until you are done with the batter.
- 17. Dice the fruit for the toppings (optional)
- 18. Serve and add toppings if so desire.

Partial use case diagram (Option 1)



Example: Prepare pancakes

- 1. Gather the ngredients for the batter:
  - 1 cu of all-purpose flour
  - 1 cu of milk
  - 1 tab spoon of melted butter or vegetable oil
  - 1 tea oon of baking powder
  - 1 tea oon of vanilla extract
  - 1 egg
  - 1/3 c o of sugar
- 2. Gather the xtras:
- Extras (optio l)
  - 1 cu of chocolate chips
  - 1 cu of blueberries
- 3. Gather the oppings
- Toppings (op onal)
  - 1 cu of diced mango
  - 1 cu of diced peach
  - Whi ing cream

1. Get ingredients

- 4. Mix the wet ingredients (milk, melted butter or vegetable oil, nilla extract and the egg) in a large bowl. The sugar is considered a “wet” ingredi t, so it dissolves well.
- 5. Mix the dry ingredients (all-purpose flour and baking powder) another bowl.
- 6. Pour the dry ingredients into the wet ones
- 7. Mix them together slowly until the batter is smooth
- 8. Heat a thick pan in high heat for 2 minutes
- 9. Turn the heat down to medium
- 10. Rub some butter on the pan
- 11. Pour some batter onto the pan
- 12. Put some extras on the batter when bubbles show
- 13. Flip the pan cake
- 14. Let it cook for a minute or so
- 15. Set aside
- 16. Repeat steps 11-16 until you are done with the batter.
- 17. Dice the fruit for the toppings (optional)
- 18. Serve and add toppings if so desire.

2. Prepare the batter

3. Cook the pancakes

4. Serve

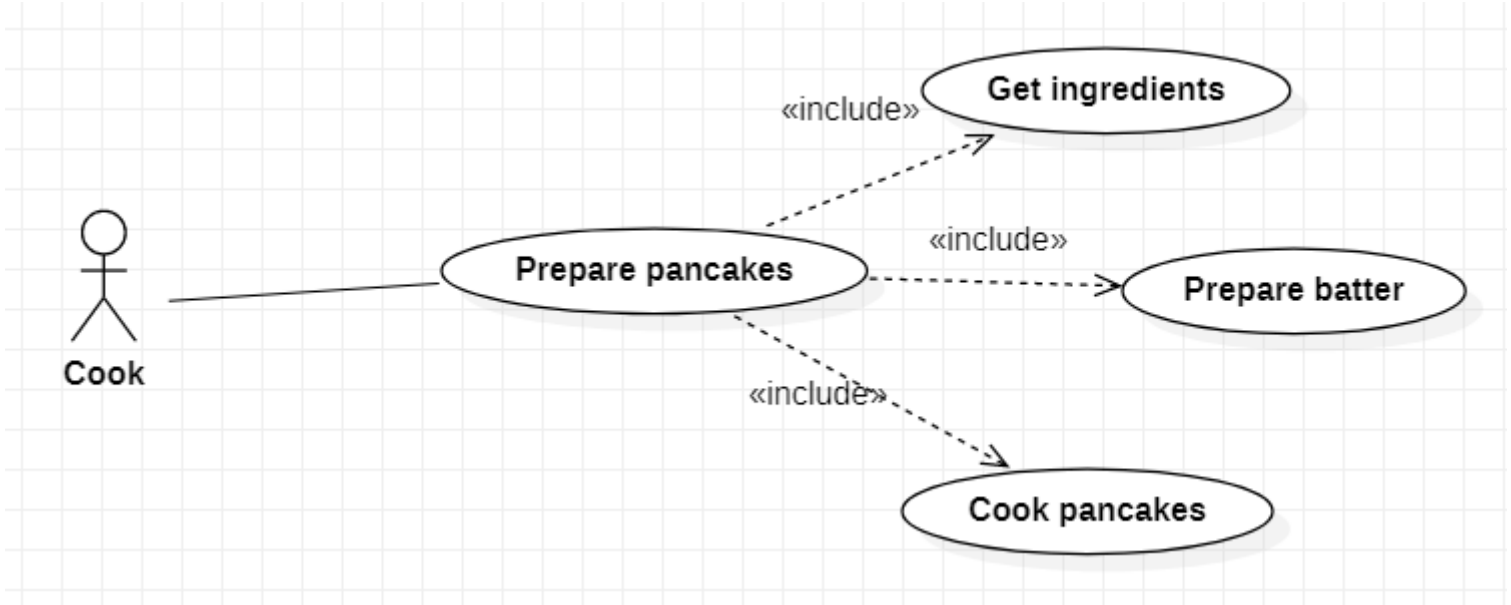
Factor out use cases

- To understand the use case “Prepare pancakes” much better

Prepare pancakes

1. Get ingredients
2. Prepare the batter
3. Cook the pancakes
4. Serve

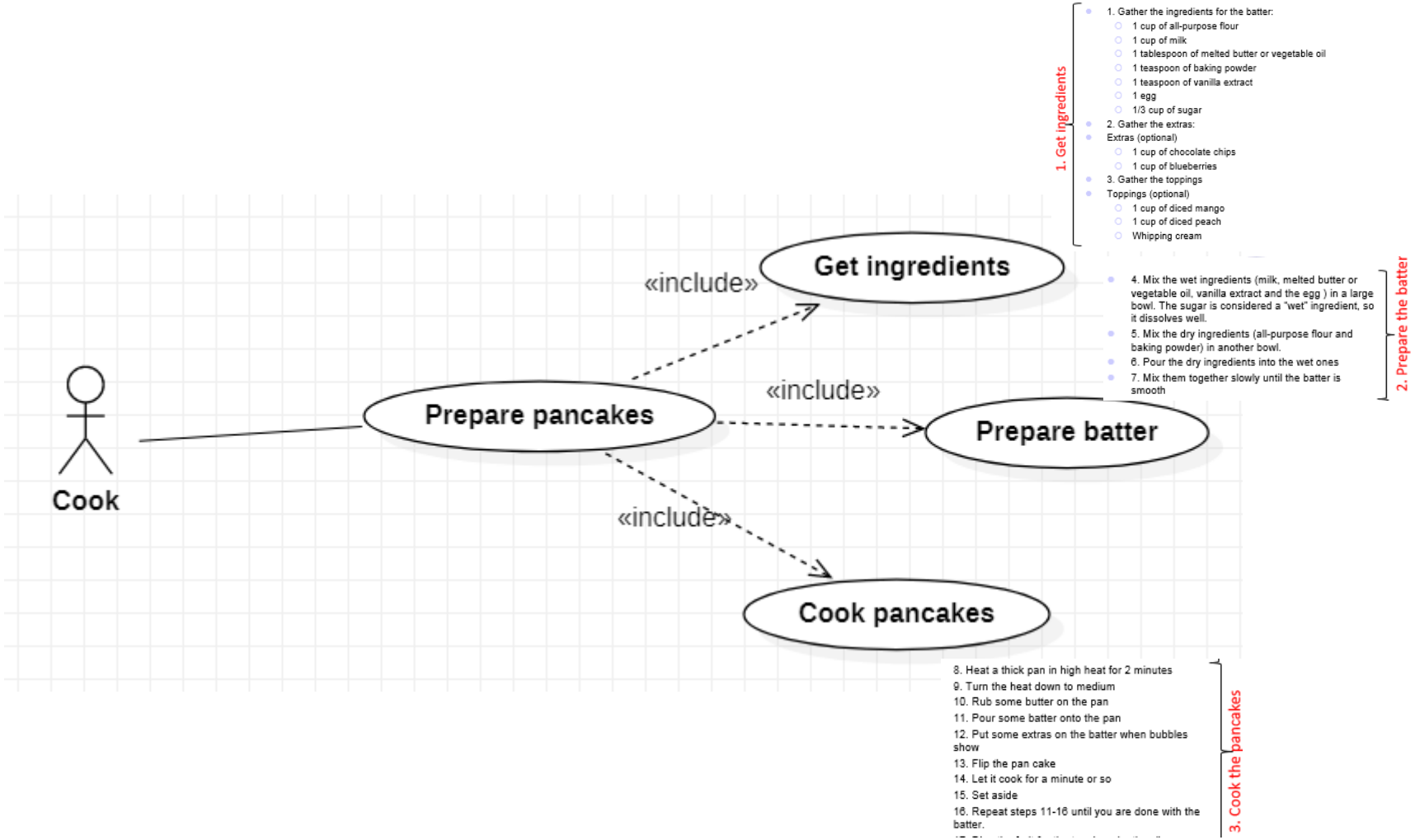
Partial use case diagram (Option 2)



Prepare pancakes

1. Get ingredients
2. Prepare the batter
3. Cook the pancakes
4. Serve

Partial use case diagram (Option 2)



“Extend” relationship

What is the *extend* relationship?

- When you want to add more behavior to a base use case

When do you use *extend*?

- 1. When the added behavior is optional
  - Not in all cases it occurs
- 2. When you want to represent a sequence of scenarios

Los casos de uso relacionados con extend son independientes

Sequences

Sequence of scenarios

- Sometimes you want to represent a sequence of scenarios

- Take lesson 1
- Take lesson 2
- Take lesson 3

Example

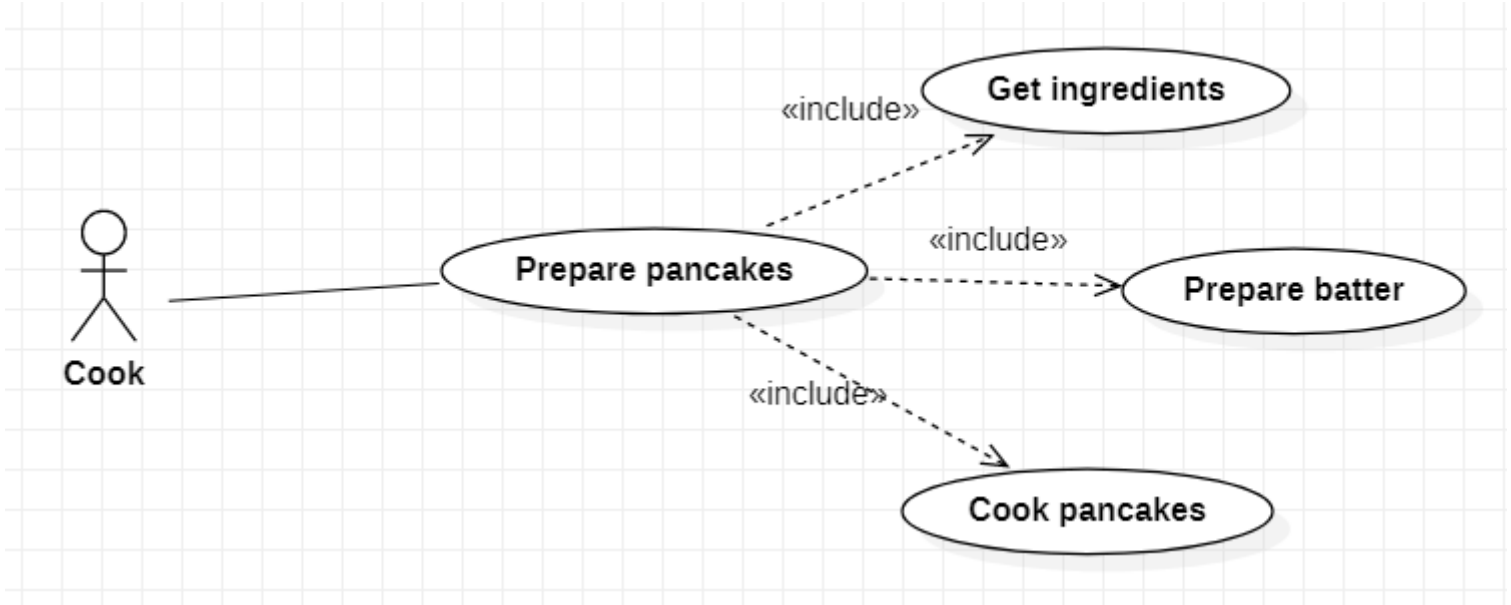
Notice the direction of the arrows

Notice the direction of arrow

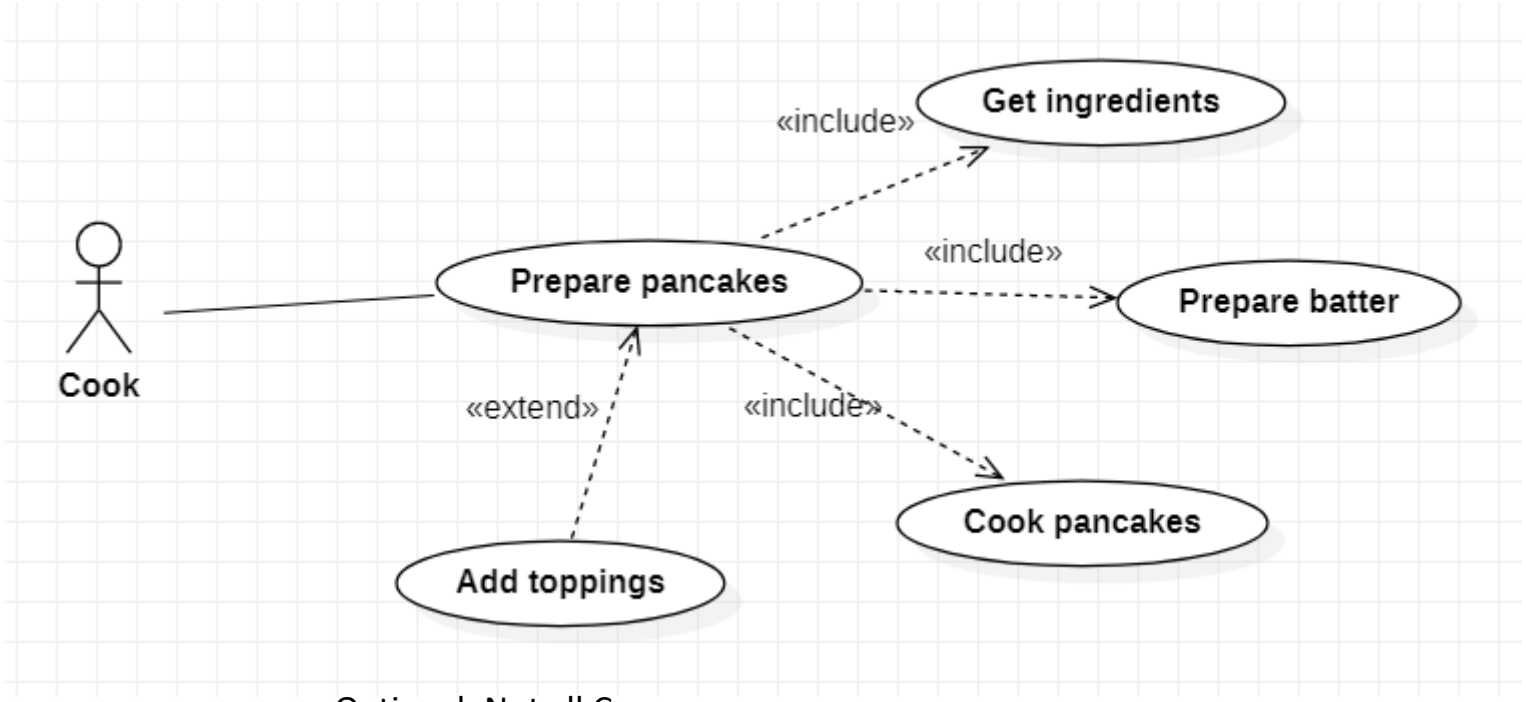
Include

Extend

Partial use case diagram



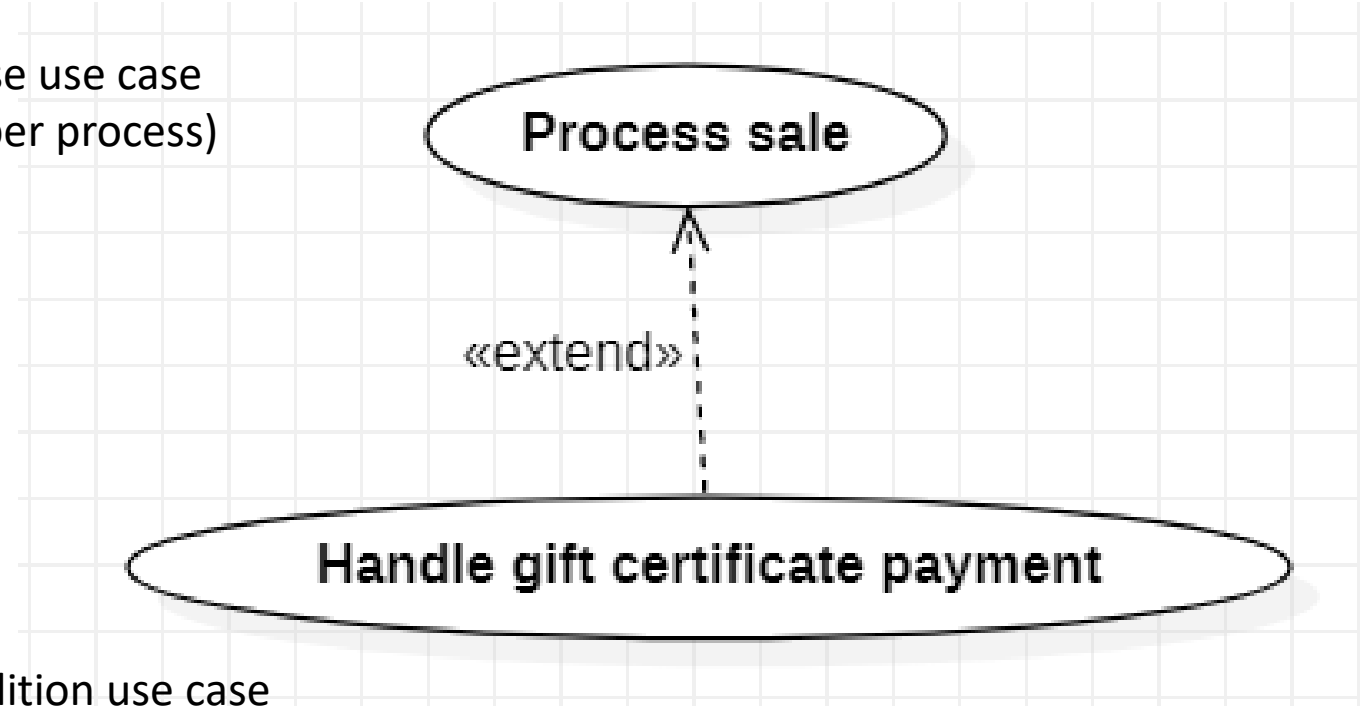
Extending “Prepare pancakes”



Optional: Not all Consumers ask for toppings

Partial use case diagram

Base use case
(super process)



Addition use case
(sub process)

Clase 2

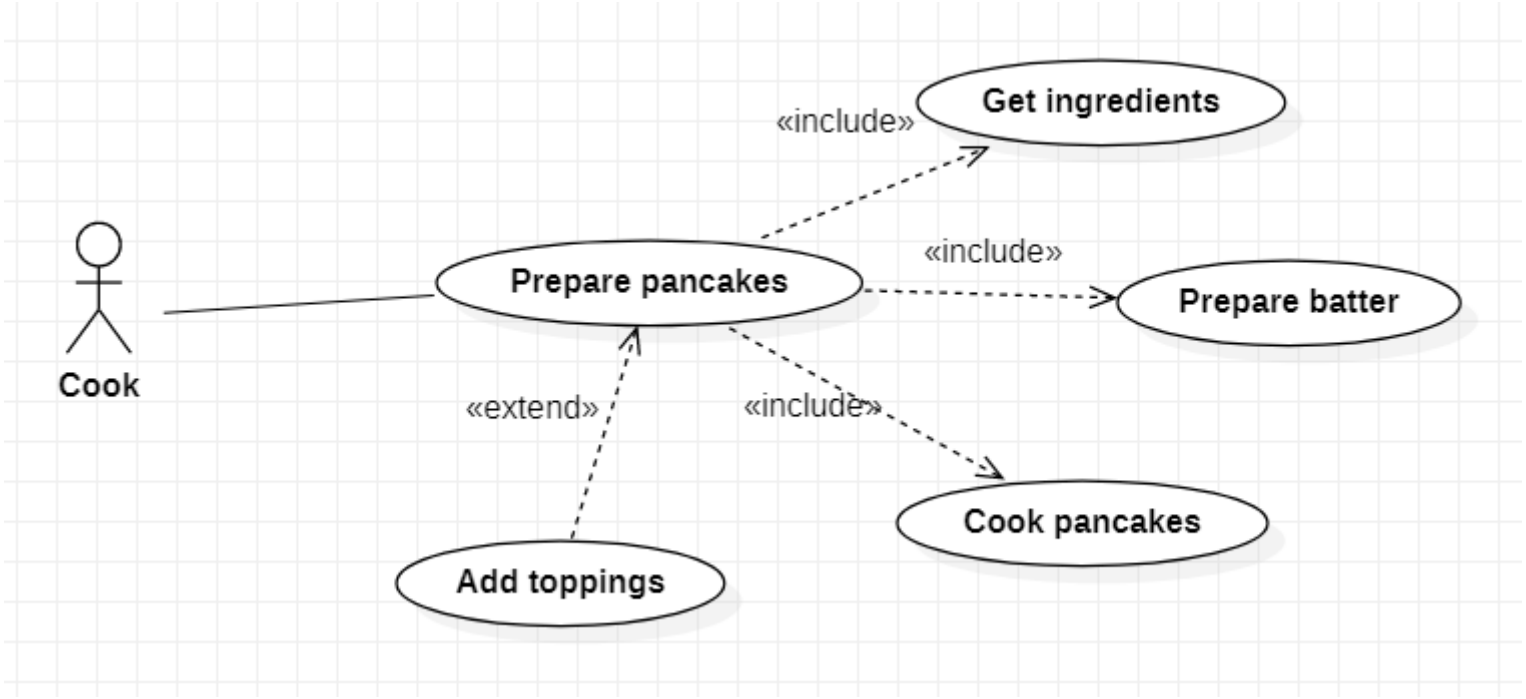
Extension points

Advanced notation

Where in the base case are we extending?

- We use “Extension points” to mark the places
- They are optional

In which step do we add toppings?

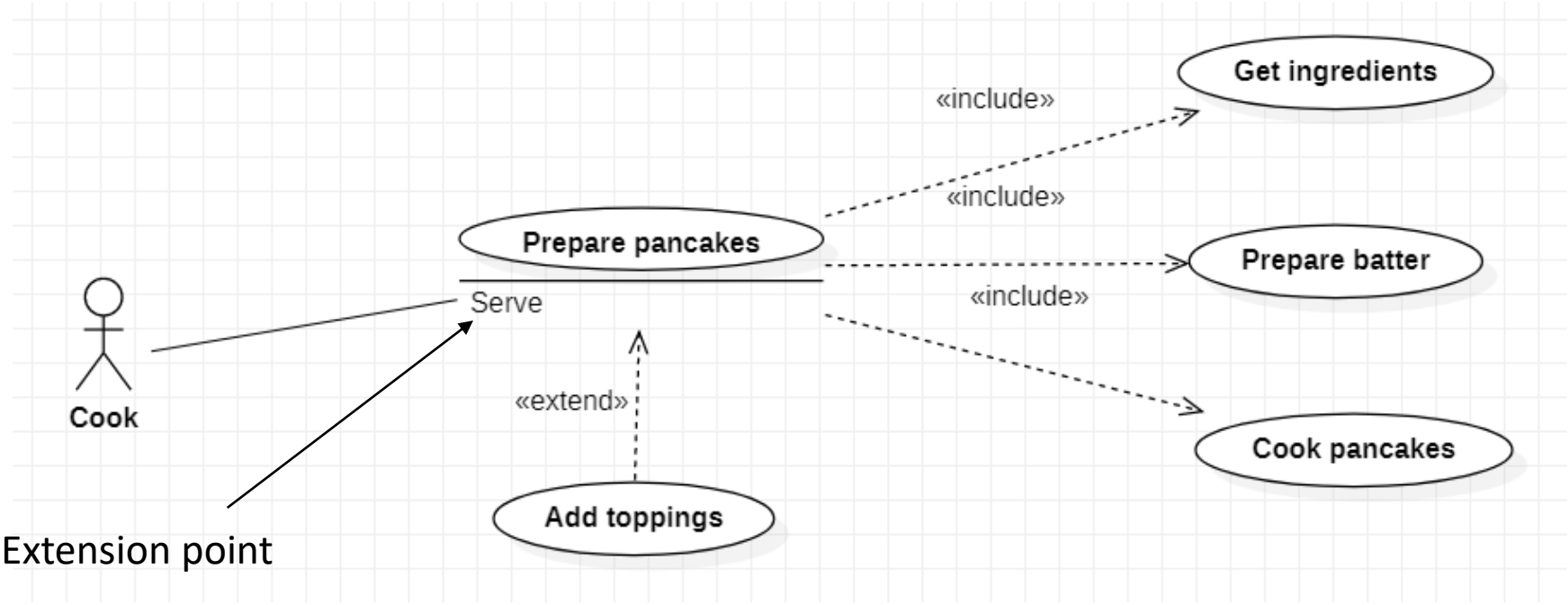


Prepare pancakes

1. Get ingredients
2. Prepare the batter
3. Cook the pancakes
4. Serve

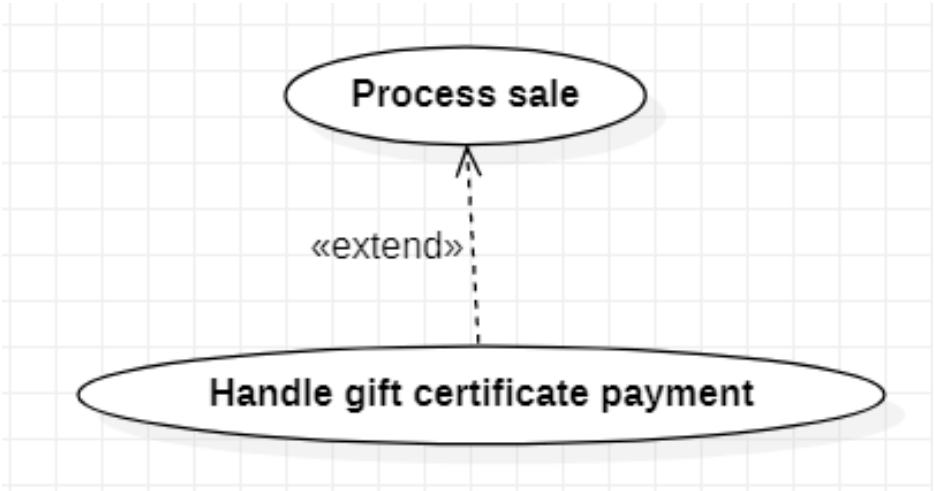
We specify it with extension points

Using StarUML

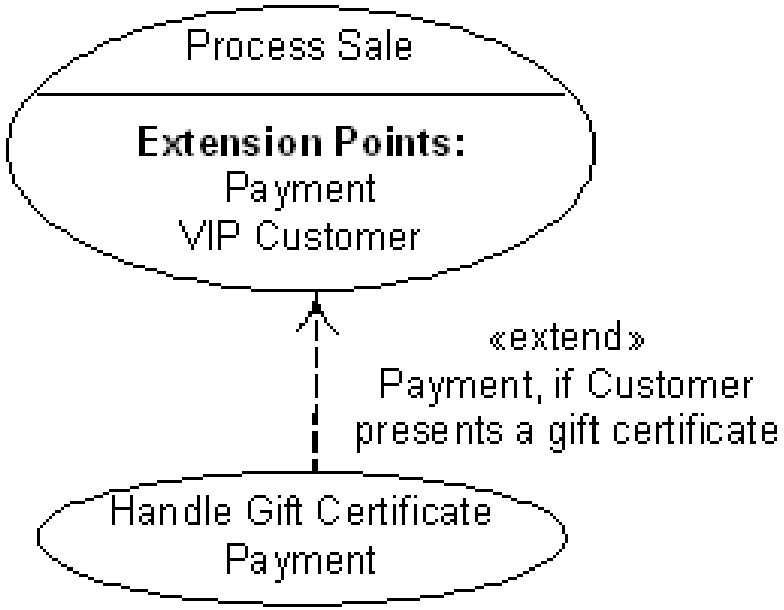


A base case with an extension

Plain



With extension points



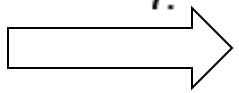
Notation in textbook [2]

Process sale Basic Flow

7. Main Success Scenario (or Basic Flow)

Actor Action	System Response
1. Customer arrives at POS checkout with goods and/or services to purchase.	
2. Cashier starts a new sale.	
3. Cashier enters item identifier *Cashier repeats steps 3-4 until indicates done.*	4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.
5. Cashier finalizes the sale	6. System presents total with taxes calculated
7. Cashier tells Customer the total and asks for payment.	
8. Customer pays and the Cashier introduces the payment	9. System handles the payment
	10. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
	11. System presents receipt.
12. Customer leaves with receipt and goods (if any).	

Specifying extension points

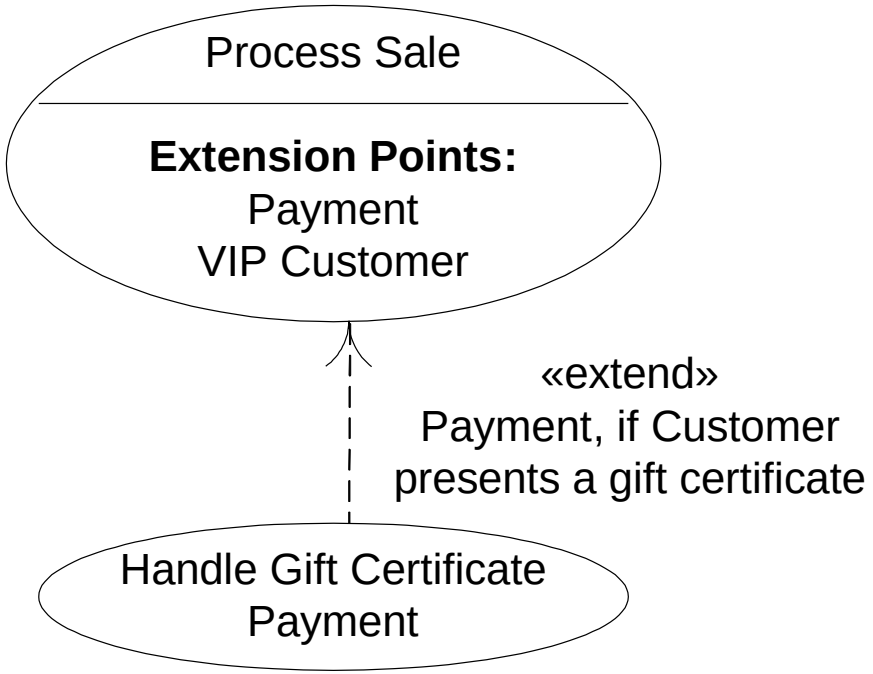


7. Main Success Scenario (or Basic Flow)

Extension Points: VIP Customer, step 1. Payment, step 7.

Actor Action	System Response
1. Customer arrives at POS checkout with goods and/or services to purchase.	
2. Cashier starts a new sale.	
3. Cashier enters item identifier *Cashier repeats steps 3-4 until indicates done.*	4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.
5. Cashier finalizes the sale	6. System presents total with taxes calculated
7. Cashier tells Customer the total and asks for payment.	
8. Customer pays and the Cashier introduces the payment	9. System handles the payment
	10. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
	11. System presents receipt.
12. Customer leaves with receipt and goods (if any).	

Extension point notation in (Larman, 2005)



UML notation:
1. The extending use case points to the base use case.

2. The condition and extension point can be shown on the line.

Figure 30.2 from [2, 4]

Extension point notation in (Larman, 2005)

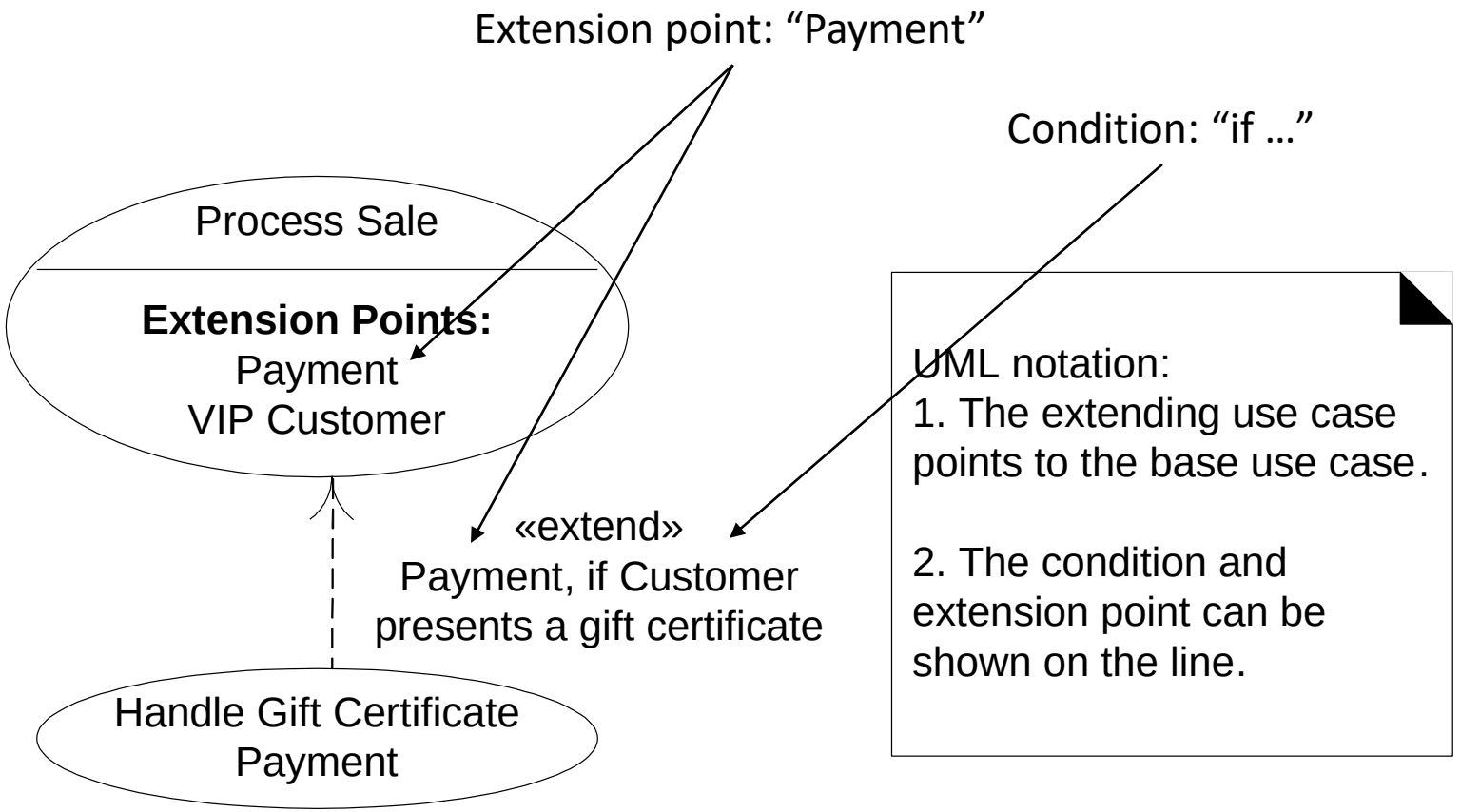


Figure 30.2 from [2, 4]

Notation in the Standard 2.5.1 [5]

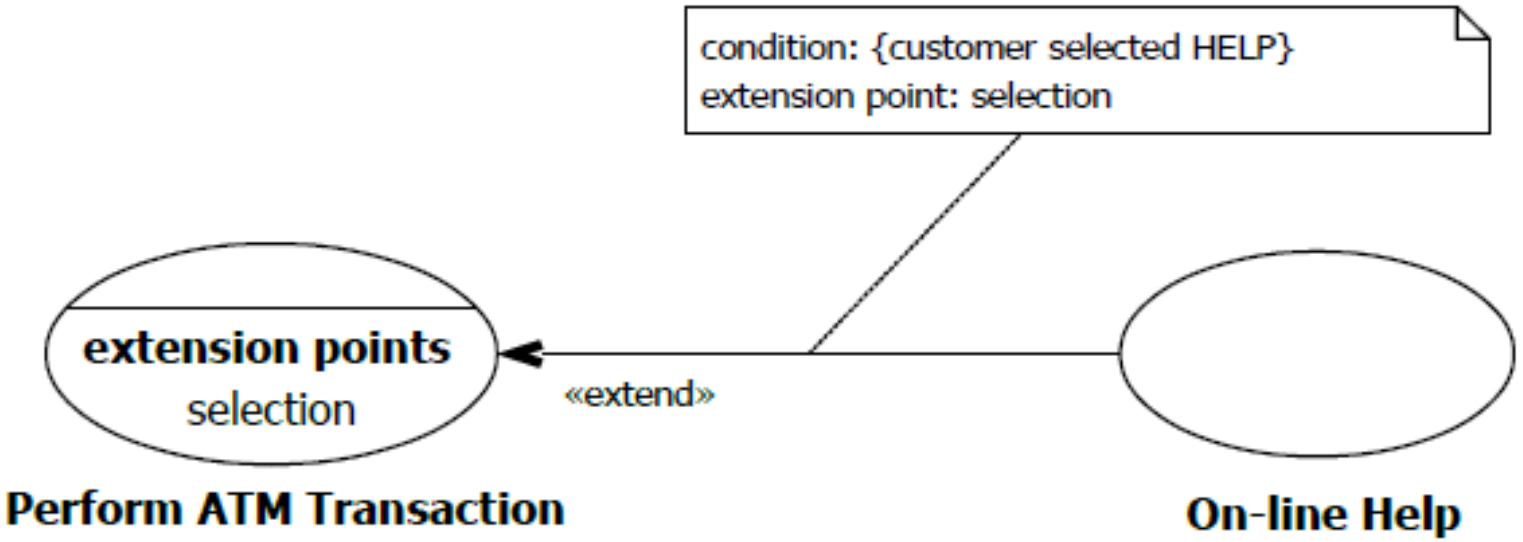


Figure 18.3 from [5]

Notation in StarUML

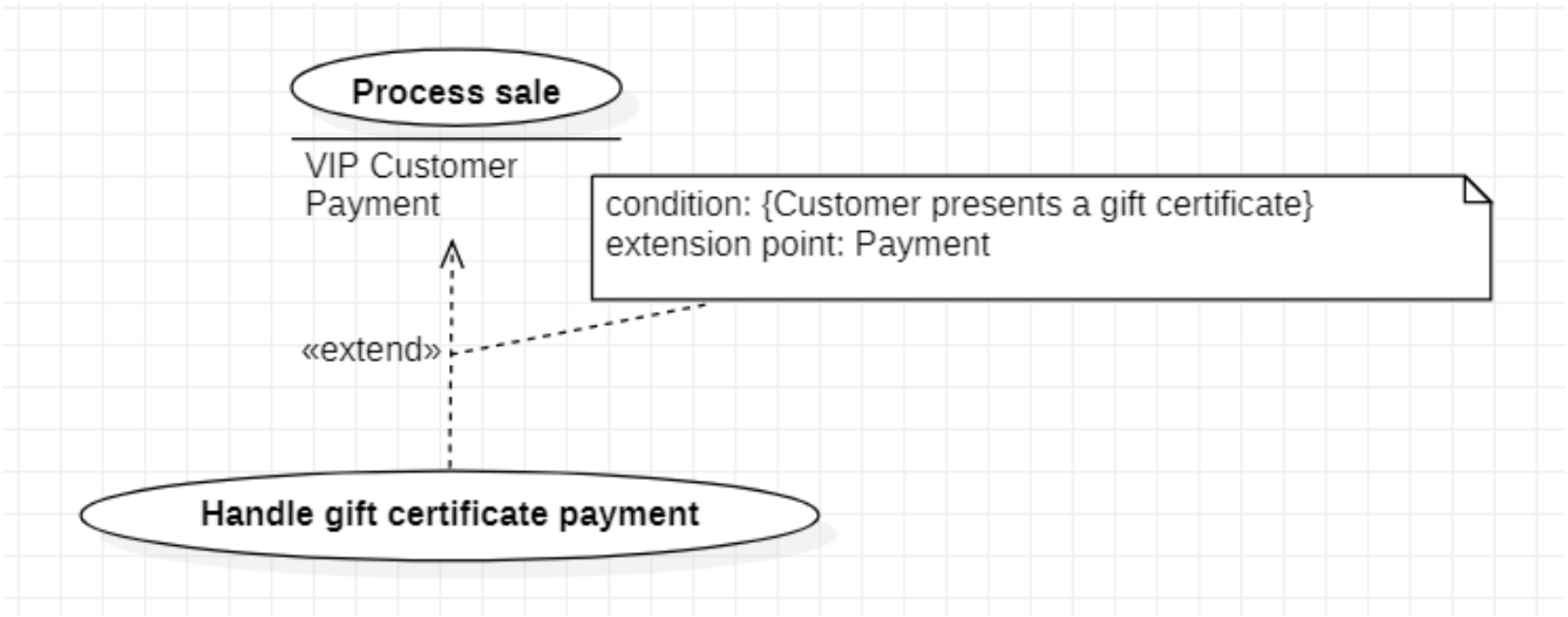
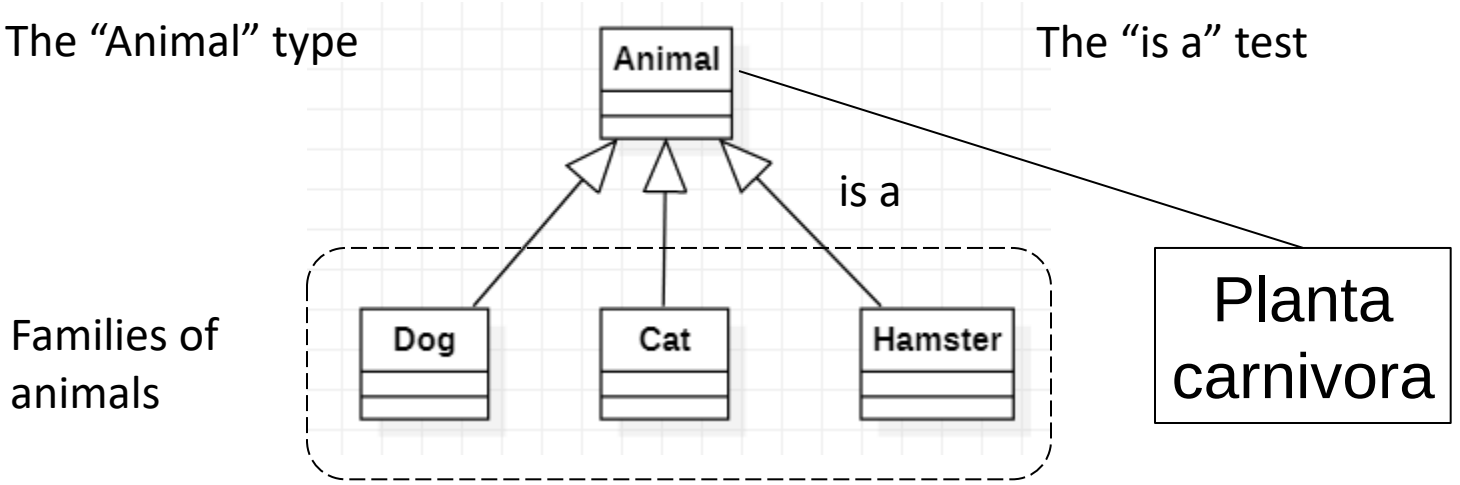


Figure inspired from Figure 18.3 in [5]

Generalization-specialization

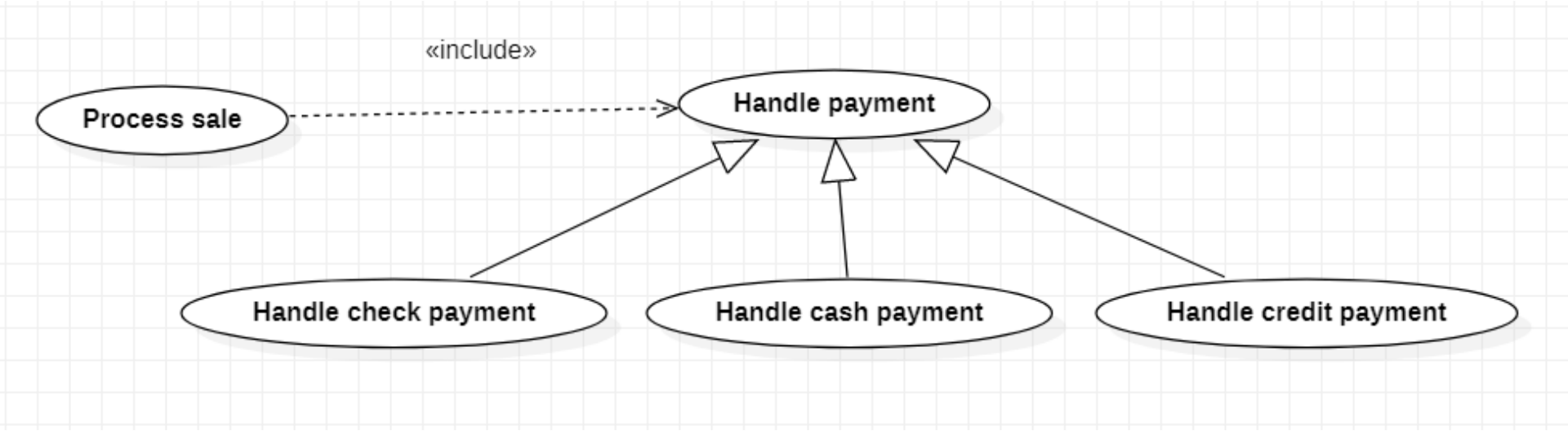
Generalization-specialization

- Just like with class diagrams where you can generalize families of classes into types, we can do the same with use cases

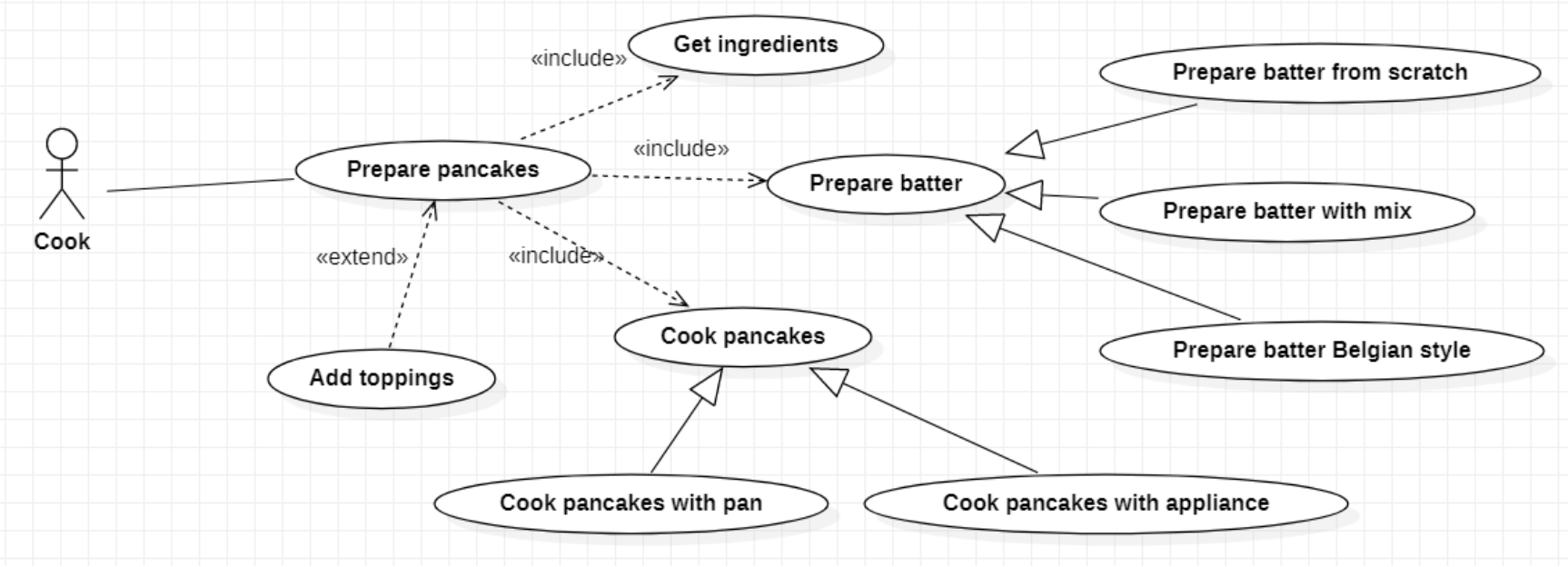


Generalization-specialization

- The sub use cases:
  - inherit behavior from the base use case
  - add behavior to the base use case if needed

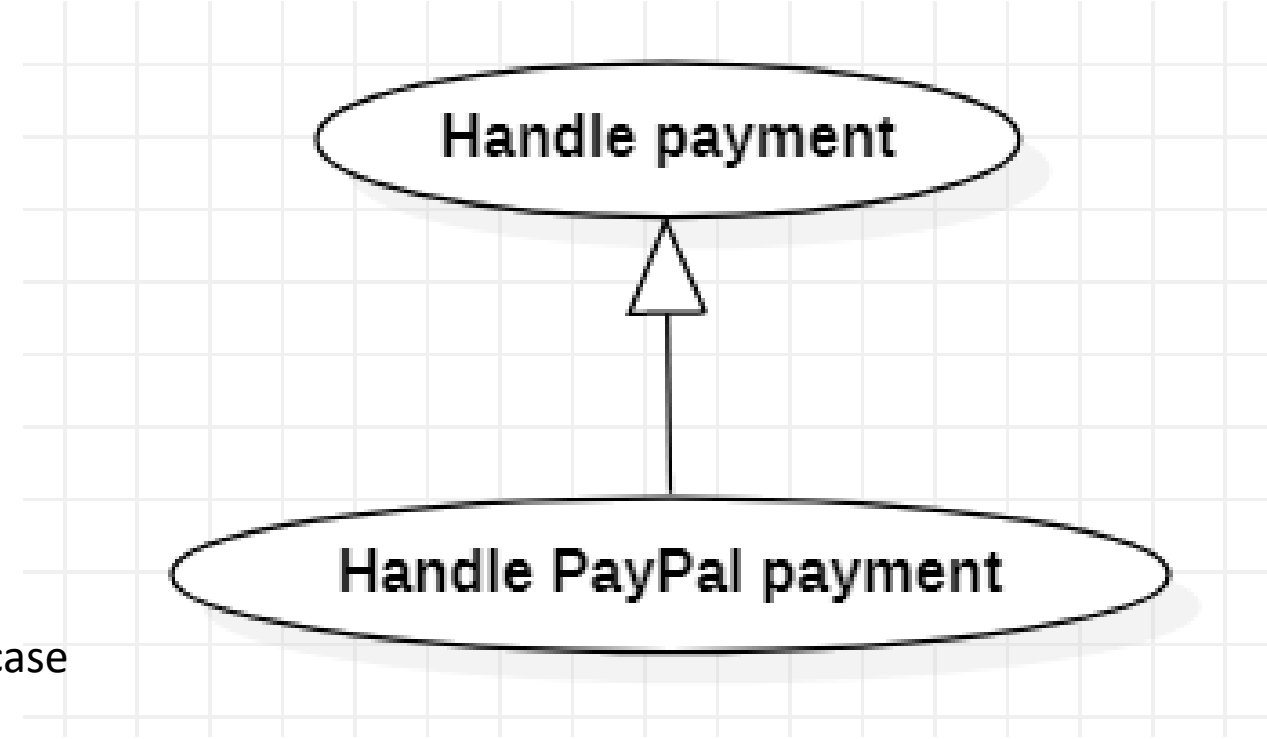


Partial use case diagram



Base and addition use cases

Base use case

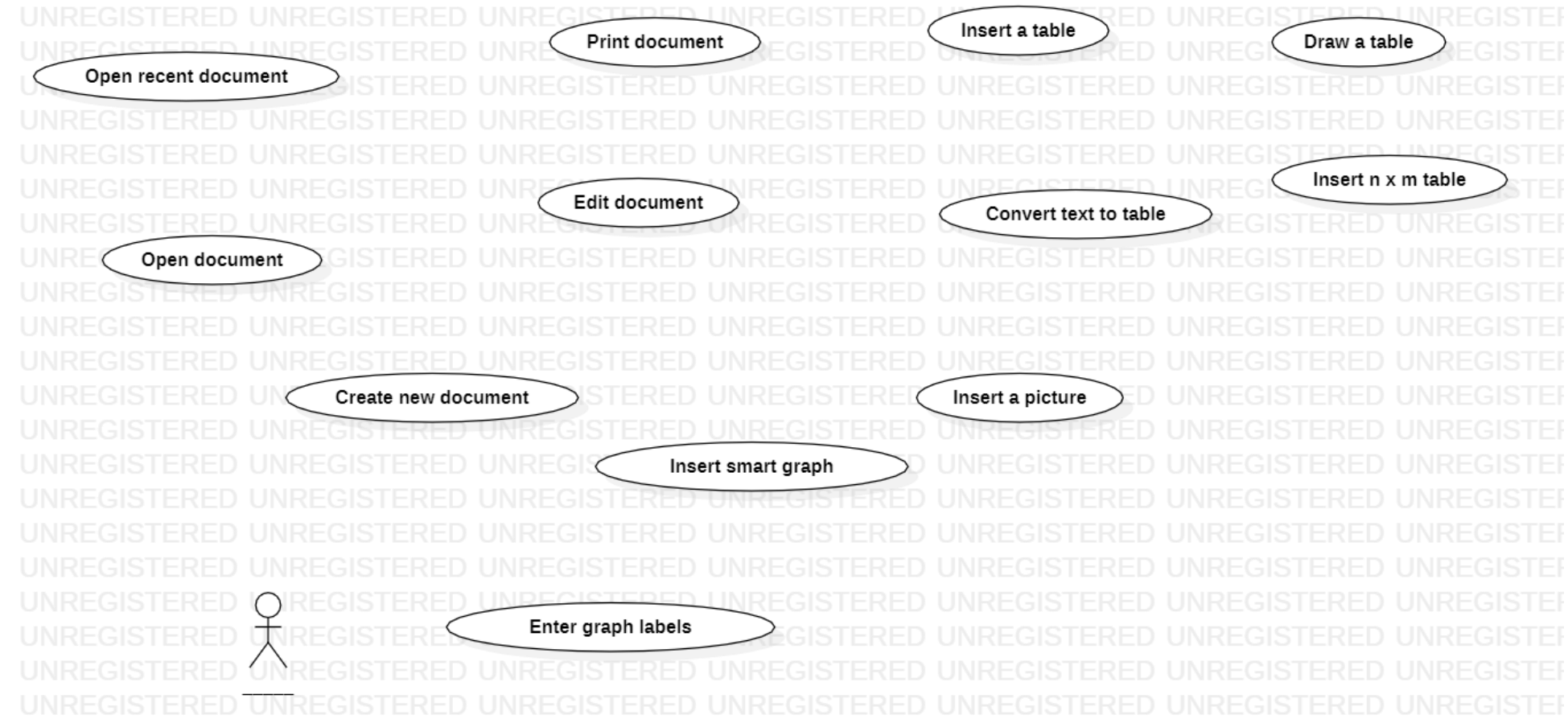


Addition use case

Activity (Optional)

- Given a set of use cases, indicate how they relate
- Application example: A word processor

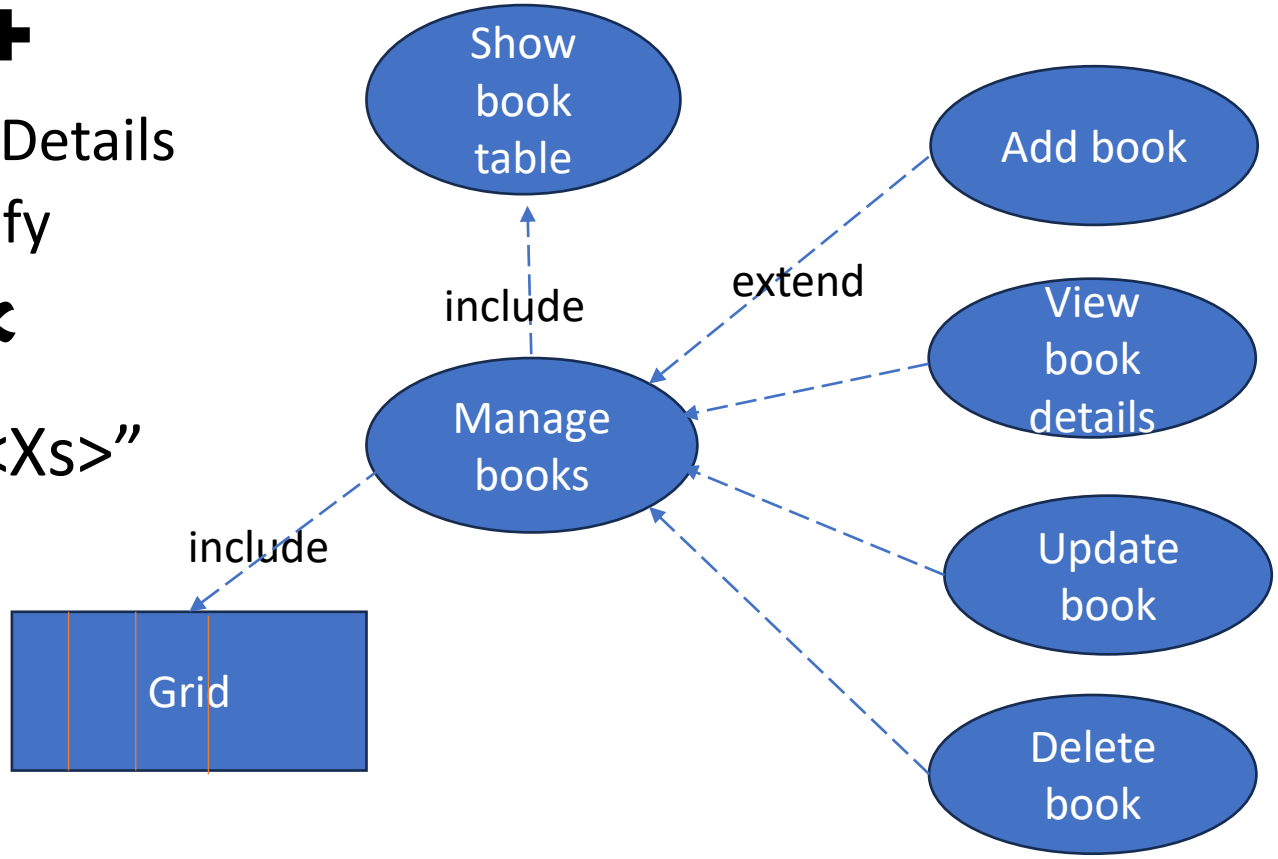
Microsoft Word use case diagram



CRUD

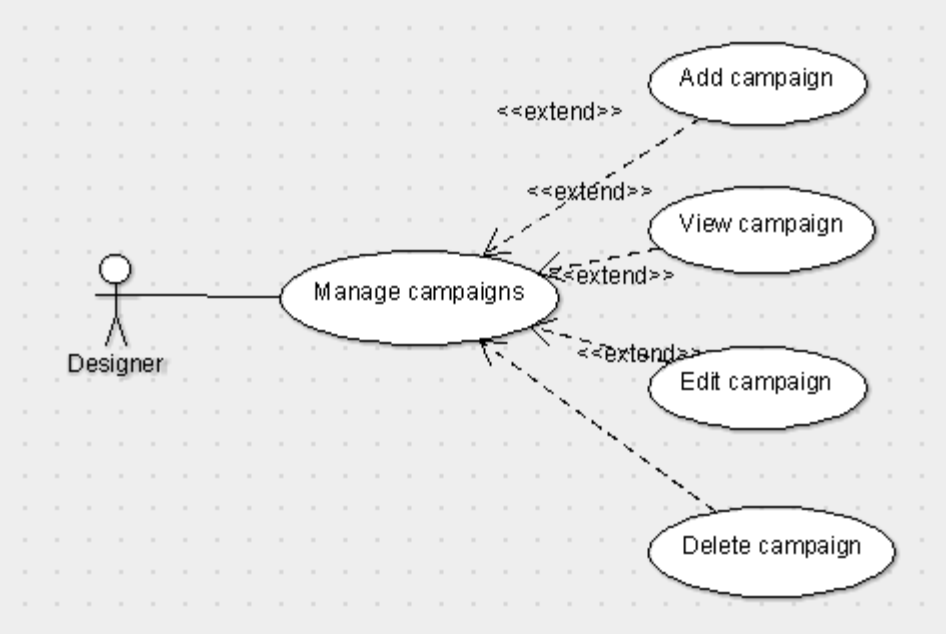
CRUD use cases [1]

- “A compound use case that collapses a create, retrieve, update, and delete use cases [1]”:
  - **C**reate: Add/New/Insert/✚
  - **R**etrieve: Read/Open/View/Details
  - **U**pdate: Update/Edit/Modify
  - **D**elete: Delete/Remove/✘
- Normally called “Manage <Xs>”

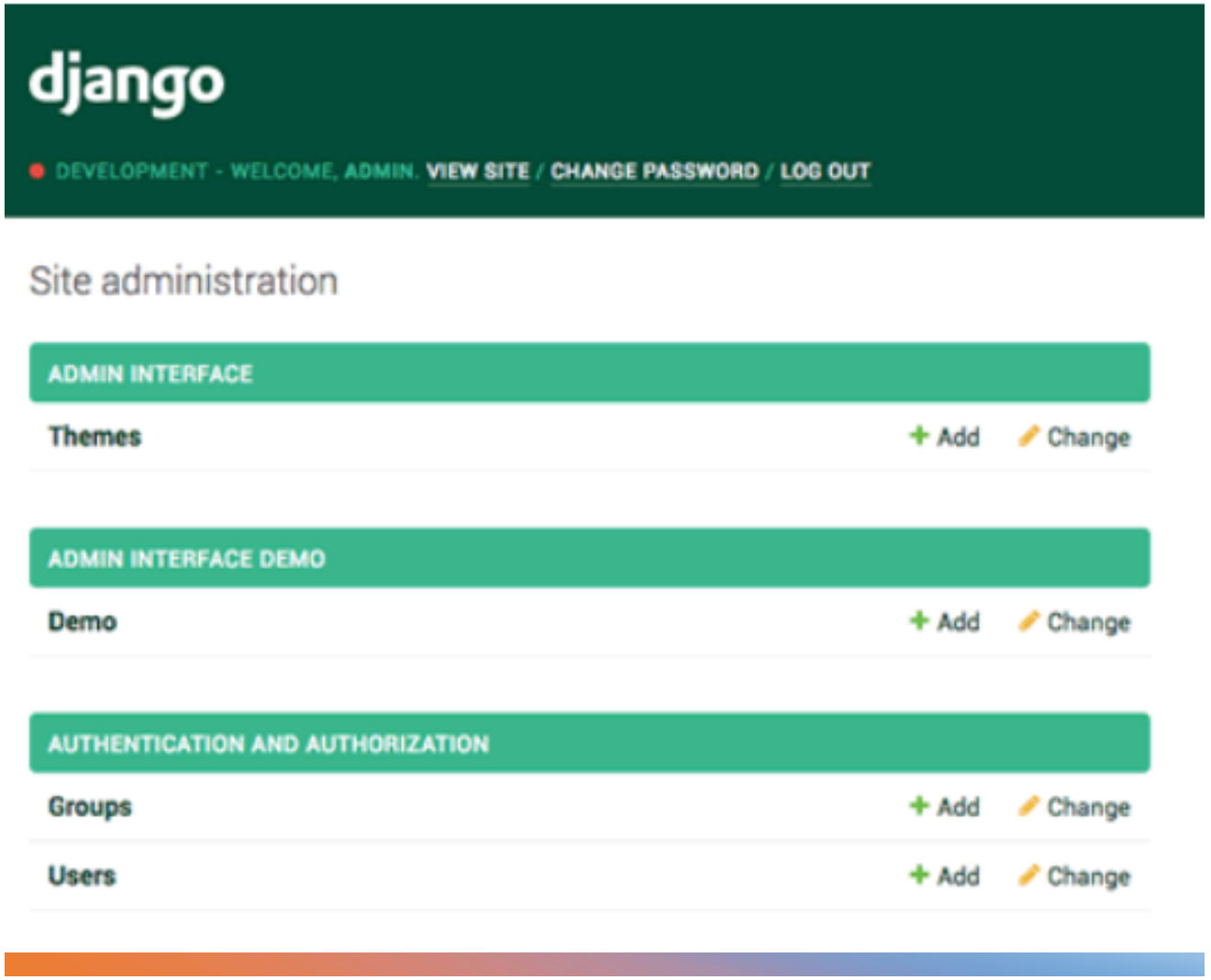


Example: Manage campaigns

- **C**reate: Add campaign
- **R**etrieve: View campaign
- **U**pdate: Edit campaign
- **D**elete: Delete campaign

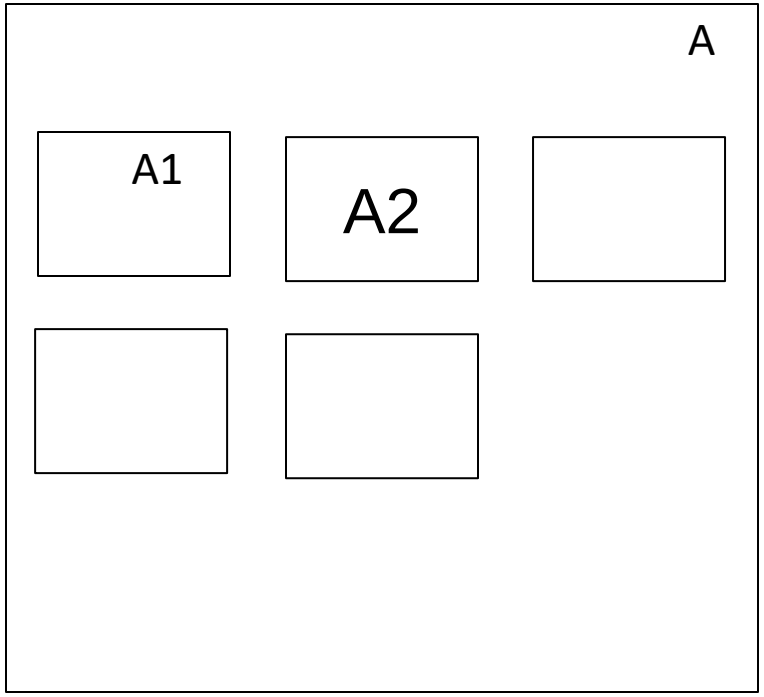


Activity: Represent the use cases suggested by the screen below from django



Student questions

Includes (composición)

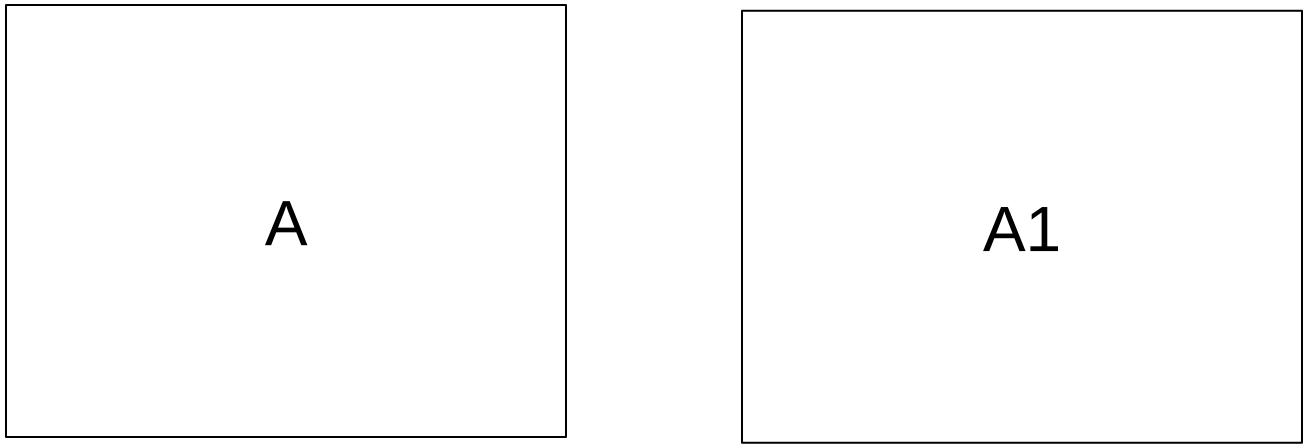


Mi “todo” incluye las partes

The “has a” test

<<include>>
A ----------> A1

Extends (opcional/secuencias)



opcional

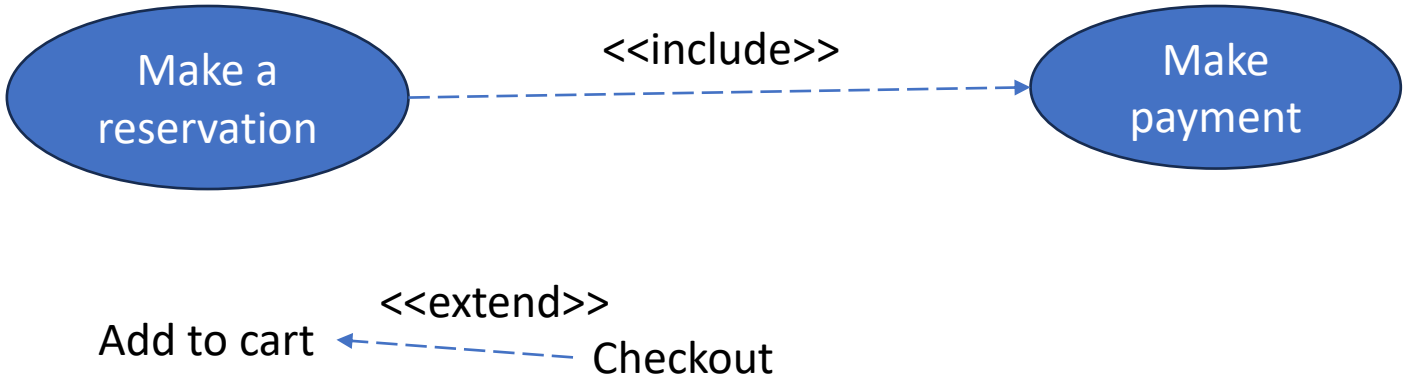
<<extend>>
A <---------- A1

Pregunta 1

In a hotel reservation system, which relationship would be most appropriate for the following scenario: "Make Payment for Reservation"?
A) Includes
B) Extends
C) Generalization-Specialization

Pregunta 1

In a hotel reservation system, which relationship would be most appropriate for the following scenario: "Make Payment for Reservation"?
A) Includes
B) Extends
C) Generalization-Specialization



Actividad

- Quiz E09-1 Repaso de diagramas de casos de uso

System Sequence Diagrams

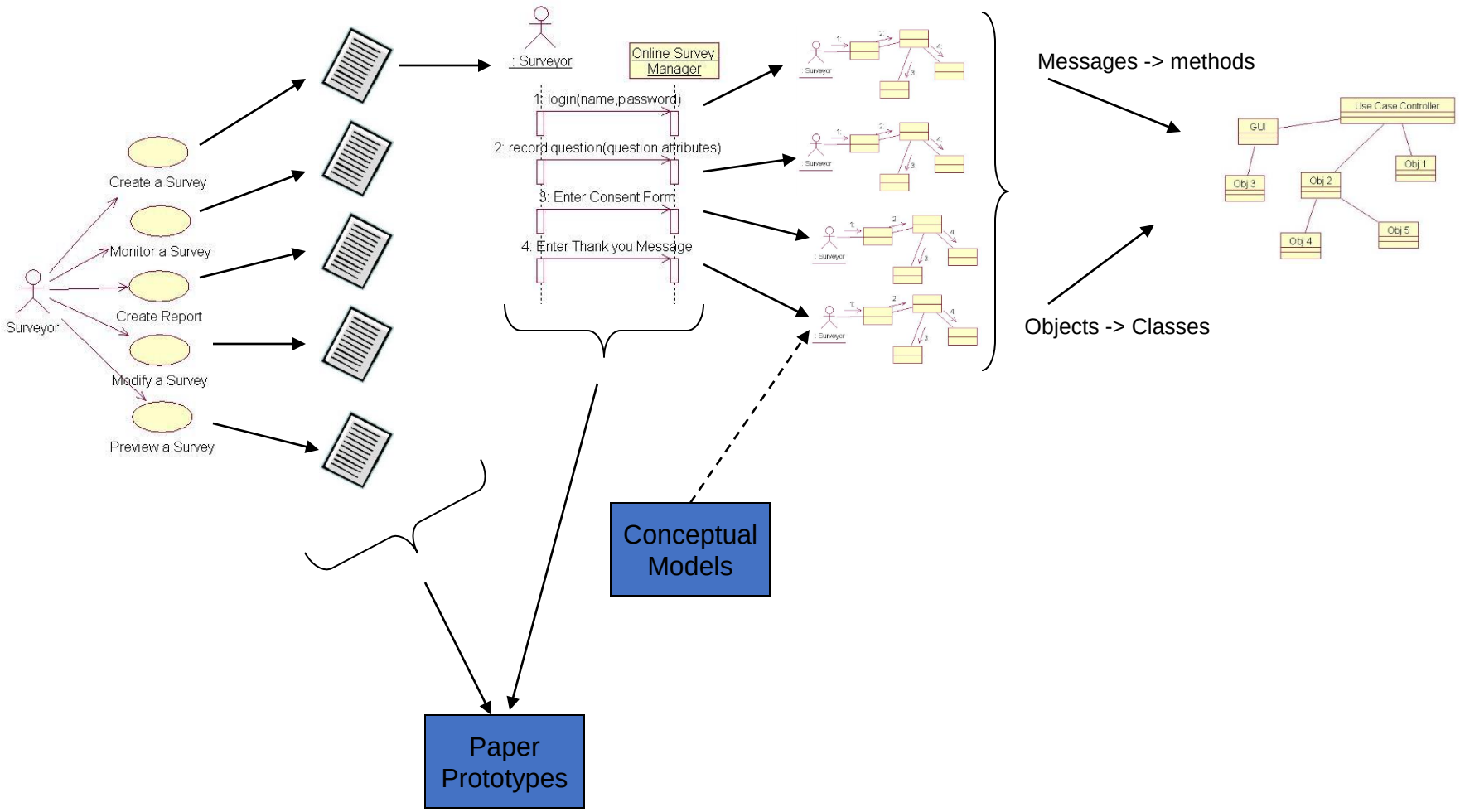
UML process [1]

Use Case Diagrams

System Sequence Diagrams

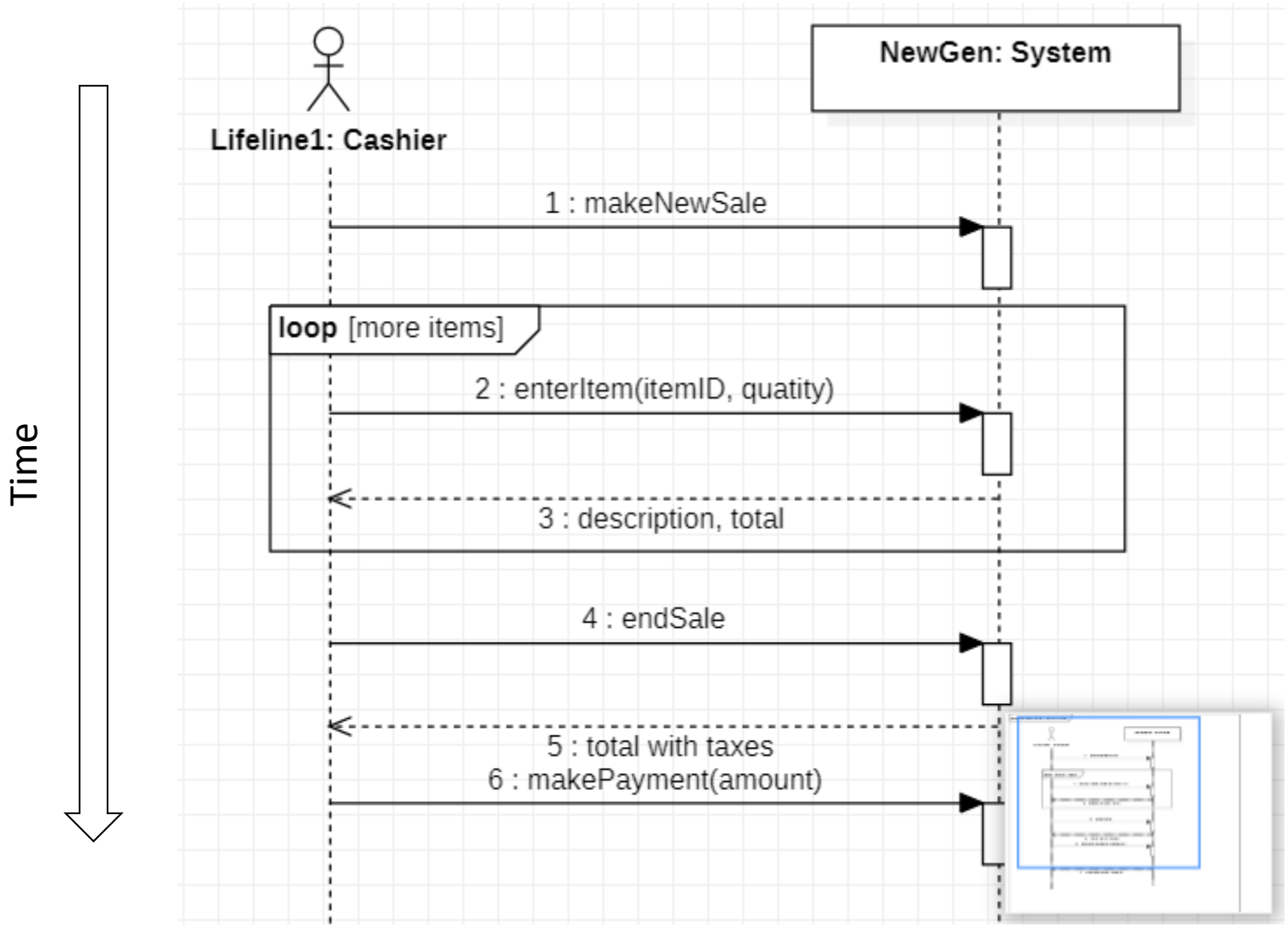
Collaboration Diagrams

Design Class Diagrams



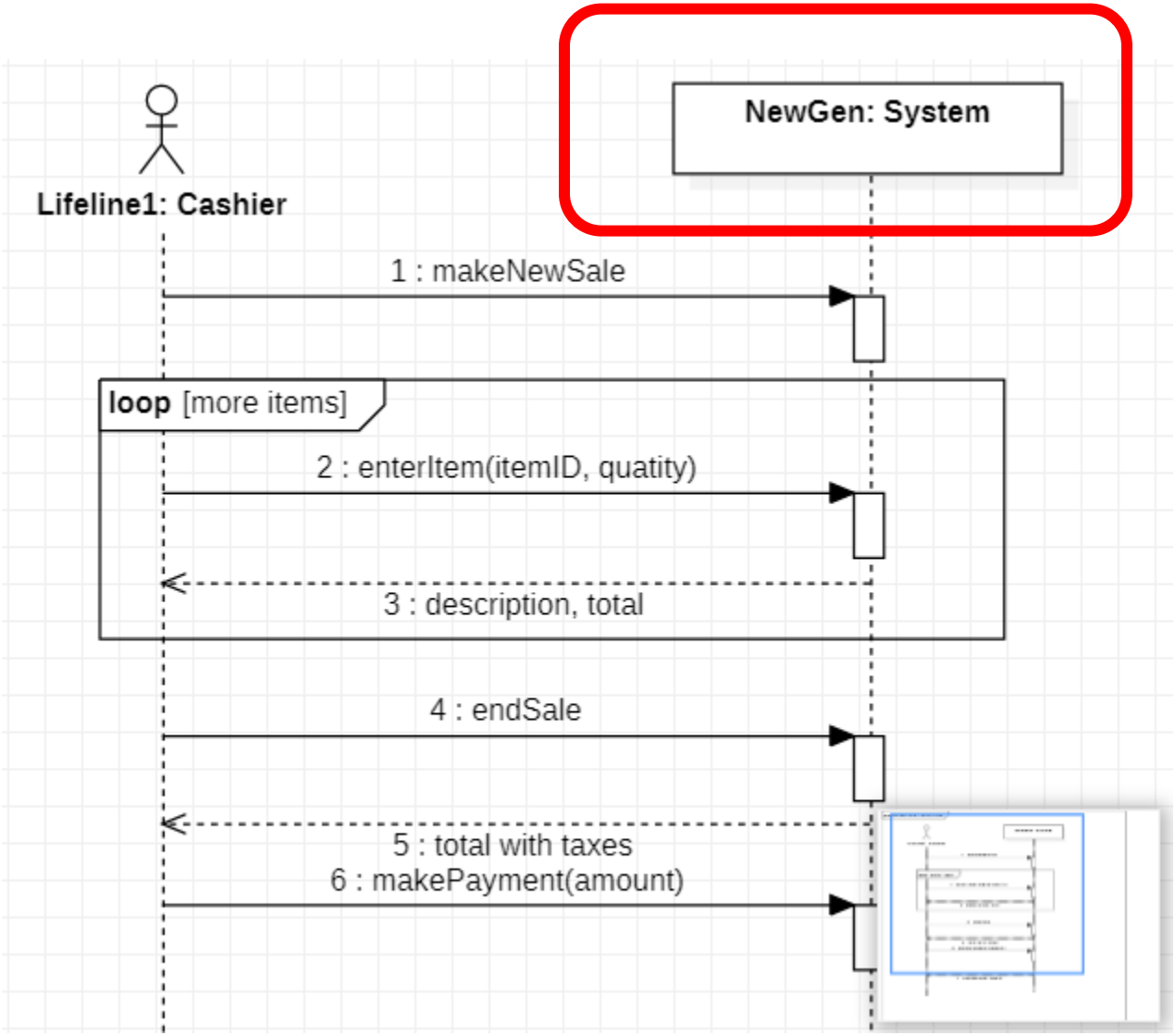
What is a System Sequence Diagram?

- It is a diagram where we depict the steps an actor goes through in time a use case or to achieve a goal


Time
NewGen: System
Lifeline1: Cashier
1 : makeNewSale
loop [more items]
2 : enterItem(itemID, quatity)
3 : description, total
4 : endSale
5 : total with taxes
6 : makePayment(amount)

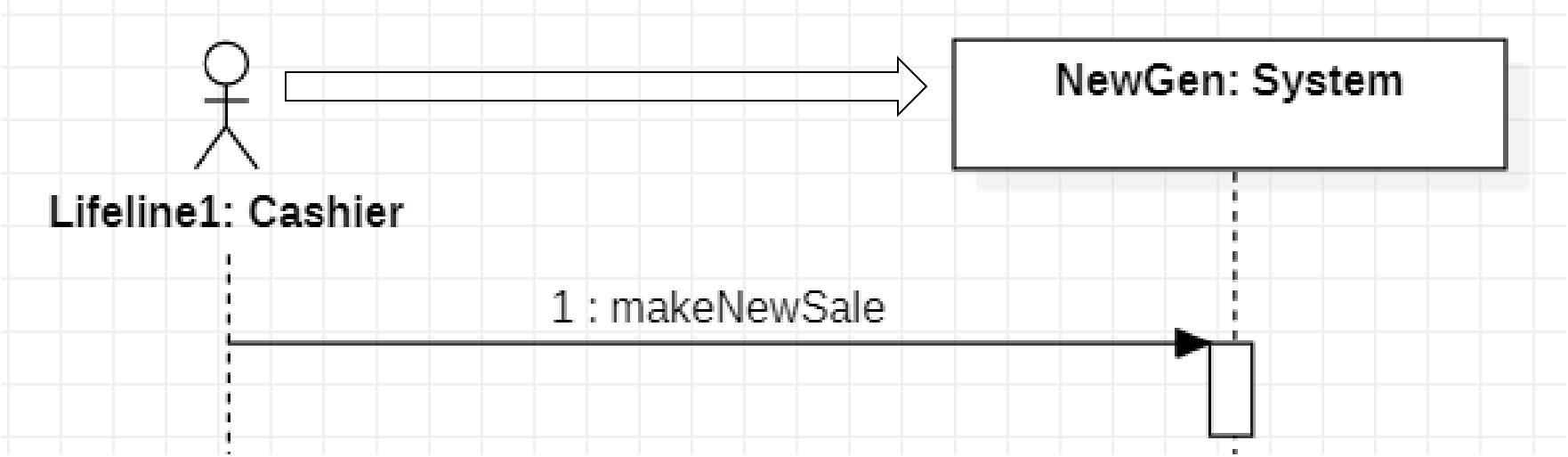
Systems are treated as a black box [2]

We don’t show
inner parts
of the system, e.g
database.



What is a system event?

- It is an interaction between an actor with the system



Basic notation

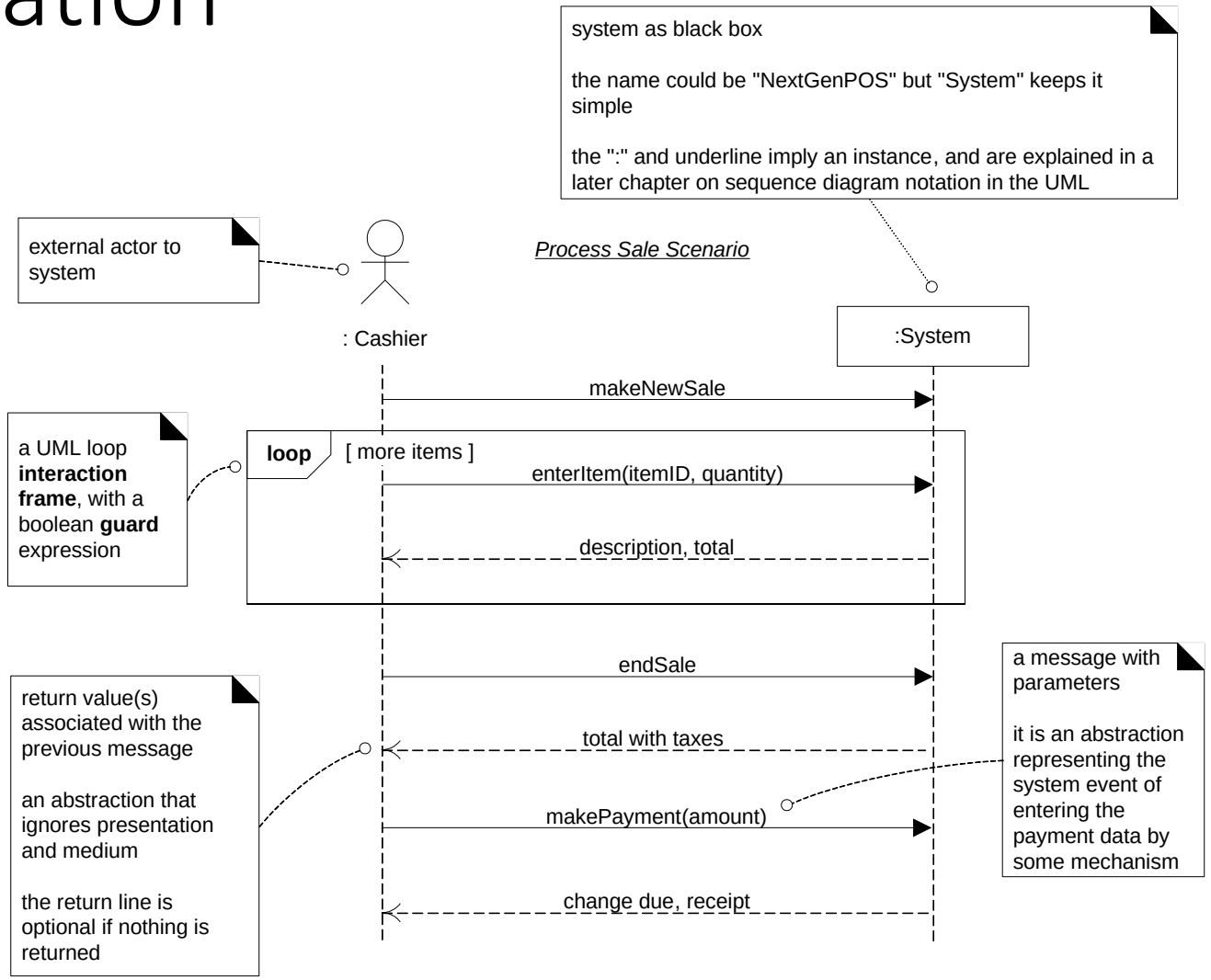
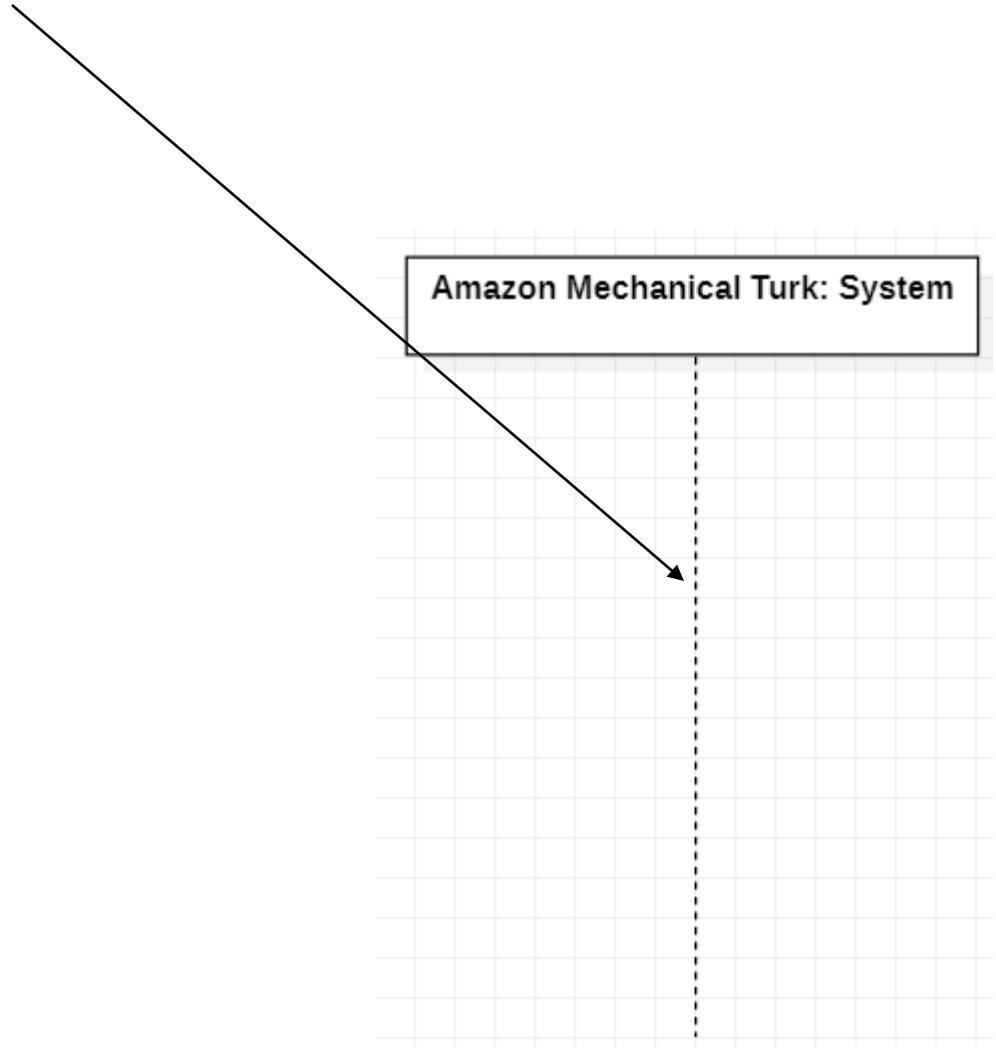


Figure 10.2 from [1]

Lifeline



Inter-system events [2]

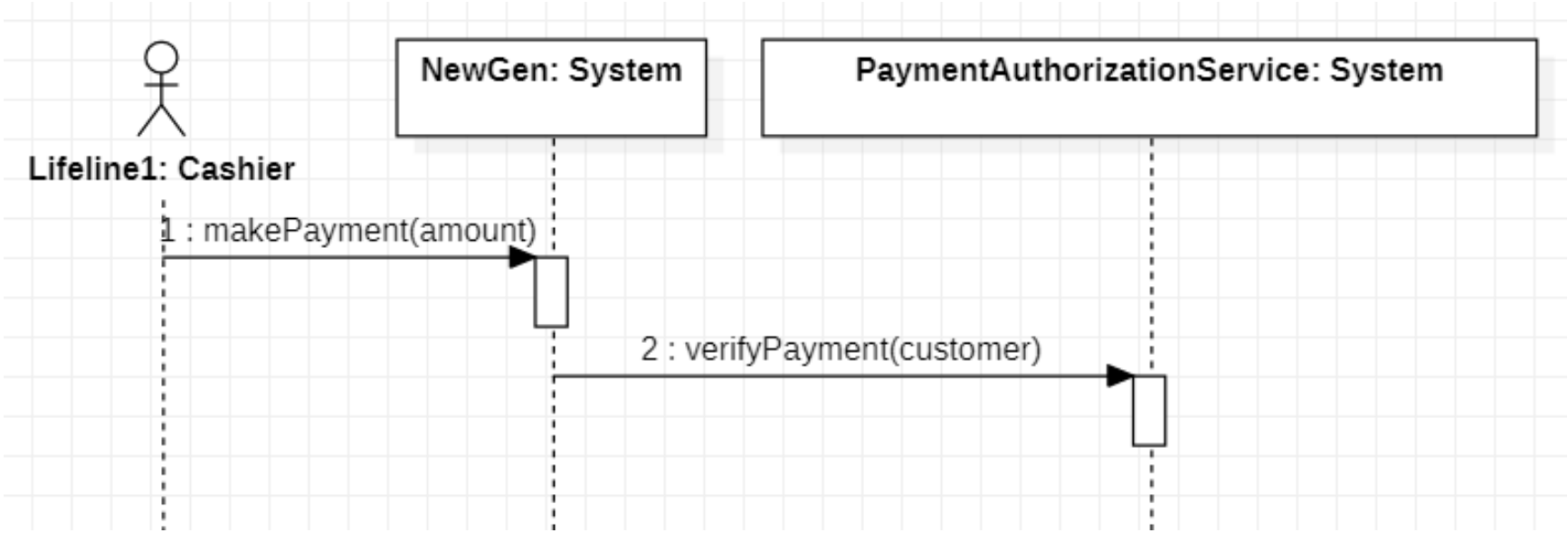
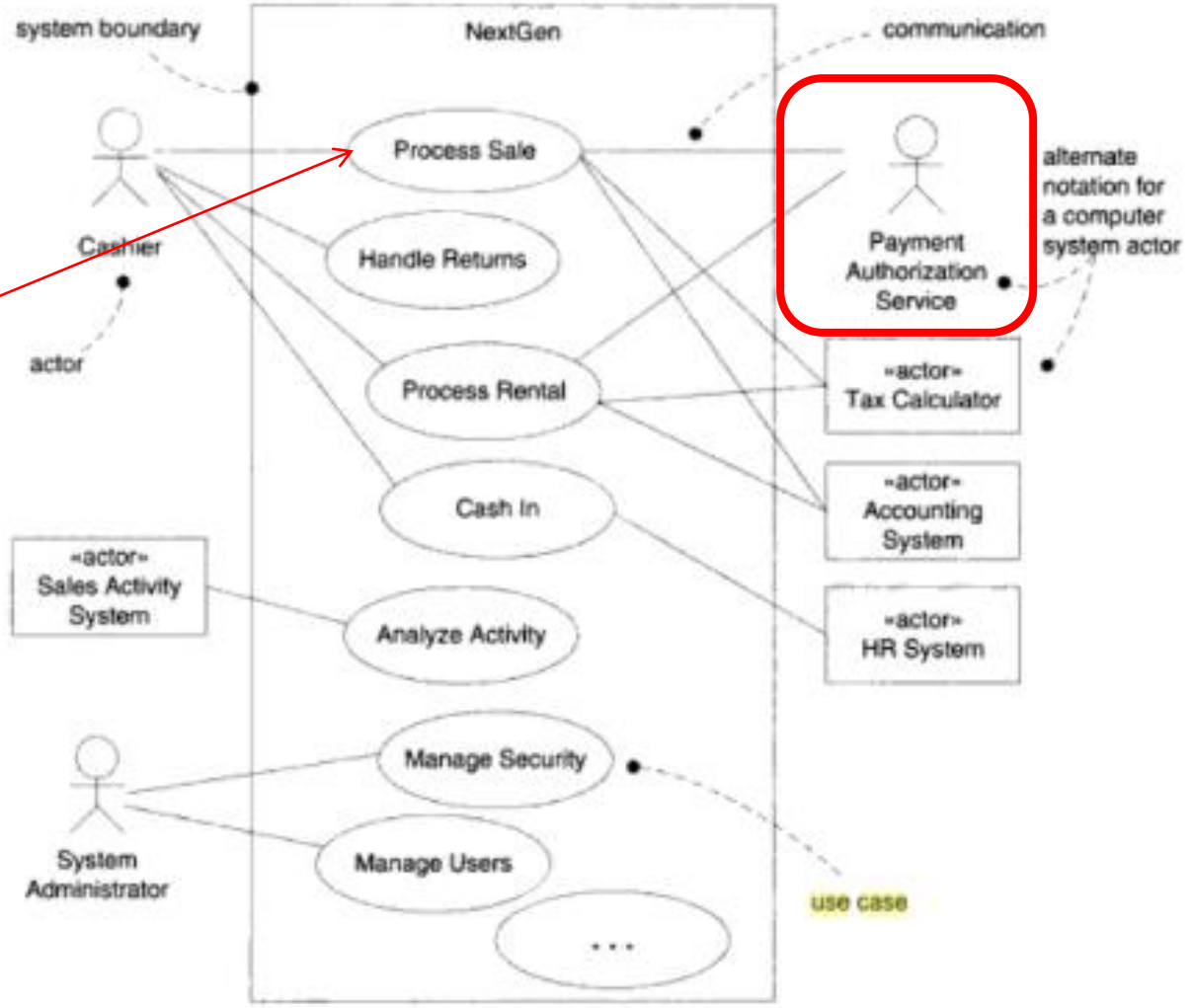


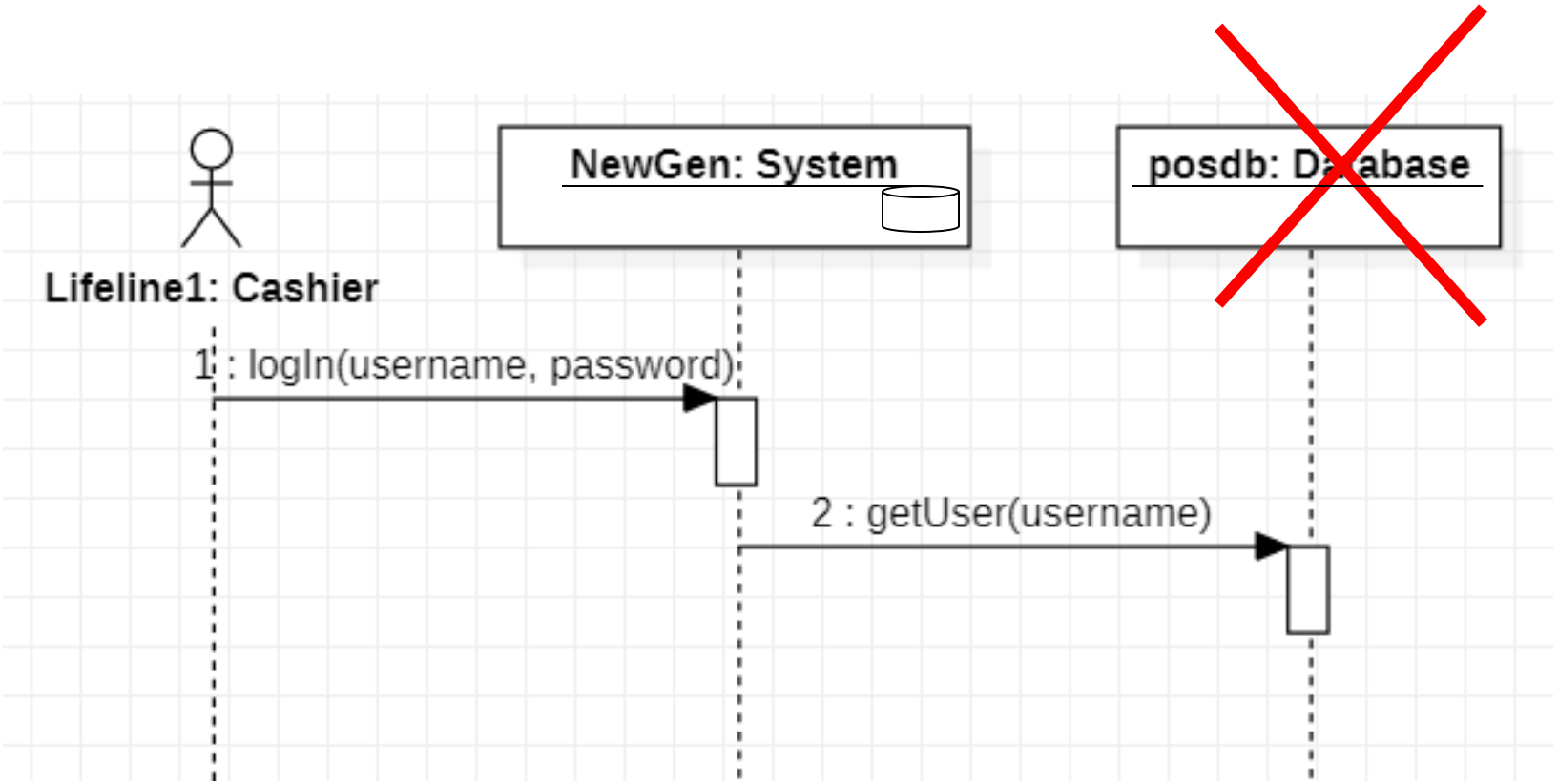
Figure adapted from [3]

Payment Authorization Service

Process Sale

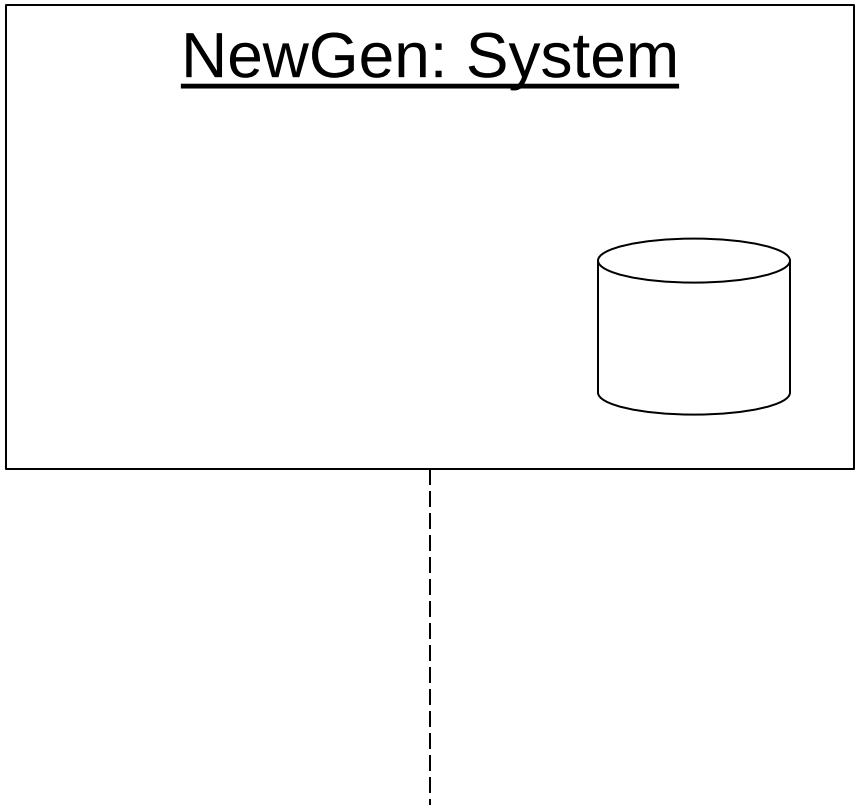


Internal databases



Internal databases

- They are part of the system



How to create an SSD?

- We derive them from basic or alternative flows
  - Specified in the Expanded Use Cases

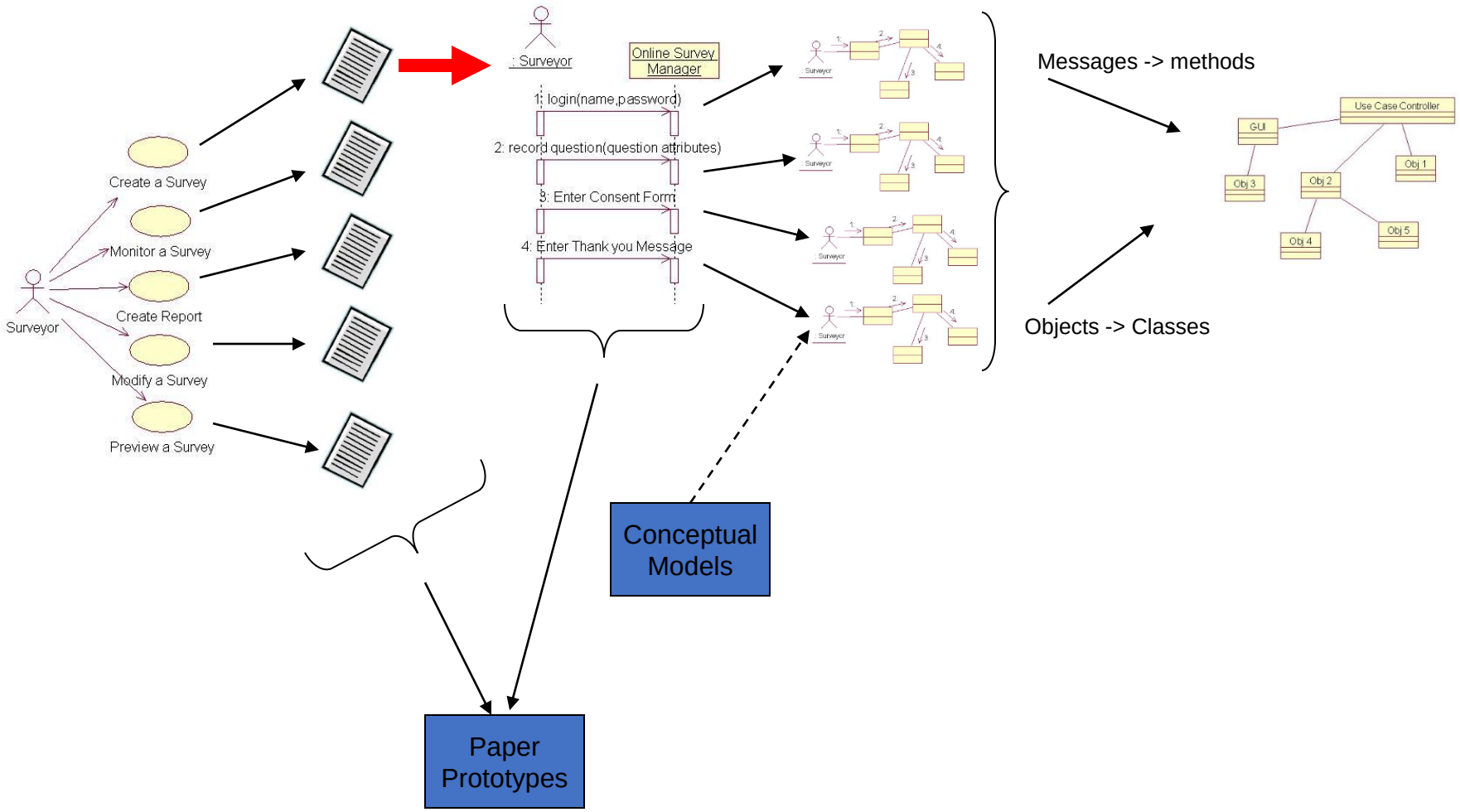
Deriving SSDs from Expanded Ucs

Use Case Diagrams

System Sequence Diagrams

Collaboration Diagrams

Design Class Diagrams



Actor Action	System Response
1. Customer arrives at POS checkout with goods and/or services to purchase.	
2. Cashier starts a new sale.	
3. Cashier enters item identifier *Cashier repeats steps 3-4 until indicates done.*	4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.
5. Cashier finalizes the sale	6. System presents total with taxes calculated
7. Cashier tells Customer the total and asks for payment.	
8. Customer pays and the Cashier introduces the payment	9. System handles the payment
	10. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
	11. System presents receipt.
12. Customer leaves with receipt and goods (if any).	

Examples adapted from [1]

Map basic flow to system events

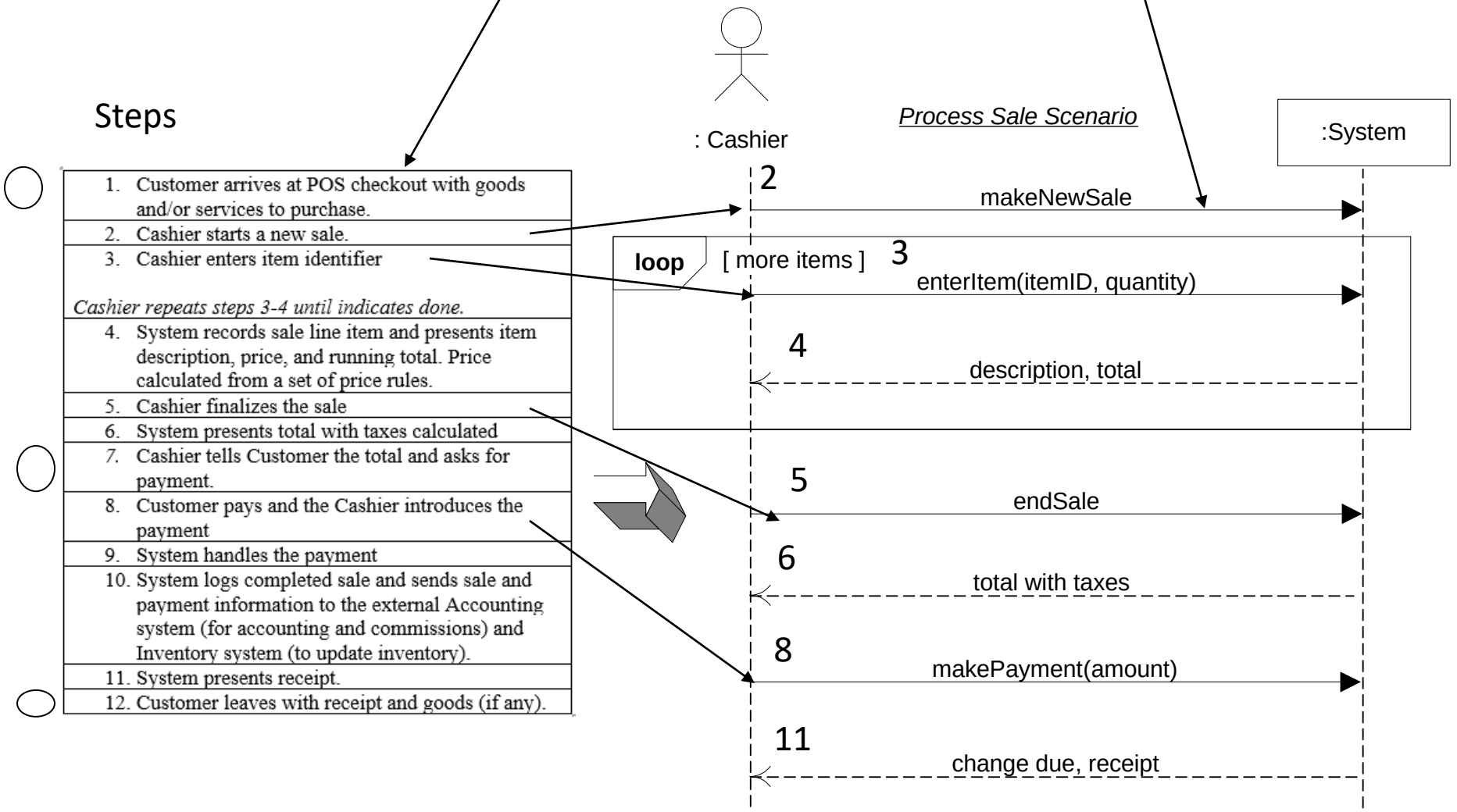
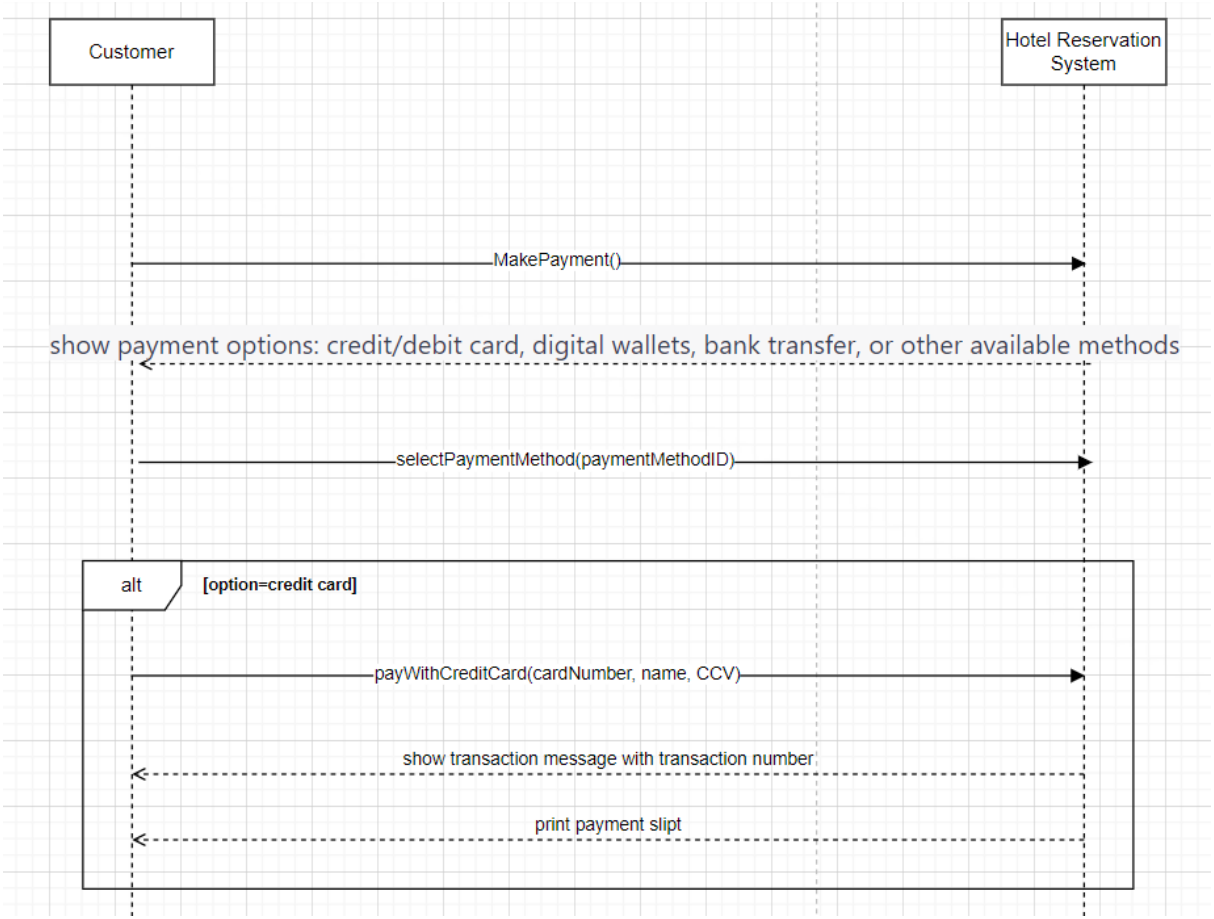


Figure 10.3 from [1]

Note: Some steps happen in the physical world, hence they are not mapped to the SSD

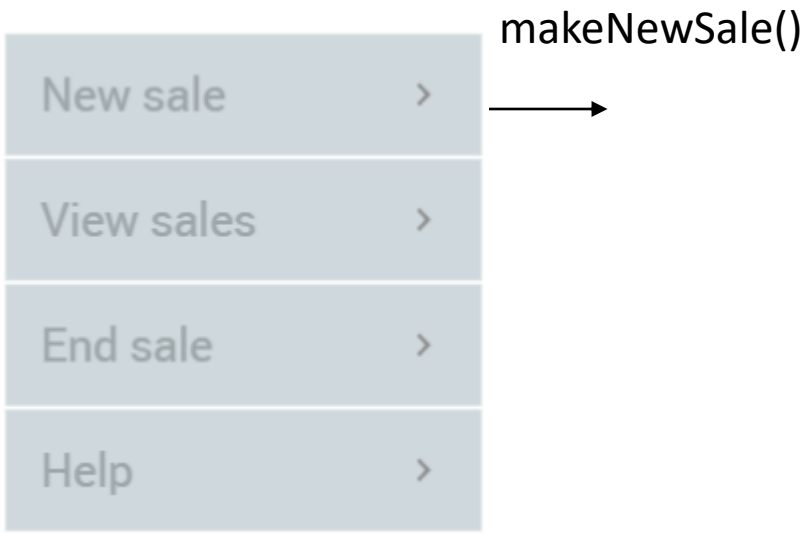
Ejemplo:

Ejercicio

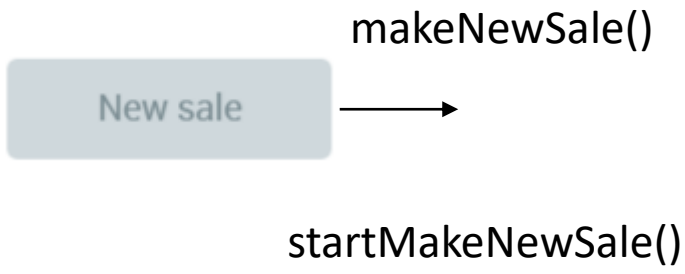

Customer
Hotel Reservation System
MakePayment()
show payment options: credit/debit card, digital wallets, bank transfer, or other available methods
selectPaymentMethod(paymentMethodID)
alt
[option=credit card]
payWithCreditCard(cardNumber, name, CCV)
show transaction message with transaction number
print payment slipt

Interpretation of initial system event

Menu option



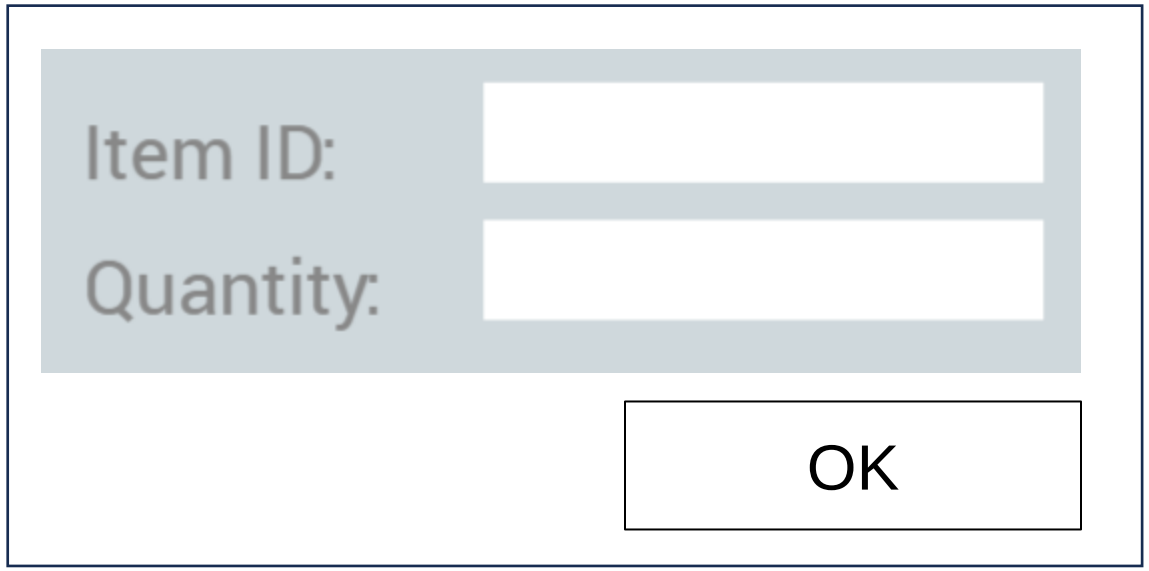
Button



Interpretation of a system event with parameters

- enterItem(itemID, quantity)

parameters in system events mean: data input



Better system event names

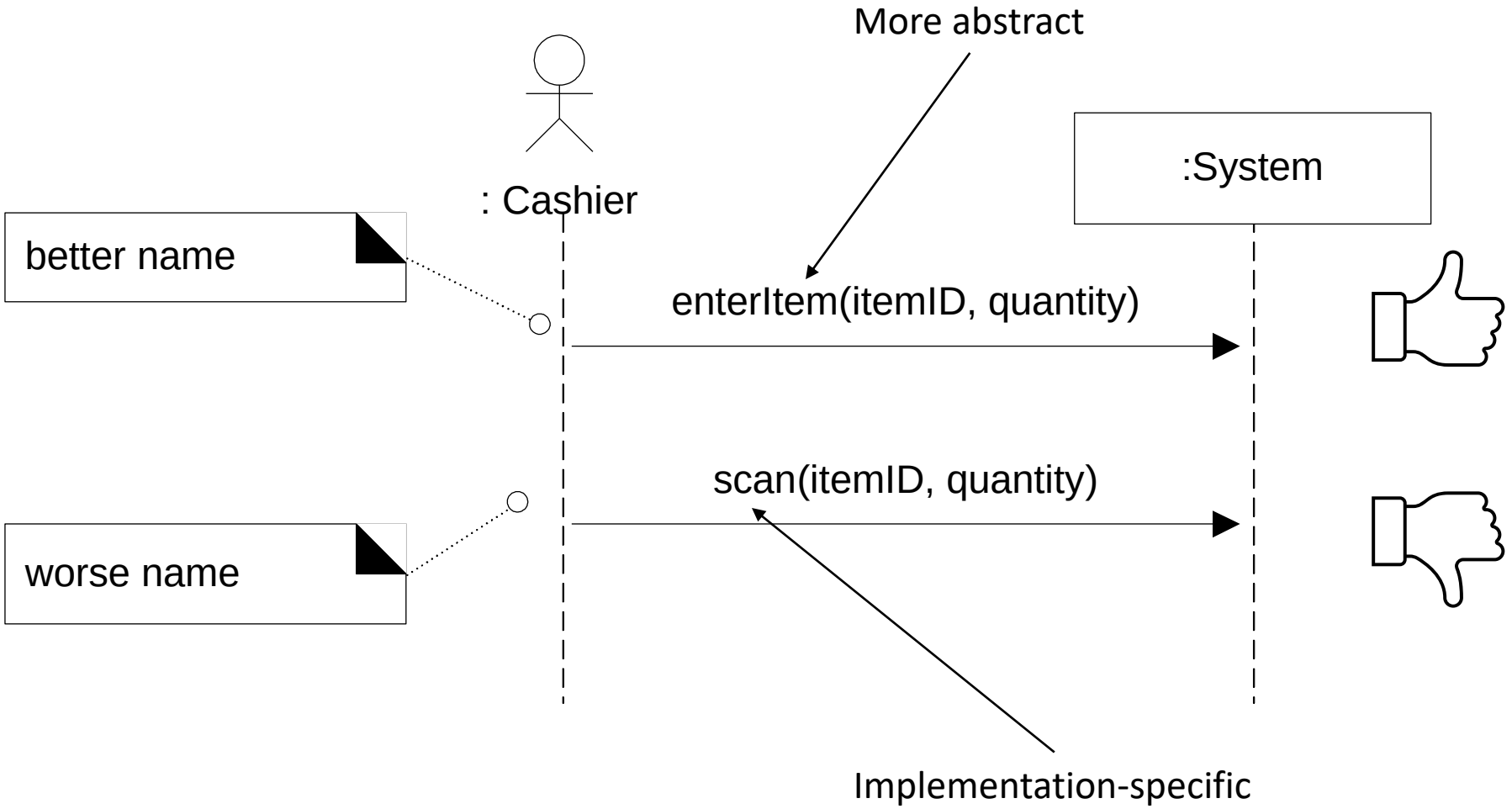


Figure 10.4 from [1]

Interaction frames

Interaction frames

- Loop
- Alt
- Opt
- Other

Loop

Interaction frame

Guard (condition)

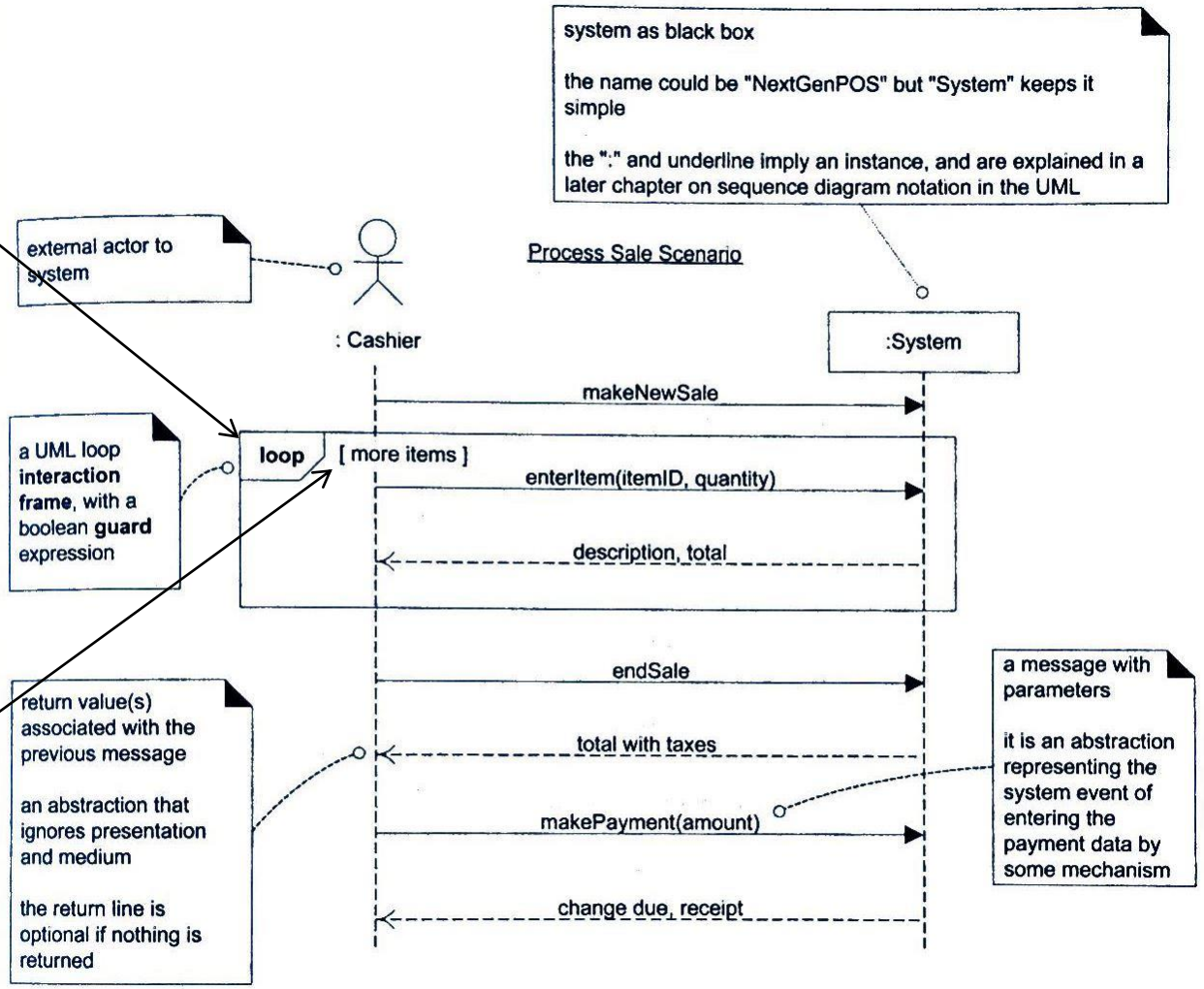


Figure 10.2 SSD for a *Process Sale* scenario.

Figure from [1]

Alt: if-else

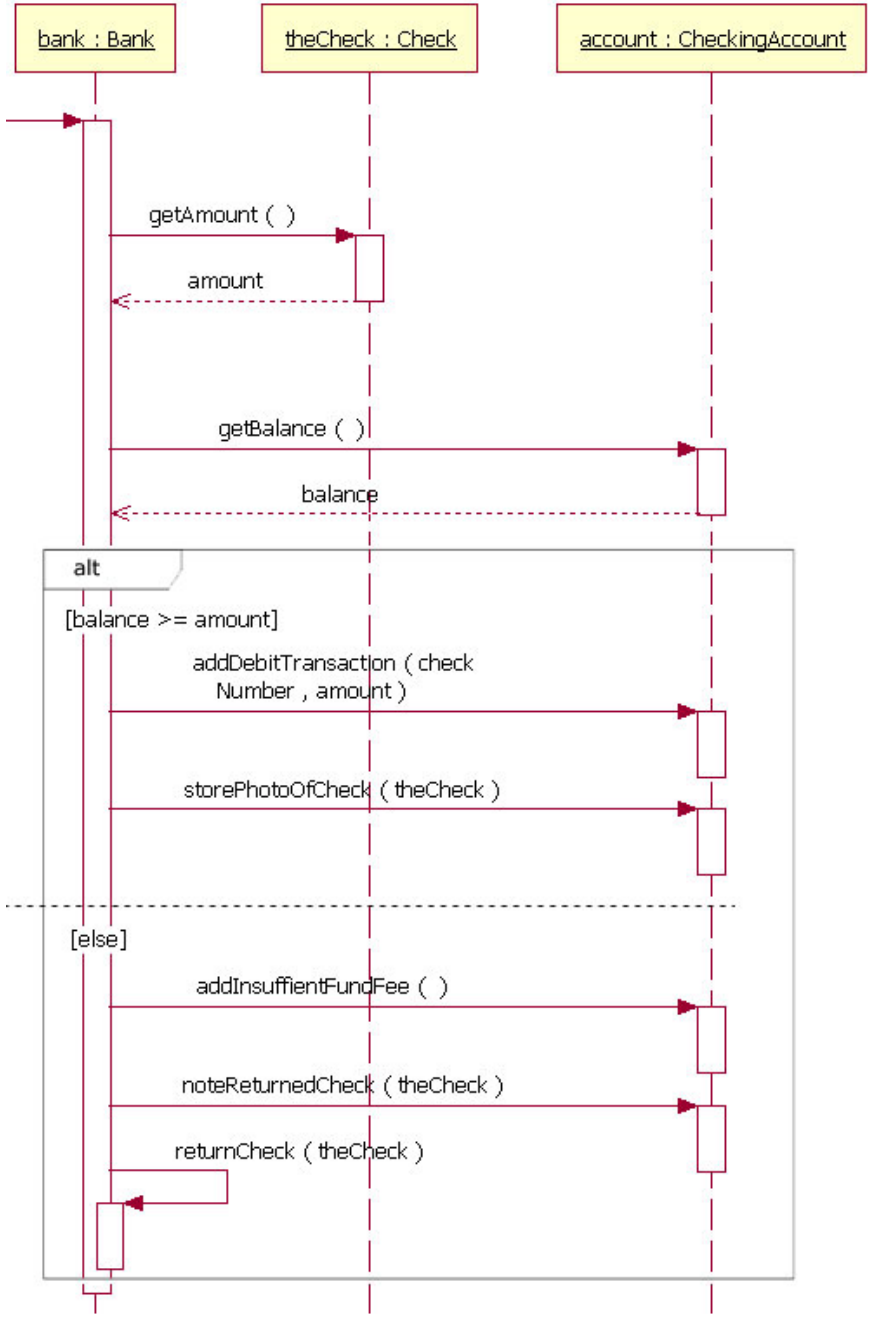


Figure 8 from: http://ima.udg.edu/Docencia/3105200728/DiagSeq.pdf

Opt: switch

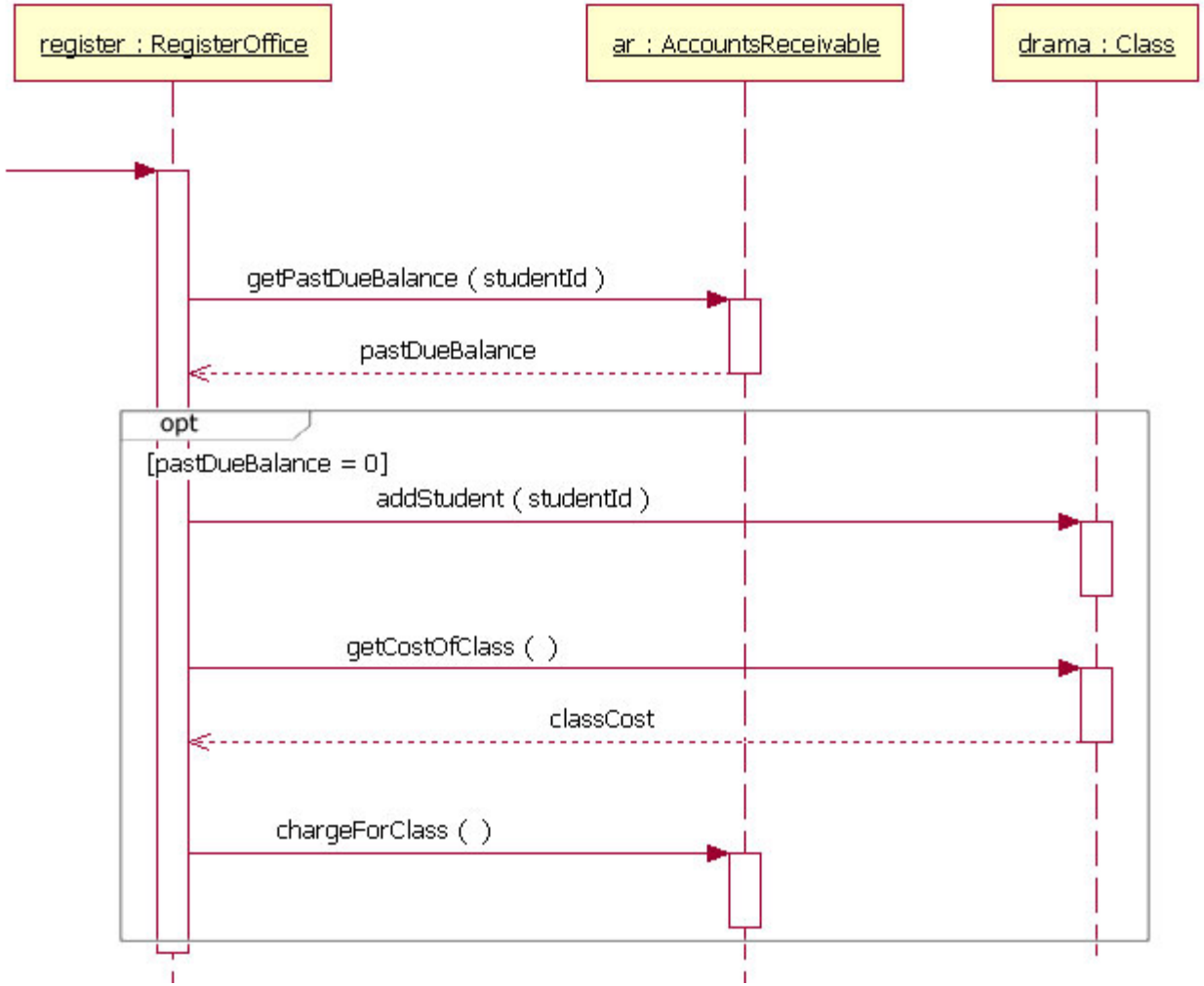


Figure 9 from: http://ima.udg.edu/Docencia/3105200728/DiagSeq.pdf

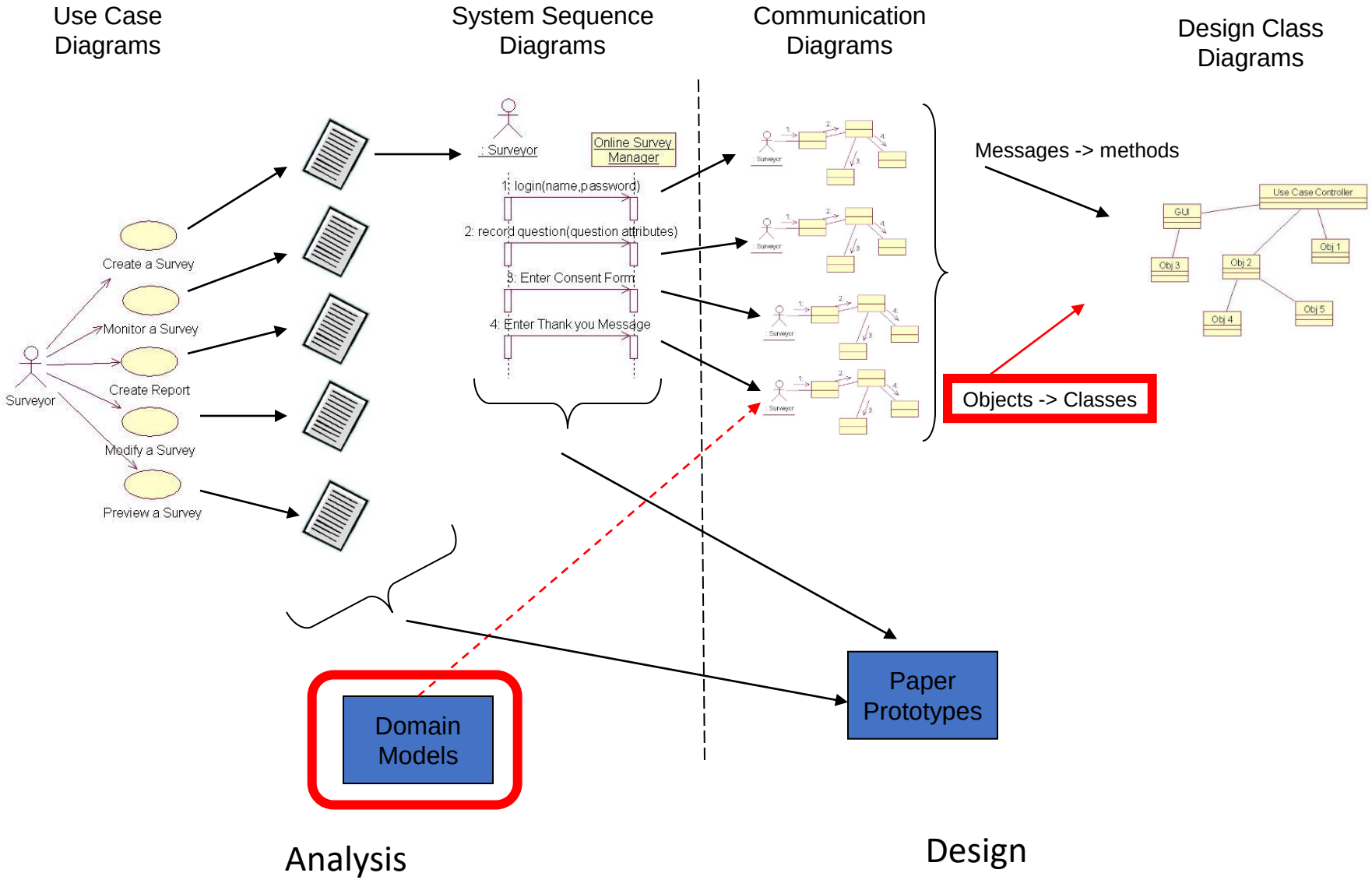
Actividad

- Quiz E09-2 Repaso de diagramas de secuencia de sistema

Domain models

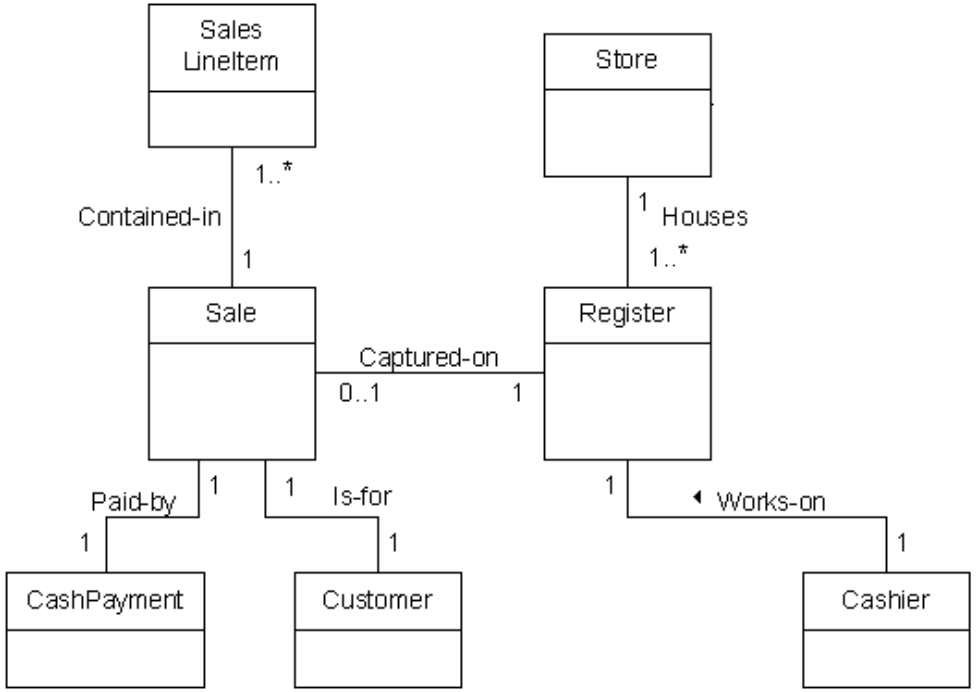
UML process [1]

Interaction diagrams:
- Sequence Diagrams
- Communication Diagrams
- Time Diagrams



What are domain models?

- A type of diagram that shows relevant concepts or objects in a domain



Ejemplo: Conceptos en el dominio de hotelería

Sujetos

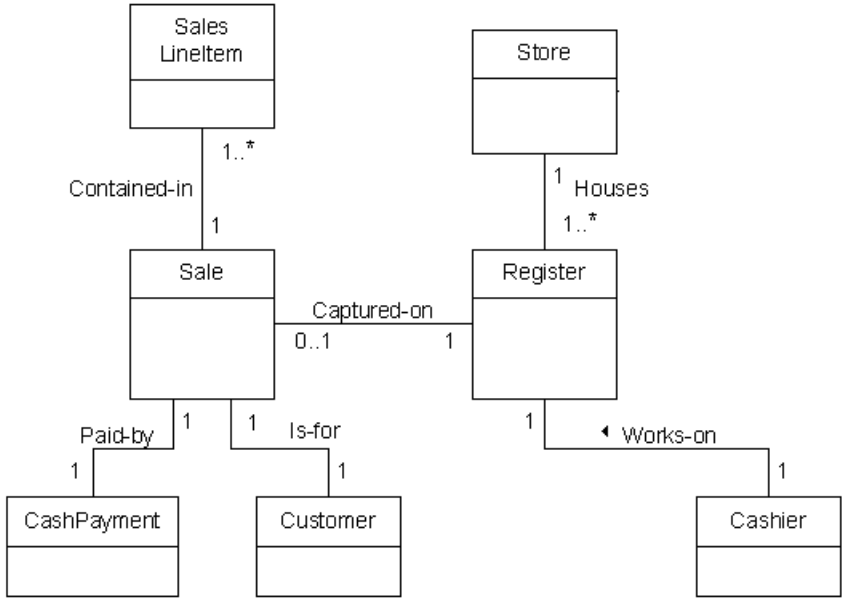
- Cliente
- Hotel
- Reservación
- Estatus
- Cuarto
- Pago
- Ticket
- Servicios
- Catálogo

Verbos

- Hacer reservación
- Pagar
- Ver cuartos disponibles
- Buscar

Domain model elements

- Concepts or objects
- Associations and their multiplicities
- Attributes of objects



Domain models

- Previously called “conceptual models [2]”

notation

Partial domain model

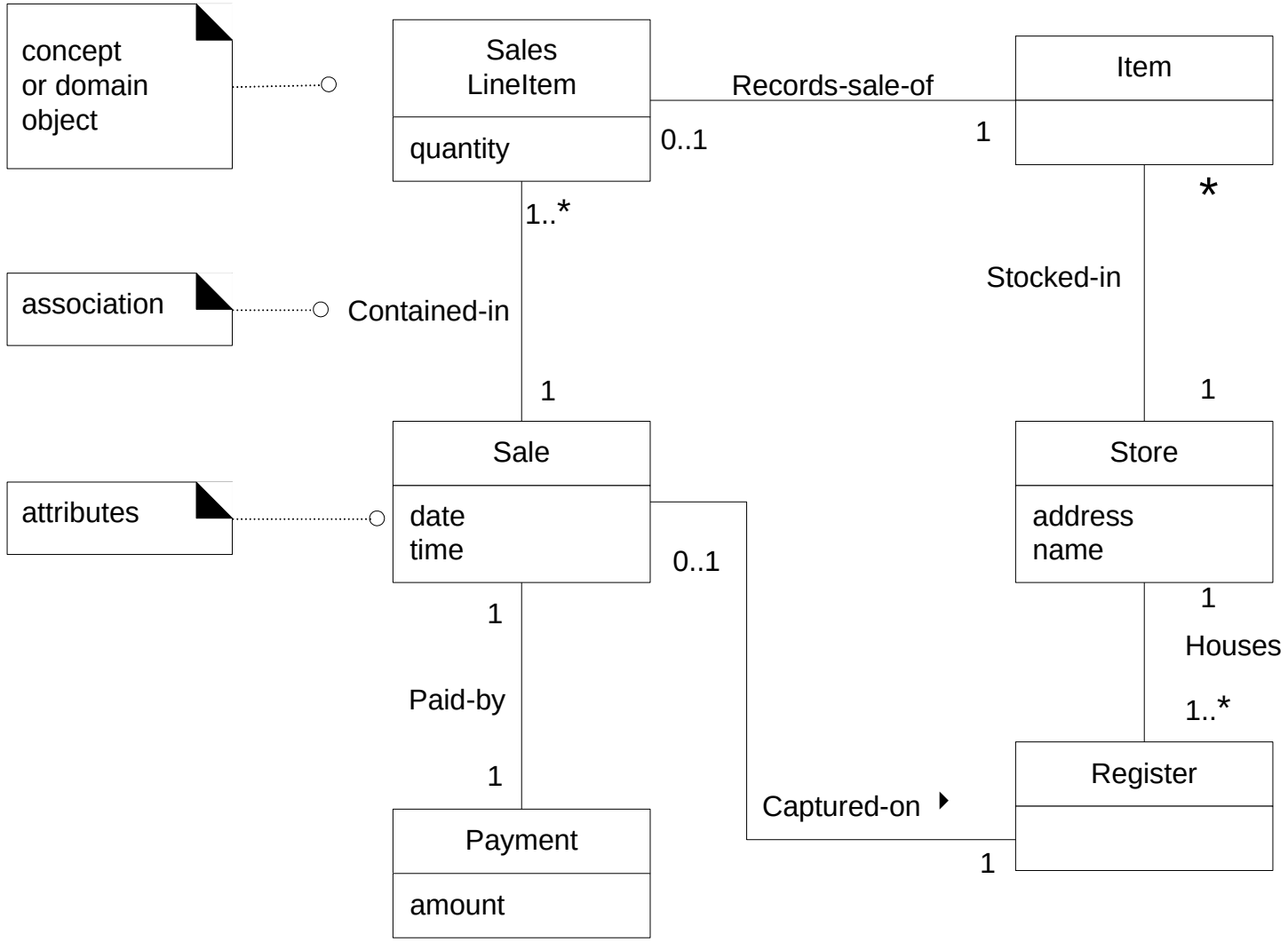


Figure 9.2 from [1, 3]

SalesLineItem

This Photo by Unknown Author is licensed under CC BY-SA-NC

quantity=3

Notebook

How to represent a concept

- A box symbolizing the concept
- Attributes
- No methods

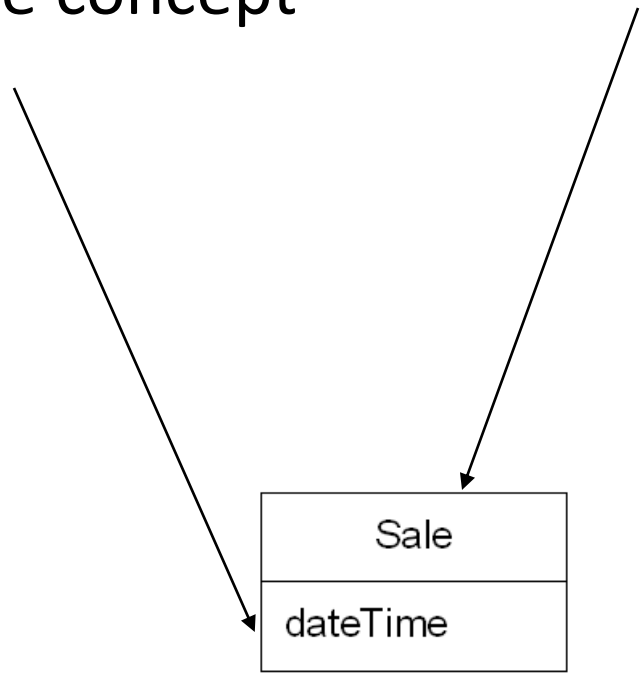


Figure 9.3 from [1, 3]

A visual dictionary

- A domain model can be viewed as a “visual dictionary”

Real-world concepts, not software classes

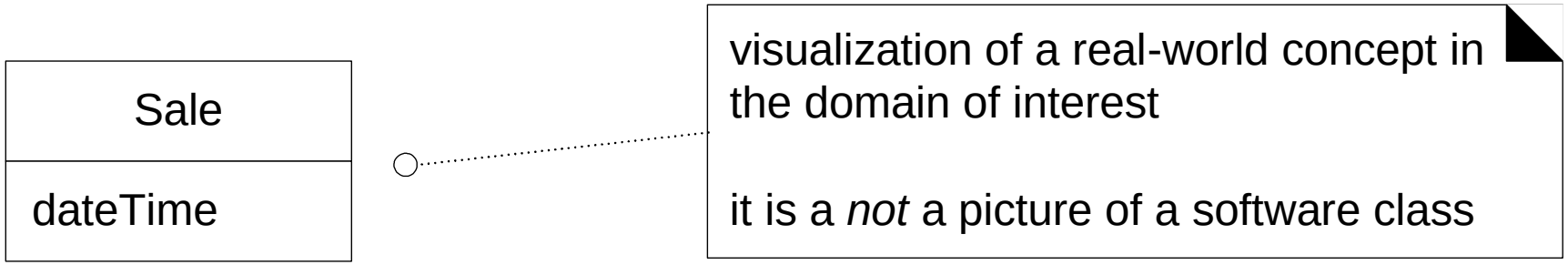


Figure 9.3 from [1, 3]

Avoid software classes/artifacts

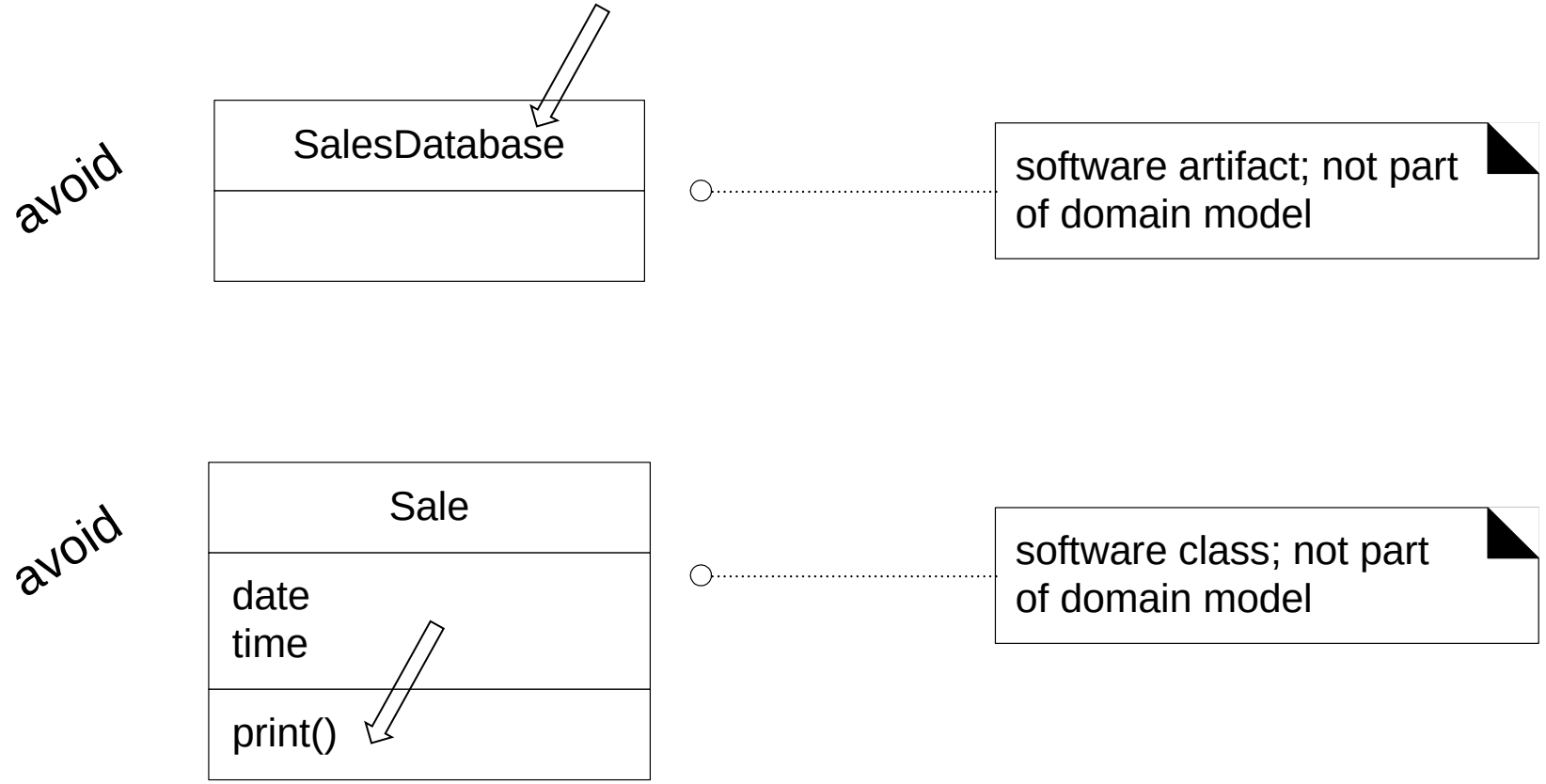


Figure 9.4 from [1, 3]

Inspiration from domain models

Getting inspired from the domain model

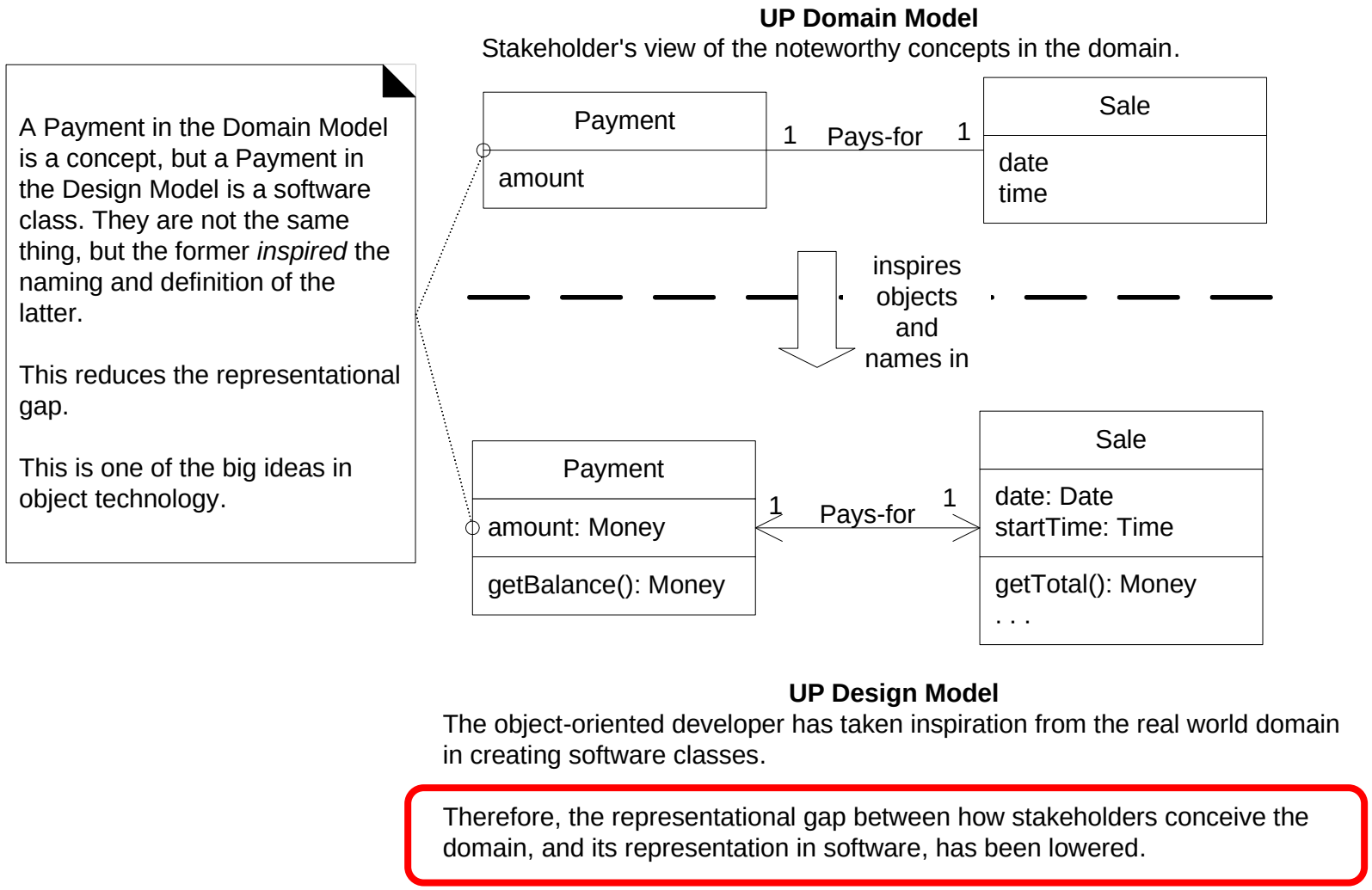
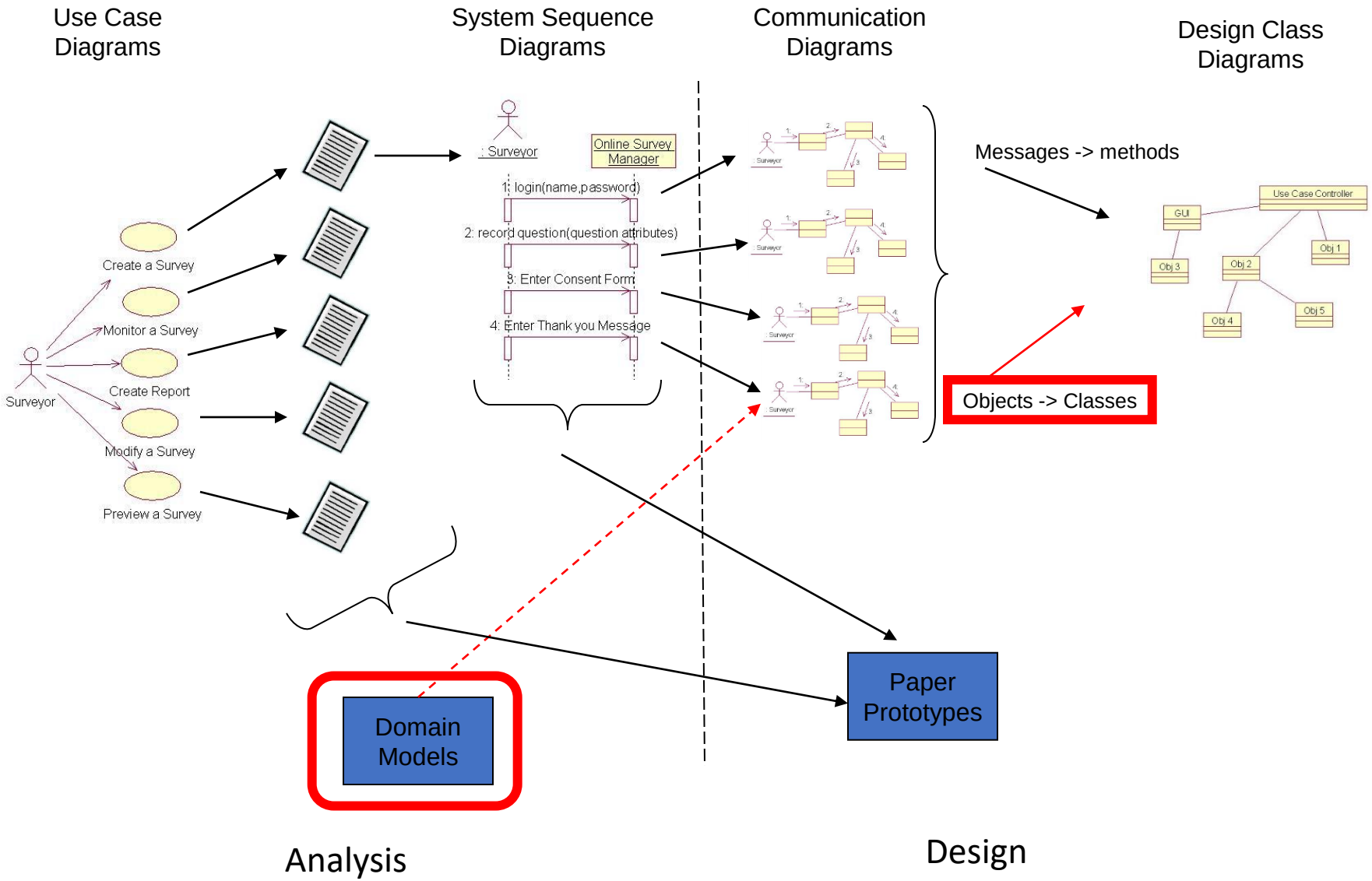


Figure 9.6 from [1, 3]

Advantages?

- Code easier to understand
- Code easier to maintain

UML process [1]



Concepts-classes relationship

- Concepts in domain models inspire objects in communication diagrams

- Objects in communication diagrams are generalized as classes in design class diagrams

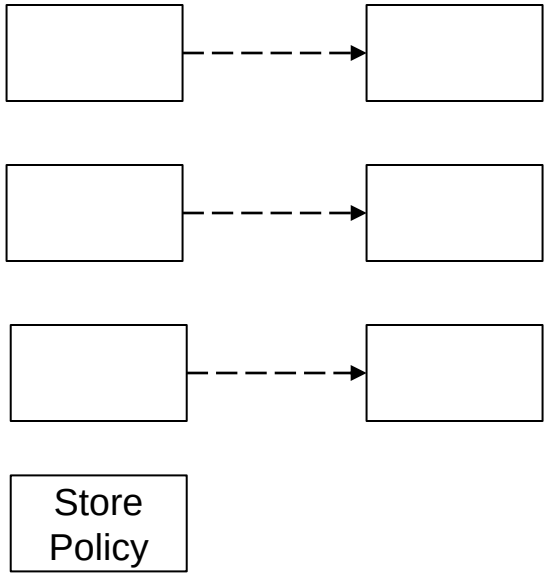
Not necessarily a 1-to-1 mapping

- Not all concepts in the domain model may suggest a software class
- Not all software classes have a mapped real-world concept

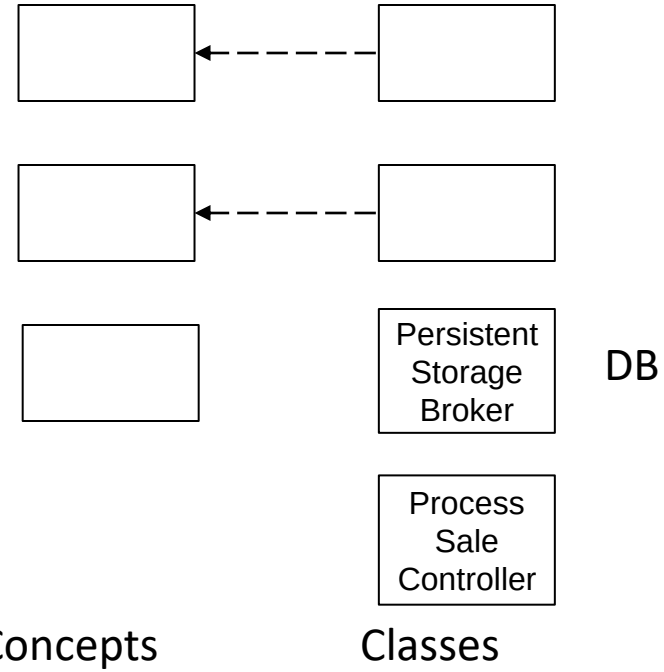
- The purpose of the domain model is to understand the problem better, visually

Not necessarily a 1-to-1 mapping

Not all concepts in the domain model suggest software classes



Not all software classes have a mapped real-world concept



How to identify concepts

Partial domain model in the store domain

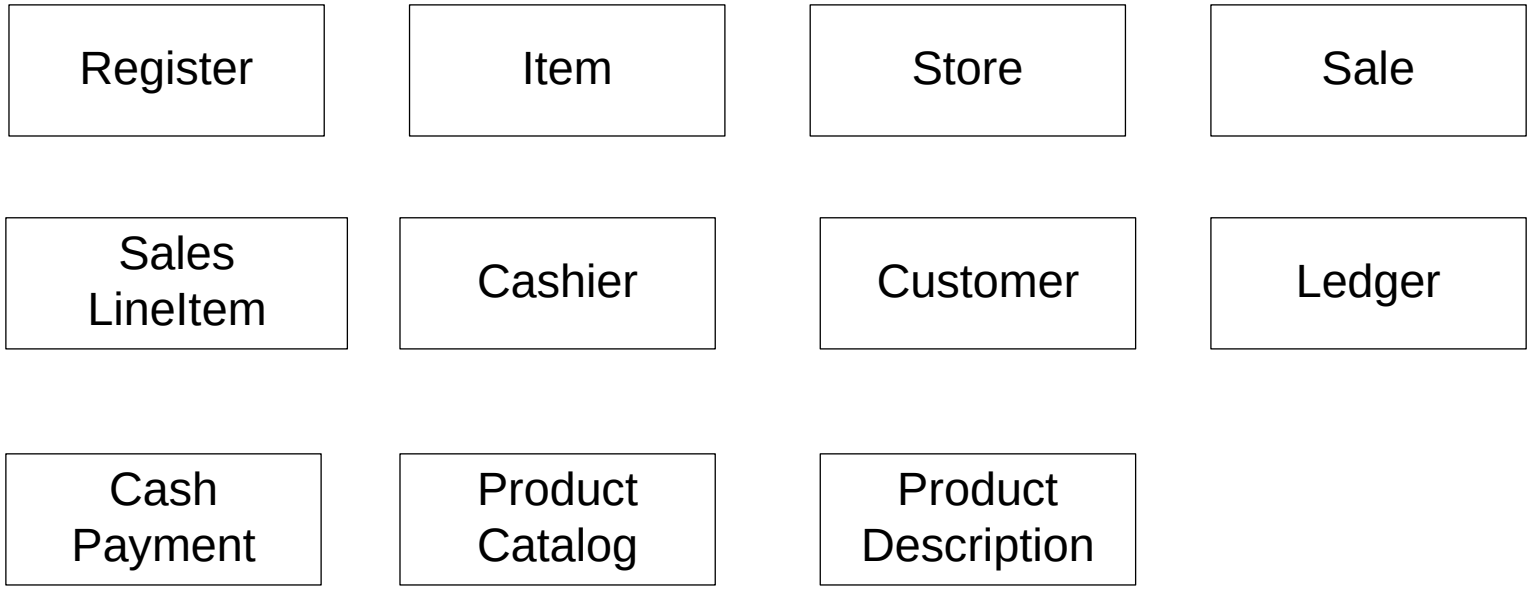


Figure 9.7 from [1, 3]

How to identify concepts in a domain?[2]

- 1. Use a list of conceptual categories
- 2. Identify nominal phrases in available documentation
- 3. Use of catalogs of partial domain models by experts

Conceptual categories [2]

#	Conceptual category	Examples
1	physical or tangible objects	Register, Airplane
2	specifications, designs, or descriptions of things	ProductSpecification, FlightDescription
3	place	Store, Airport
4	transactions	Sale, Payment, Reservation
5	transaction line items	SalesLineItem
6	roles of people	Cashier, Pilot
7	containers of other things	Store, Bin, Airplane
8	things in a container	Item, Passenger
9	other computer or electro-mechanical systems external to the system	CreditPaymentAuthorizationSystem, AirTrafficControl
10	abstract noun concepts	Hunger, Acrophobia

EspecificacionCuarto

AreaHotel

Conceptual categories [2]…

#	Conceptual category	Examples
11	organizations	SalesDepartment, AirlineCompany
12	events	Sale, Payment, Meeting, Flight, Crash, Landing
13	processes (often not represented as a concept, but may be)	SellingAProduct, BookingASeat
14	rules and policies	RefundPolicy, CancellationPolicy
15	catalogs	ProductCatalog, PartsCatalog
16	records of finance, work, contracts, legal matters	Receipt, Ledger, EmploymentContract, MaintenanceLog
17	financial instruments and services	LineOfCredit, Stock
18	manuals, documents, reference, papers, books	DailyPriceChangeList, RepairManual

How to identify concepts in a domain?[2]

- 1. Use a list of conceptual categories
- 2. Identify nominal phrases in available documentation
- 3. Use of catalogs of partial domain models by experts

Nouns → Concepts/Ideas
Verbs → Associations between concepts/ideas

Documentation examples: Use cases, SRS, documentation provided by the customer

2. Extracting nominal phrases

- From documentation that you have
  - E.g. expanded use cases, SRS, manuals, etc.

Example: Process Sale UC

Main Success Scenario (or Basic Flow):

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale.**
3. **Cashier** enters **item identifier.**
4. System records **sale line item** and presents **item description, price,** and running **total.** Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment.**
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions)** and **Inventory** systems (to update inventory).
9. System presents **receipt.**
10. Customer leaves with receipt and goods (if any).

Example: Process Sale UC

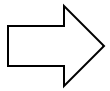
Extensions (or Alternative Flows):

7a. Paying by cash:

2. System presents the **balance due,** and releases the **cash drawer.**
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

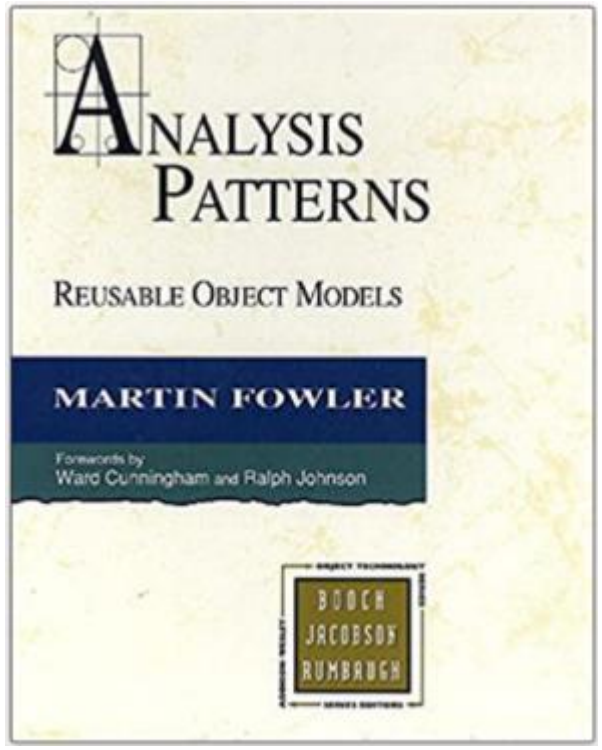
How to identify concepts in a domain?[2]

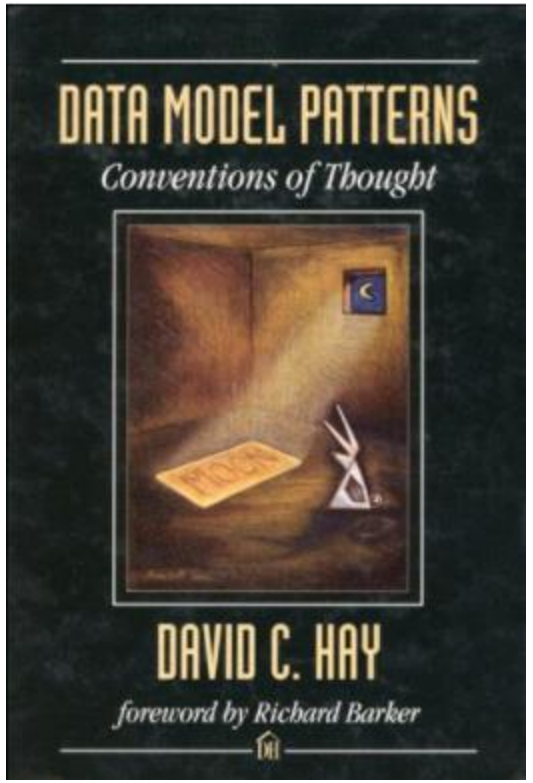
- 1. Use a list of conceptual categories
- 2. Identify nominal phrases in available documentation
- 3. Use of catalogs of partial domain models by experts



3. Catalogs of concepts

- Created by experts (Fowler, 1996; Hay 1996)





Some domains (Fowler, 1996)

- Accountability
- Observations and measurements
- Inventory and accounting
- Planning
- Trading

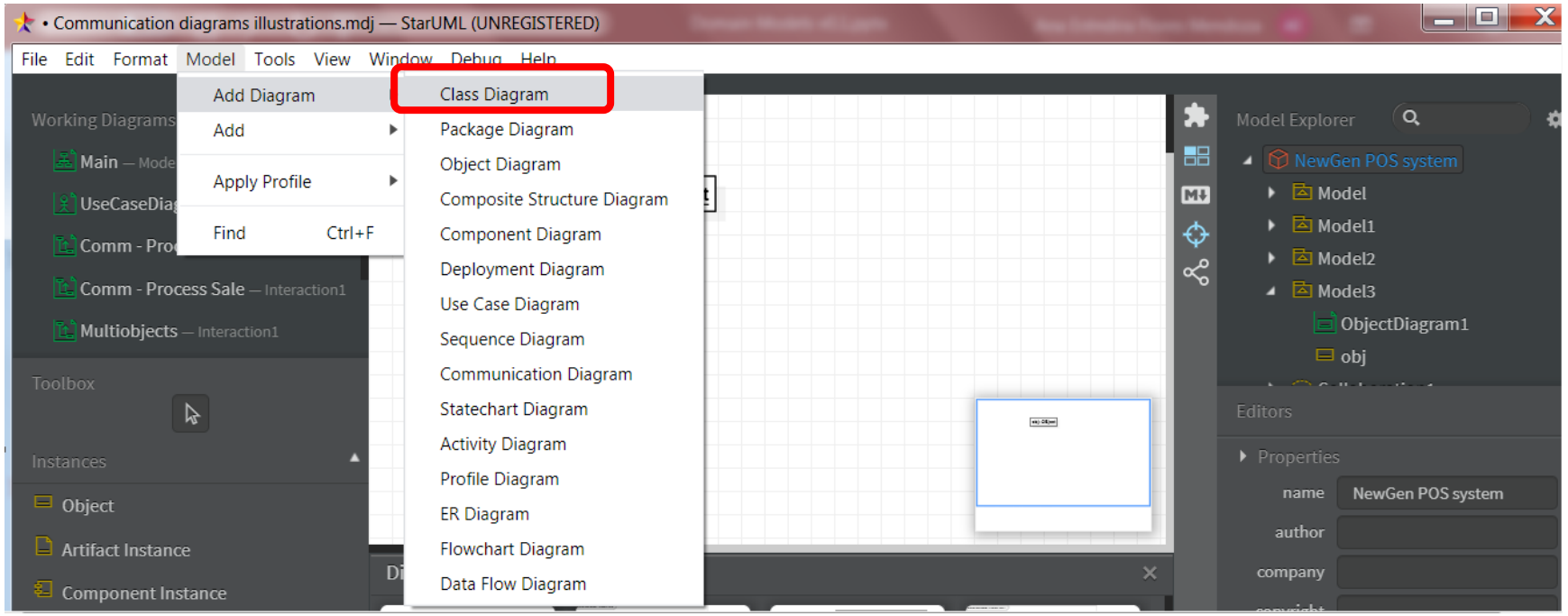
How to make a domain model?

- 1.List candidate concepts using 1 or more strategies we mentioned
- 2. Draw them in a domain model
- 3. Add associations between concepts
- 4. Add attributes to concepts

How to make a domain model?

- 1.List candidate concepts using 1 or more strategies we mentioned
- 2. Draw them in a domain model
- 3. Add associations between concepts
- 4. Add attributes to concepts

We use a class diagram to create domain models



How to make a domain model?

- 1.List candidate concepts using 1 or more strategies we mentioned

- 2. Draw them in a domain model
- 3. Add associations between concepts
- 4. Add attributes to concepts

Associations

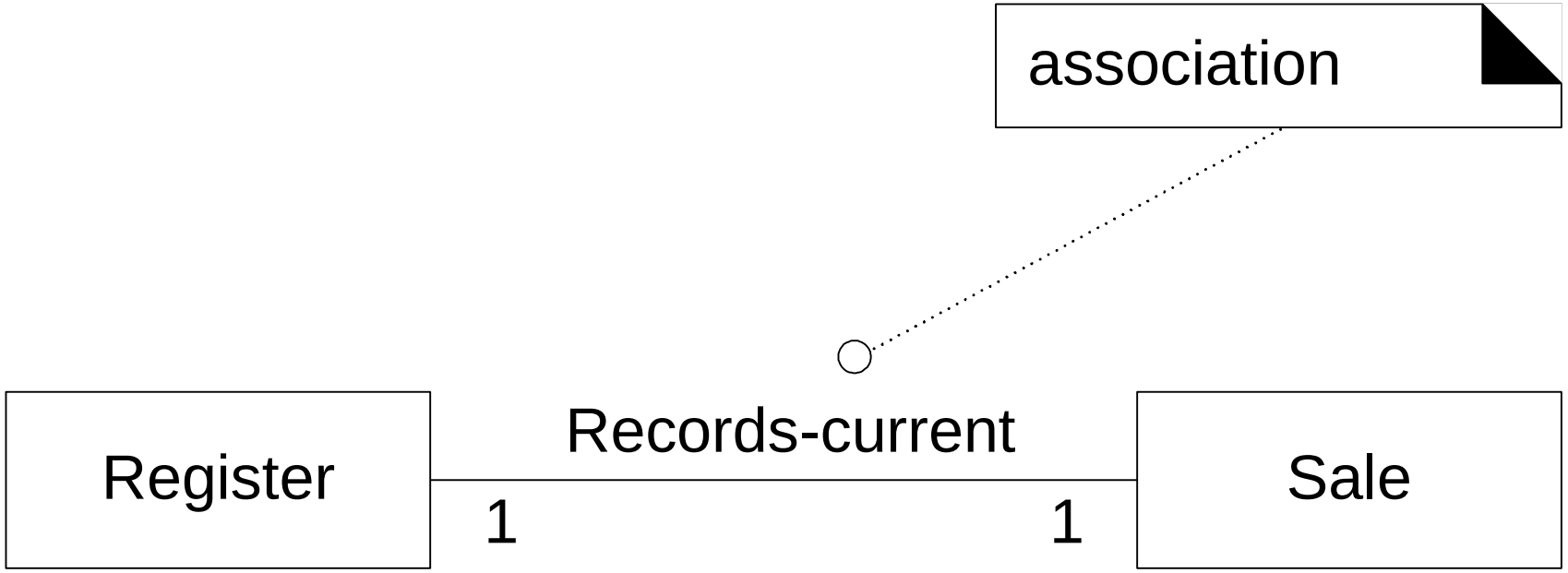


Figure 9.11 from [1, 3]

Example: Partial domain mode

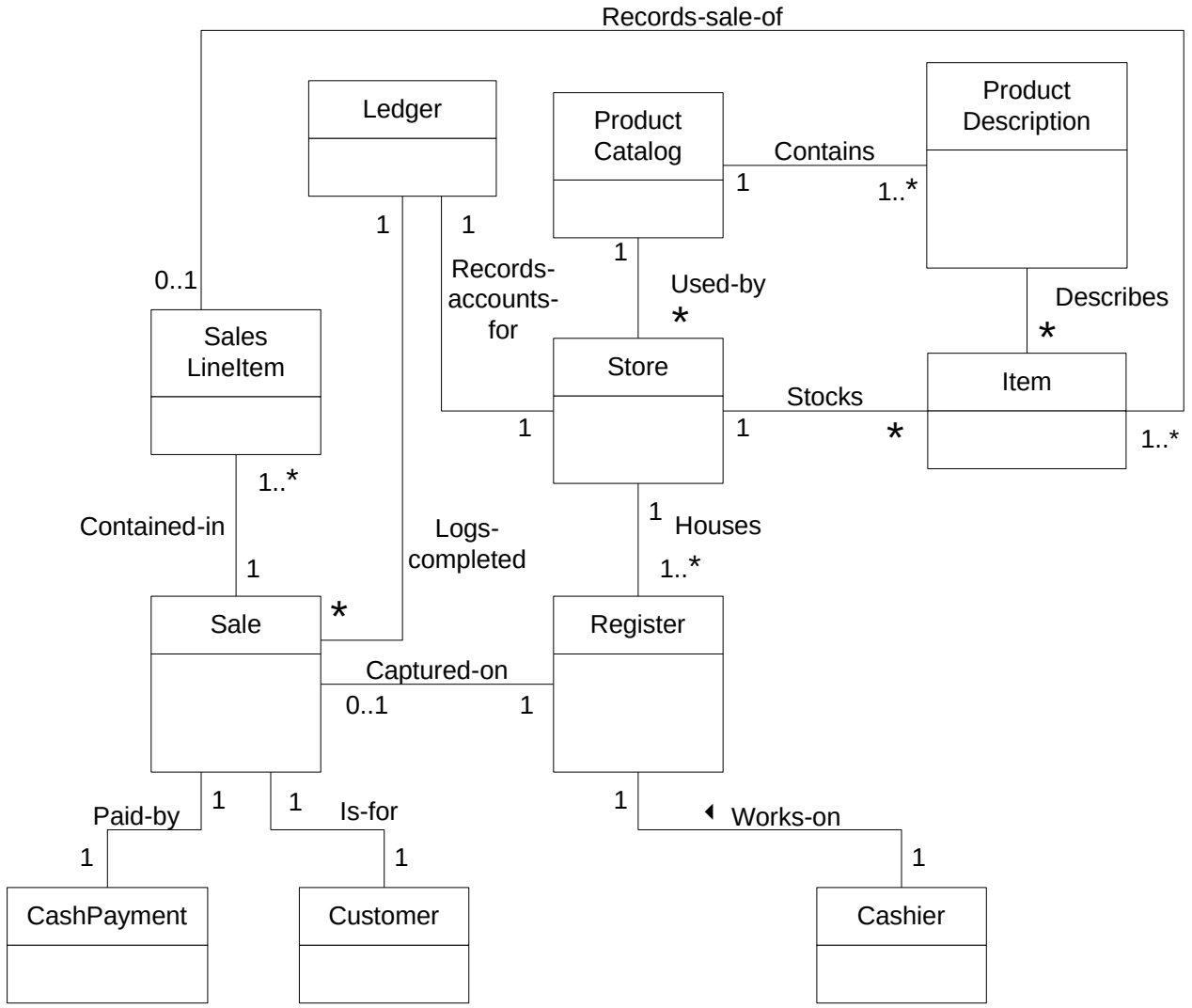


Figure 9.17 from [1, 3]

Association categories [2]

#	Association category	Examples
1	A is a physical part of B	Drawer -- Register (or more specifically, a POST), Wing -- Airplane
2	A is a logical part of B	SalesLineItem -- Sale, FlightLeg -- FlightRoute
3	A is physically cotained in/on B	Register -- Store, Item -- Shelf, Passenger -- Airplane
4	A is logically contained in B	ItemDescription -- Catalog, Flight -- FlightSchedule
5	A is a description for B	ItemDescription -- Item, FlightDescription -- Flight
6	A is a line item of a transaction or report B	SalesLineItem -- Sale, MaintenanceJob -- MaintenanceLog
7	A is known/logged/recorded/reported/captured in B	Sale -- Register, Reservation -- FlightManifest
8	A is a member of B	Cashier -- Store, Pilot -- Airline

Association categories [2] …

#	Association category	Examples
9	A is an organizational subunit of B	Department -- Store, Maintenance -- Airline
10	A uses or manages B	Cashier -- Register, Pilot -- Airplane
11	A communicates with B	Customer -- Cashier, Reservation Agent -- Passenger
12	A is related to a transaction B	Customer -- Payment, Passenger -- Ticket
13	A is a transaction related to another transaction B	Payment -- Sale, Reservation -- Cancellation
14	A is next to B	SalesLineItem -- SalesLineItem, City -- City
15	A is owned by B	Register -- Store, Plane -- Airline
16	A is an event related to B	Sale -- Customer, Sale -- Store, Departure -- Flight

Reading direction “ >” ▶

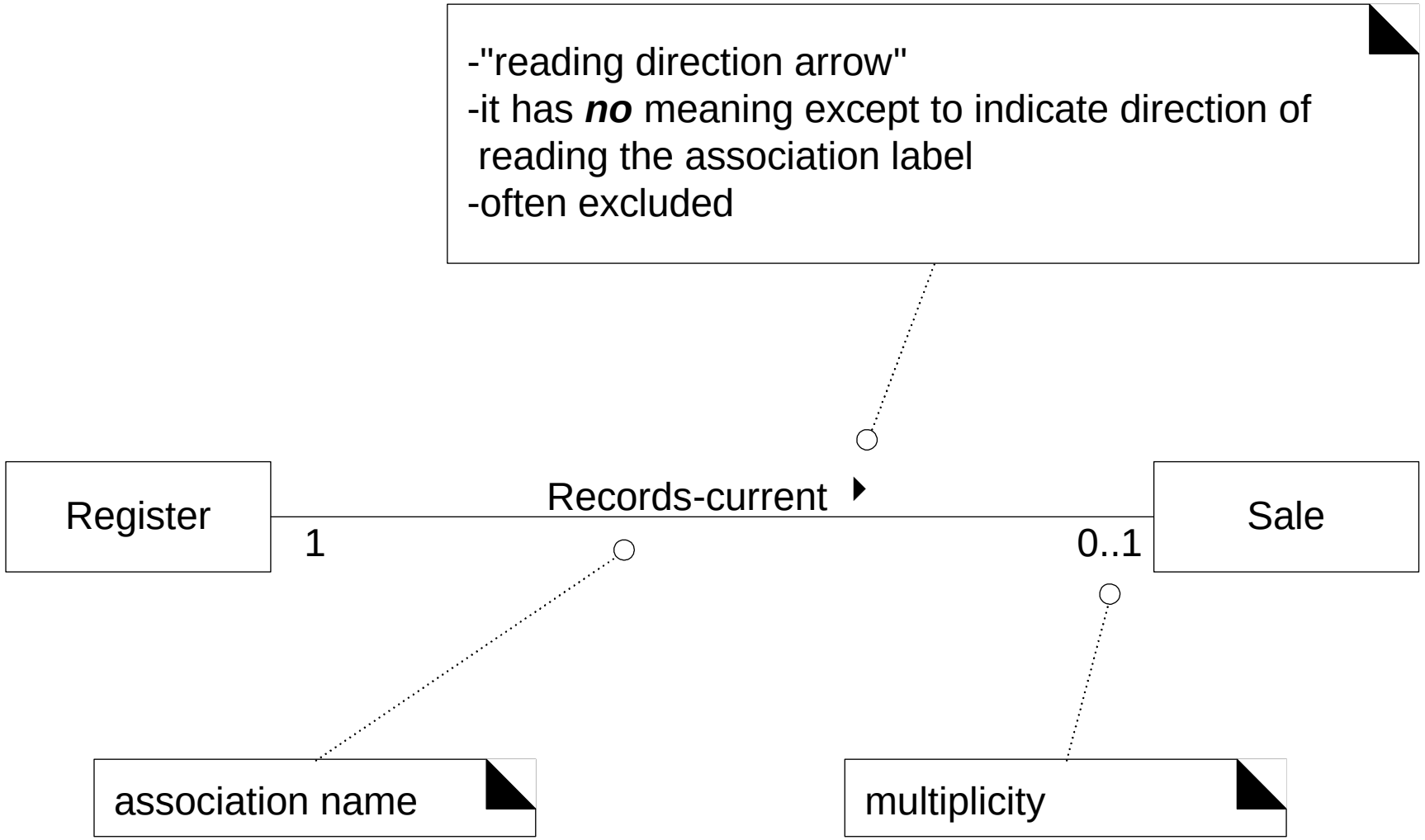


Figure 9.12 from [1, 3]

Multiplicity

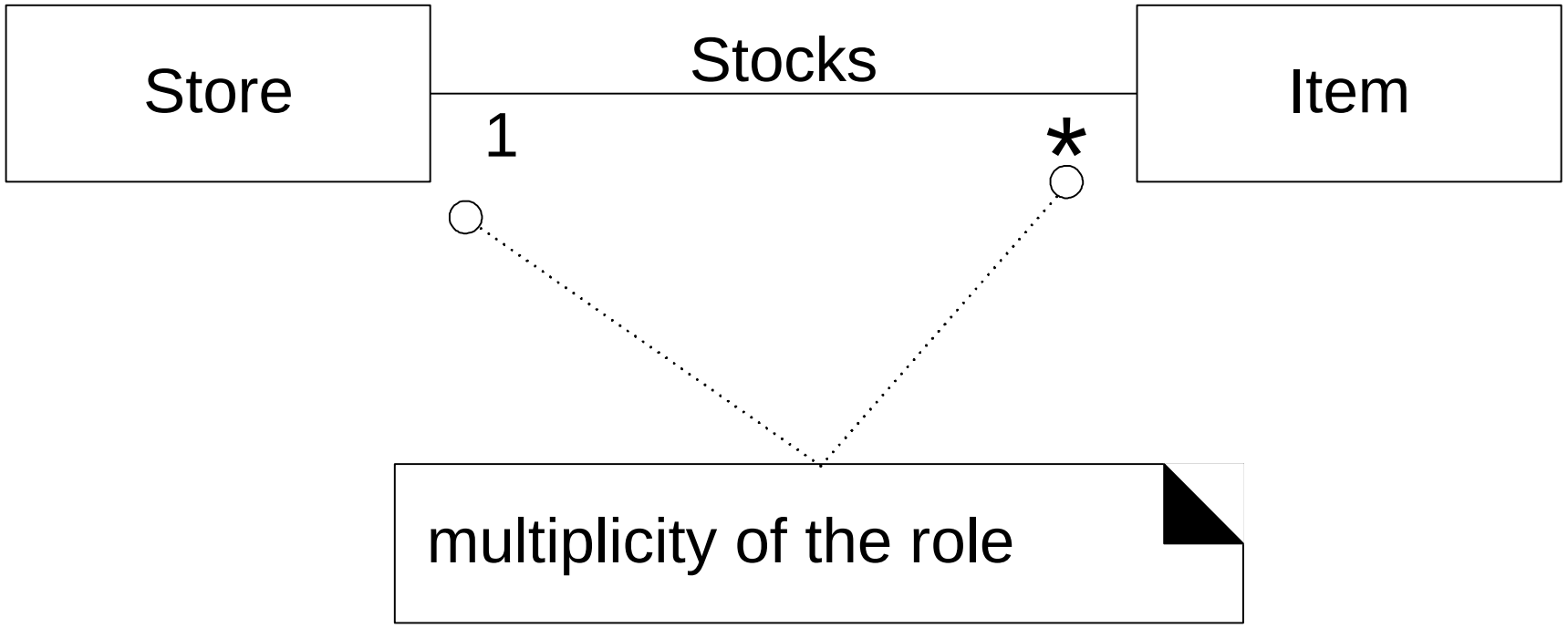


Figure 9.13 from [1, 3]

Multiplicity

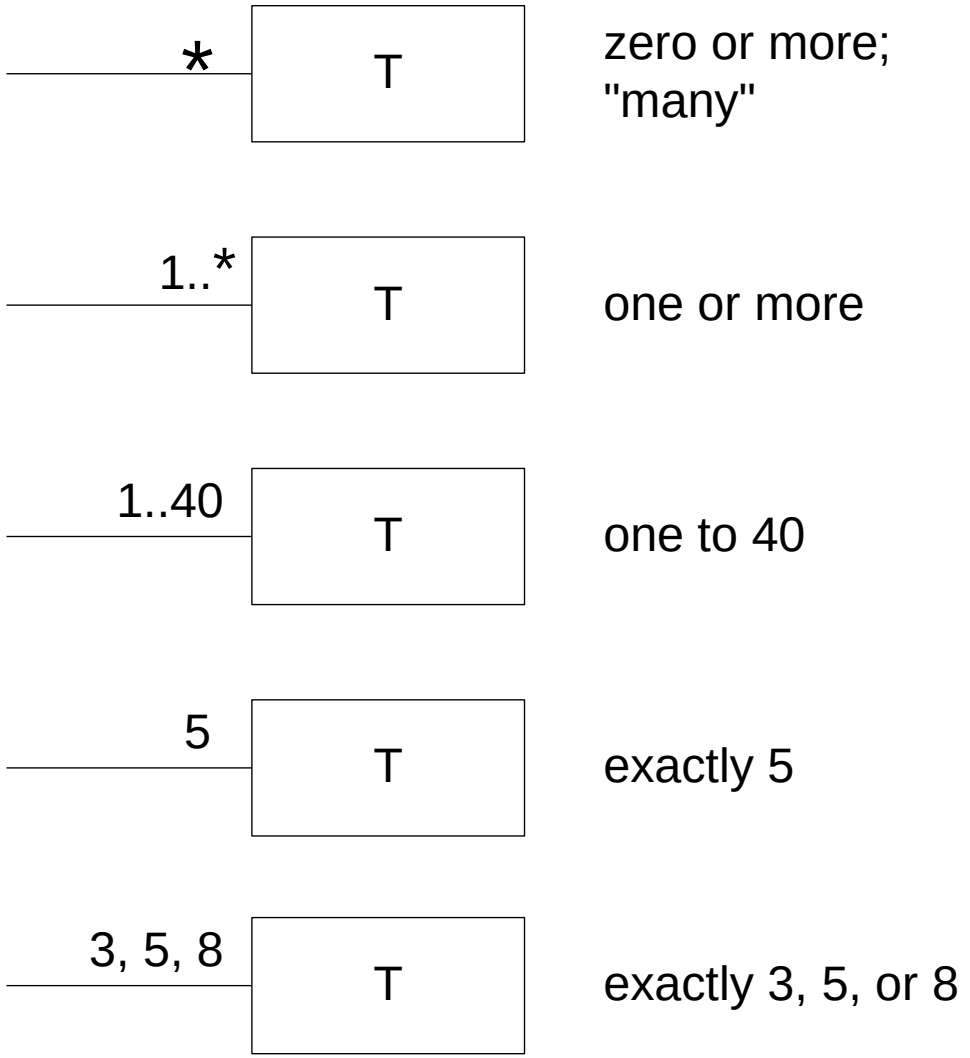


Figure 9.14 from [1, 3]

Activity: Identify associations for the class project

- Work in teams
- Use the same worksheet

How to make a domain model?

- 1.List candidate concepts using 1 or more strategies we mentioned
- 2. Draw them in a domain model
- 3. Add associations between concepts
- 4. Add attributes to concepts

Example

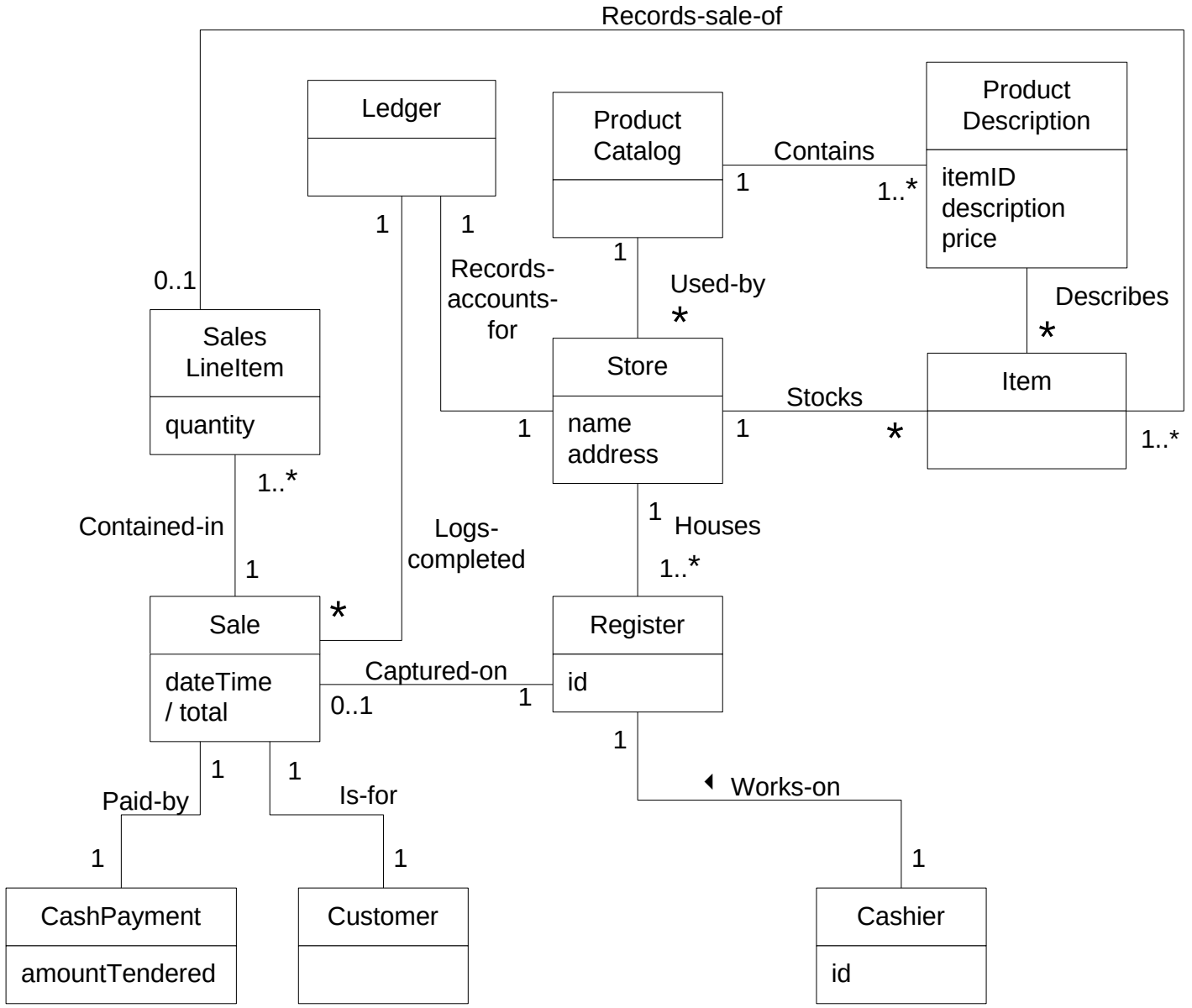


Figure 9.27 from [1, 3]

No foreign keys

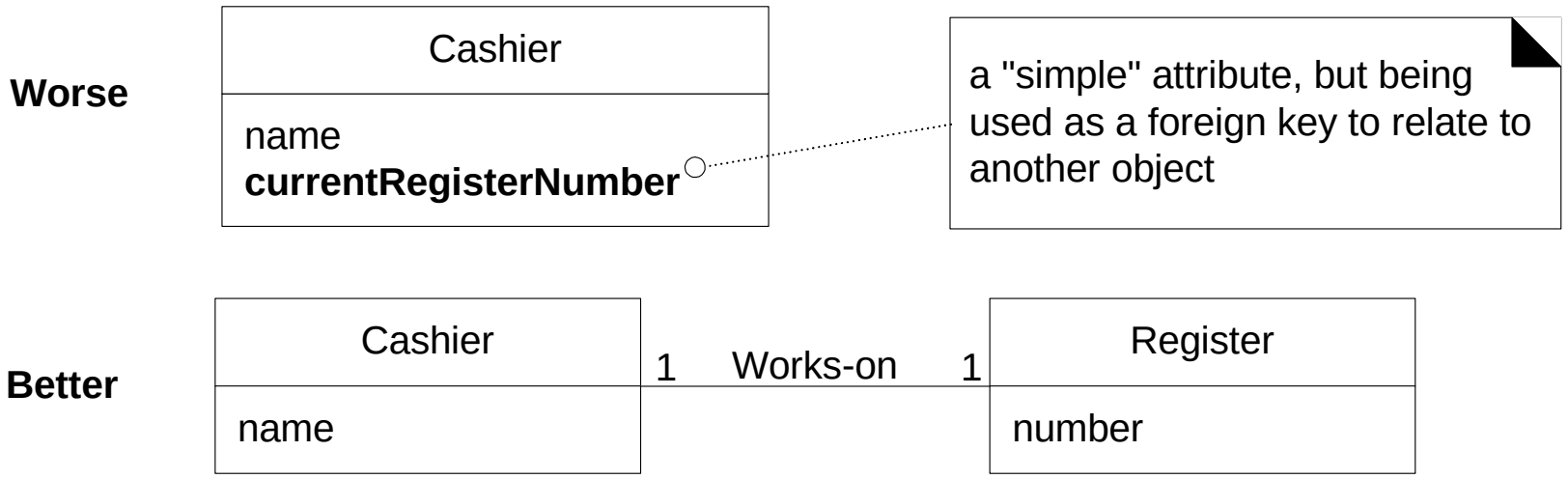
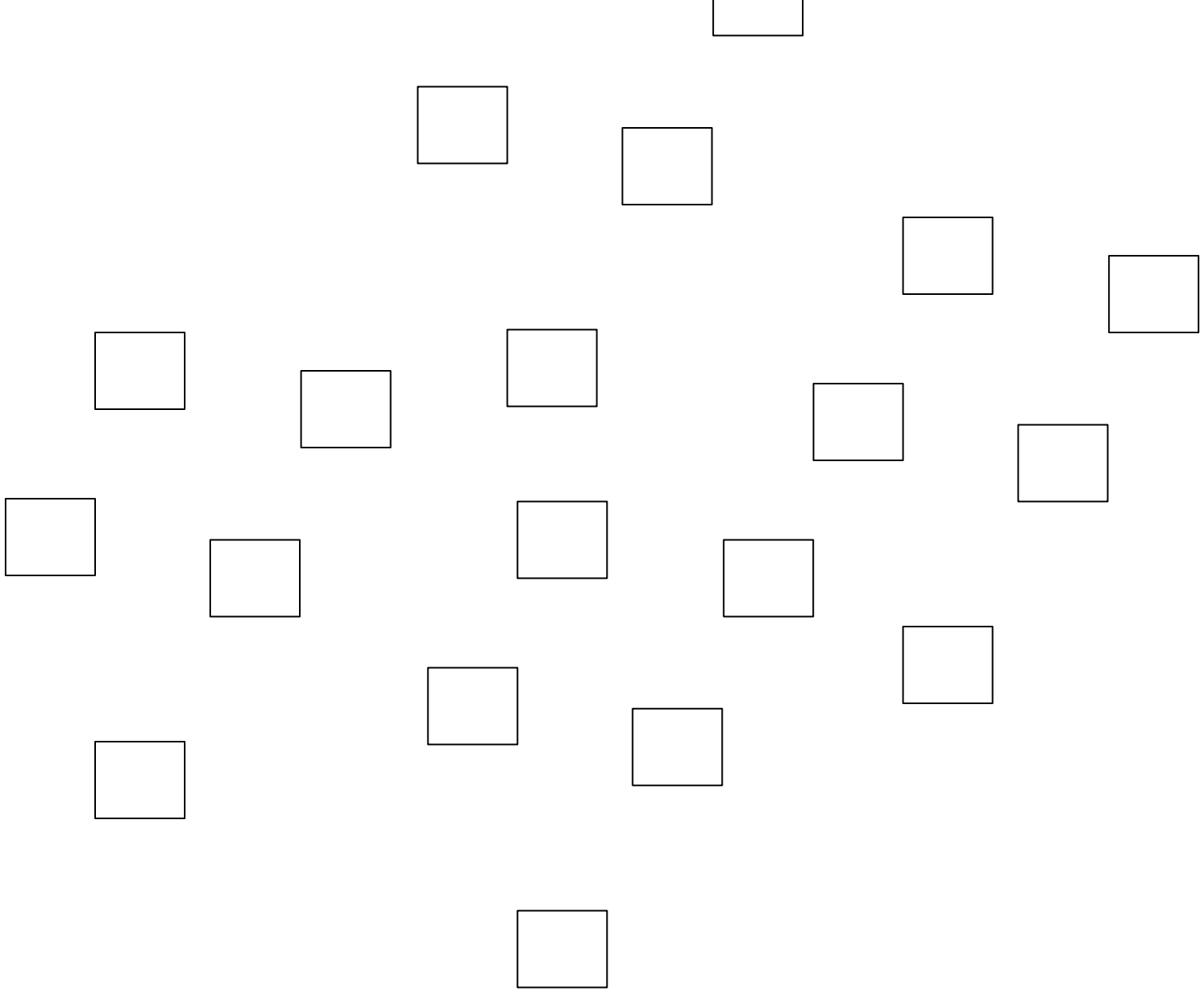


Figure 9.25 from [1, 3]



Make the relationships visual.
Avoid relating concepts with foreign keys.

Complex concepts not as attributes

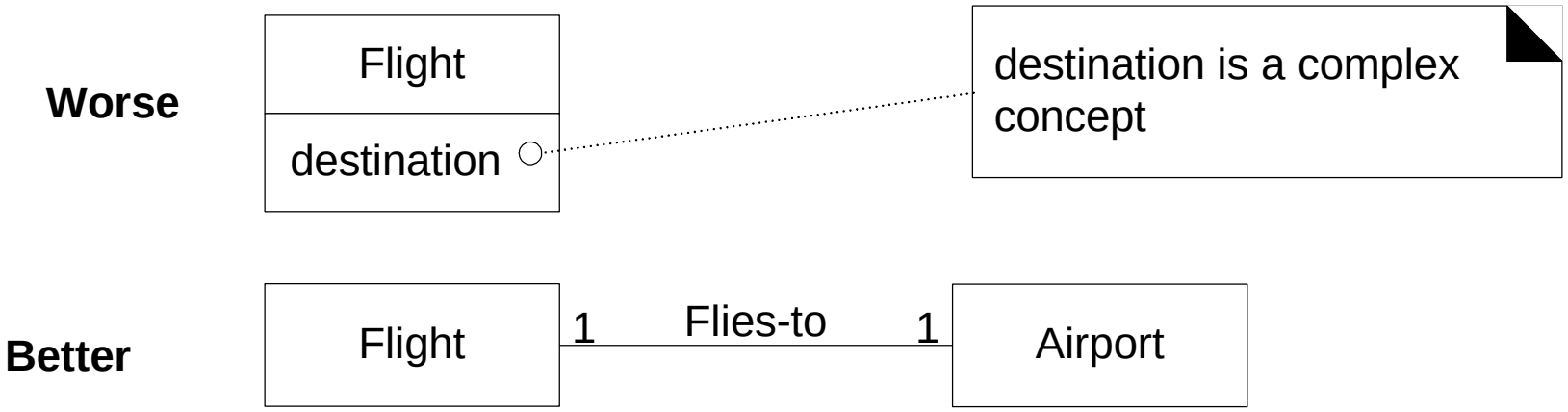


Figure 9.23 from [1, 3]

Actividad

- Quiz E09-3 Repaso de modelos de dominio

State diagrams

Outline

- State machine diagram notation
- Exercise

What is a State machine diagram?

- A state machine diagram shows the lifecycle of things [1] in time as they respond to events [5]

What is a State machine diagram?

- A state machine diagram shows the lifecycle of things [1] in time as they respond to events [5]

- What kind of “things”?

When create state machine diagrams? [5]

“Things”	**In which model?**
real-world concepts [5]	• Domain model
transactions [1]	• Domain model
use cases [1]	• Use case model
people [1]	• Domain model
software objects [5]	• Design model
classes [5]	• Design model
physical object	• Domain model

For business information systems[1]

“Things”	**In which model?**
process controllers [1]	• Domain/Design model
device controllers [1]	• Domain/Design model
protocol handlers [1]	• Domain/Design model
telecommunication objects [1]	• Design model

State machine diagram for a telephone

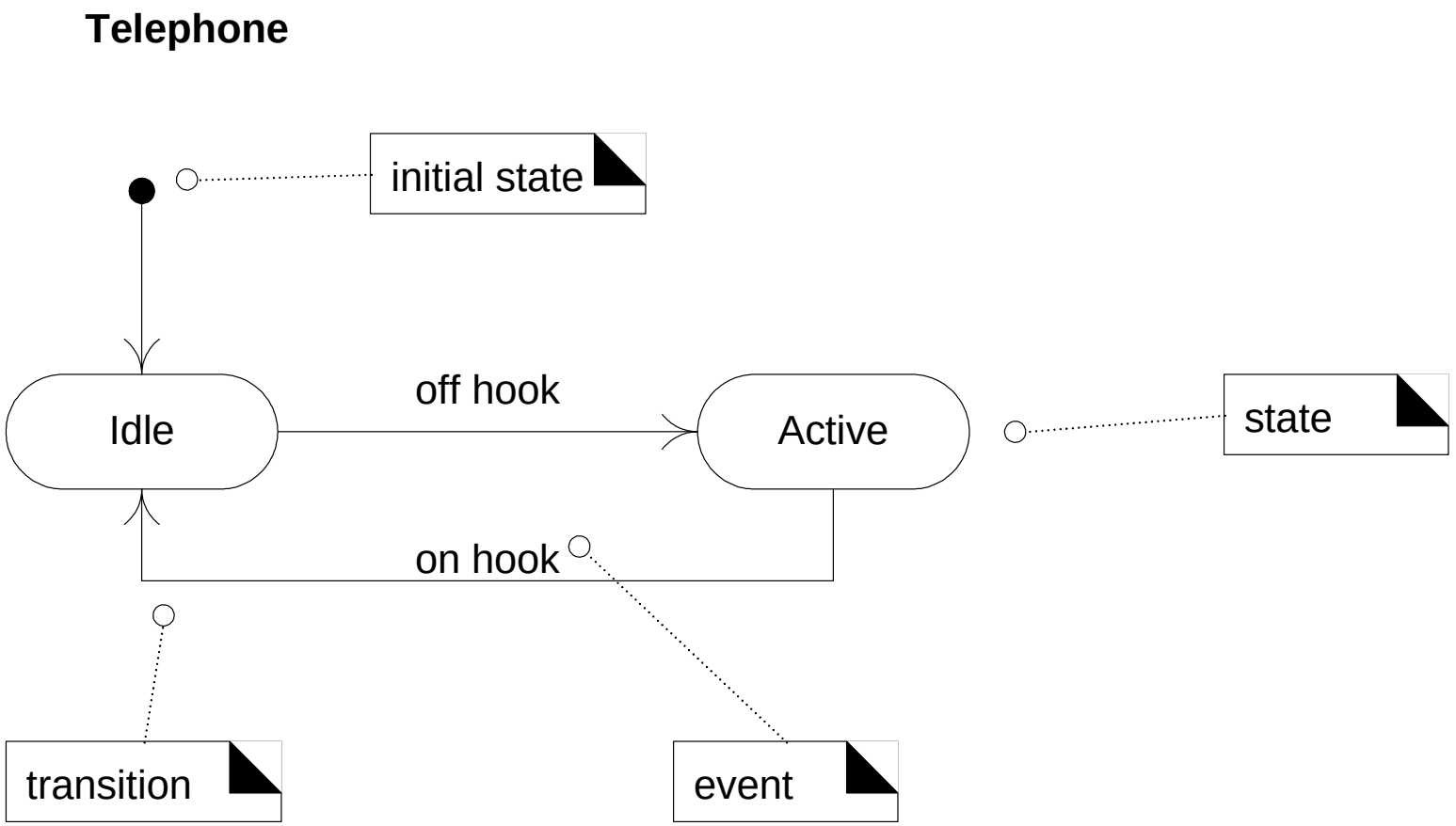
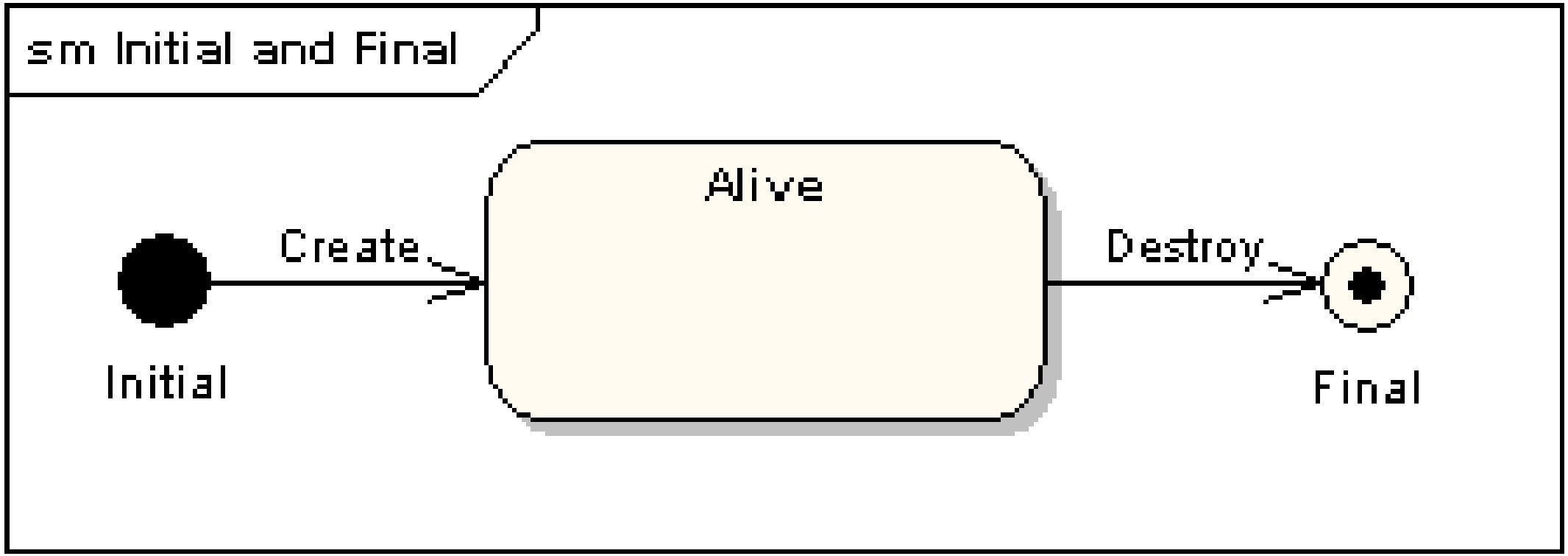


Figure 29.1 from [3]



Start and Final states

Transition action and guard notation

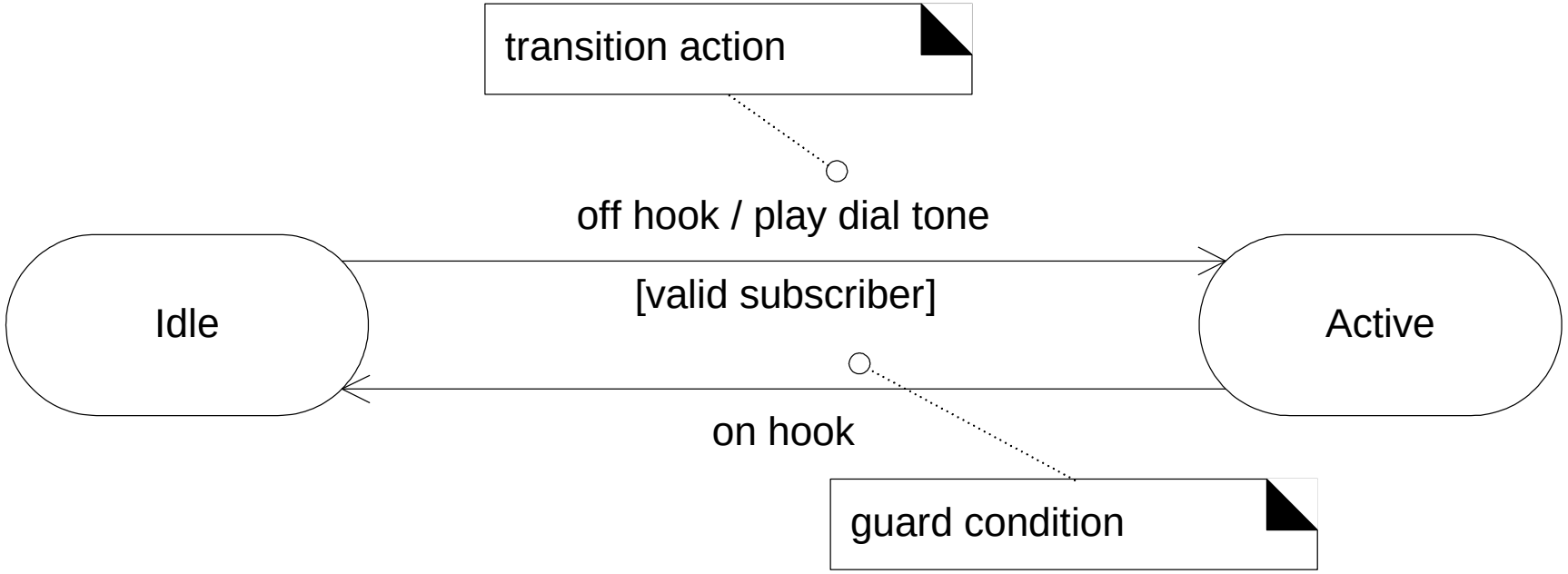


Figure 29.1 from [3]

Nested states

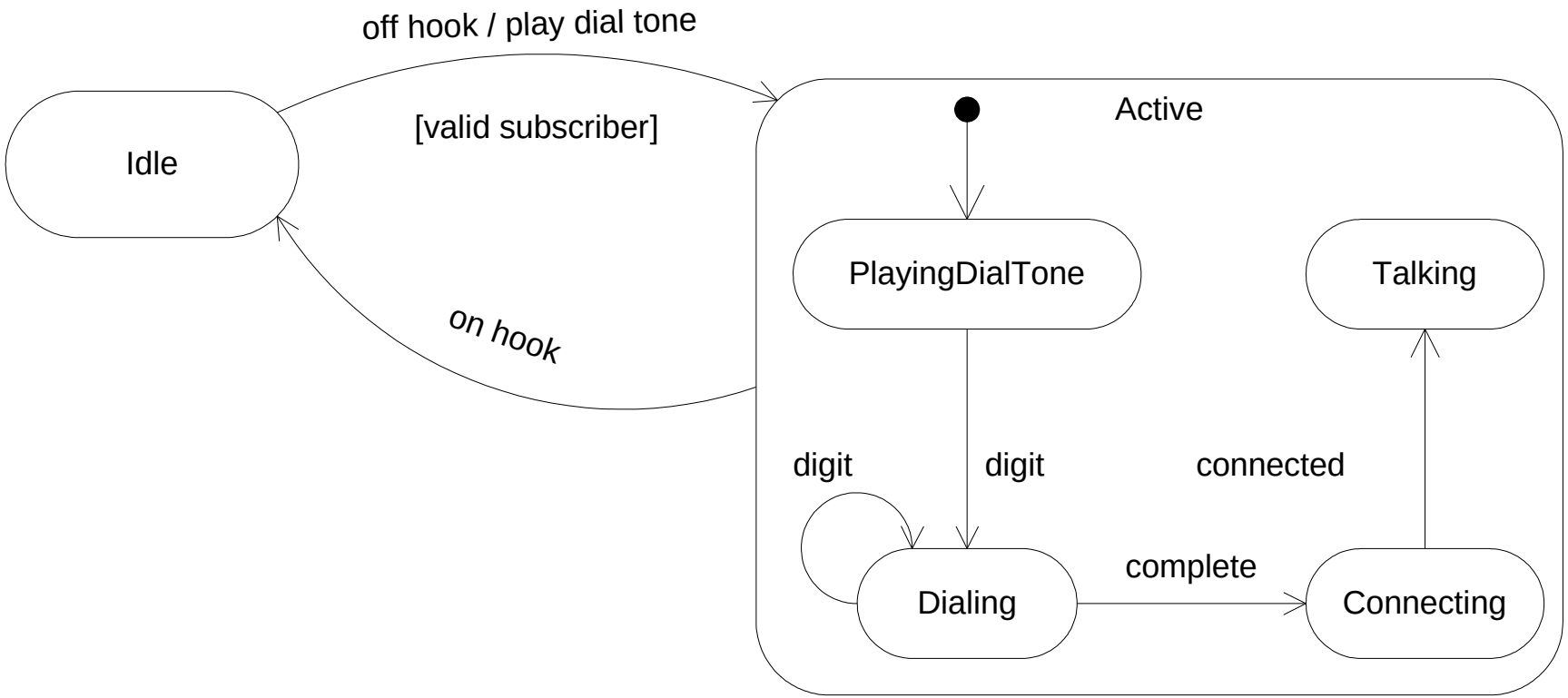


Figure 29.3 from [3]

Applying a state machine to Web page navigation modeling

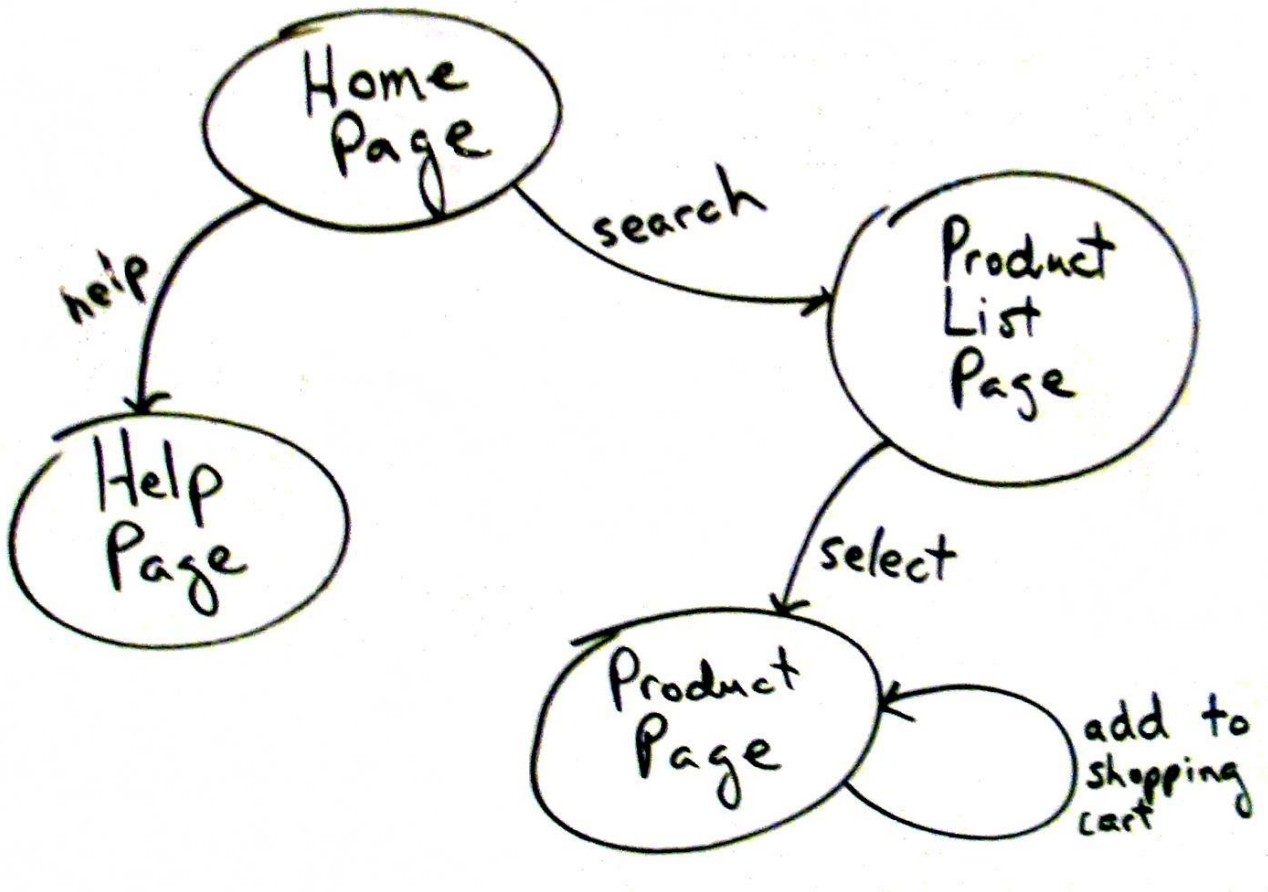


Figure 29.4 from [3]

Use case state diagrams

Show legal sequence
of use case operations

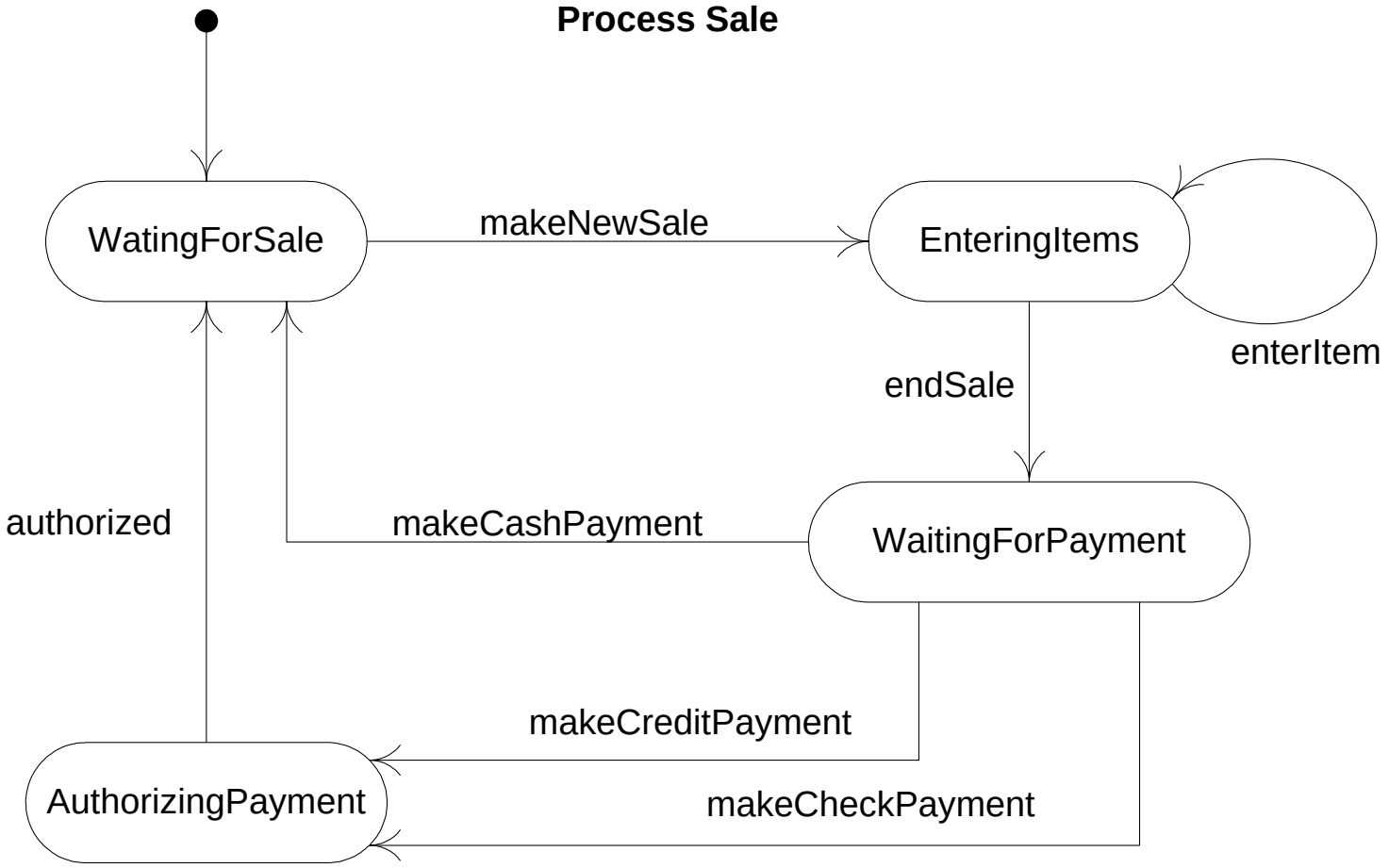
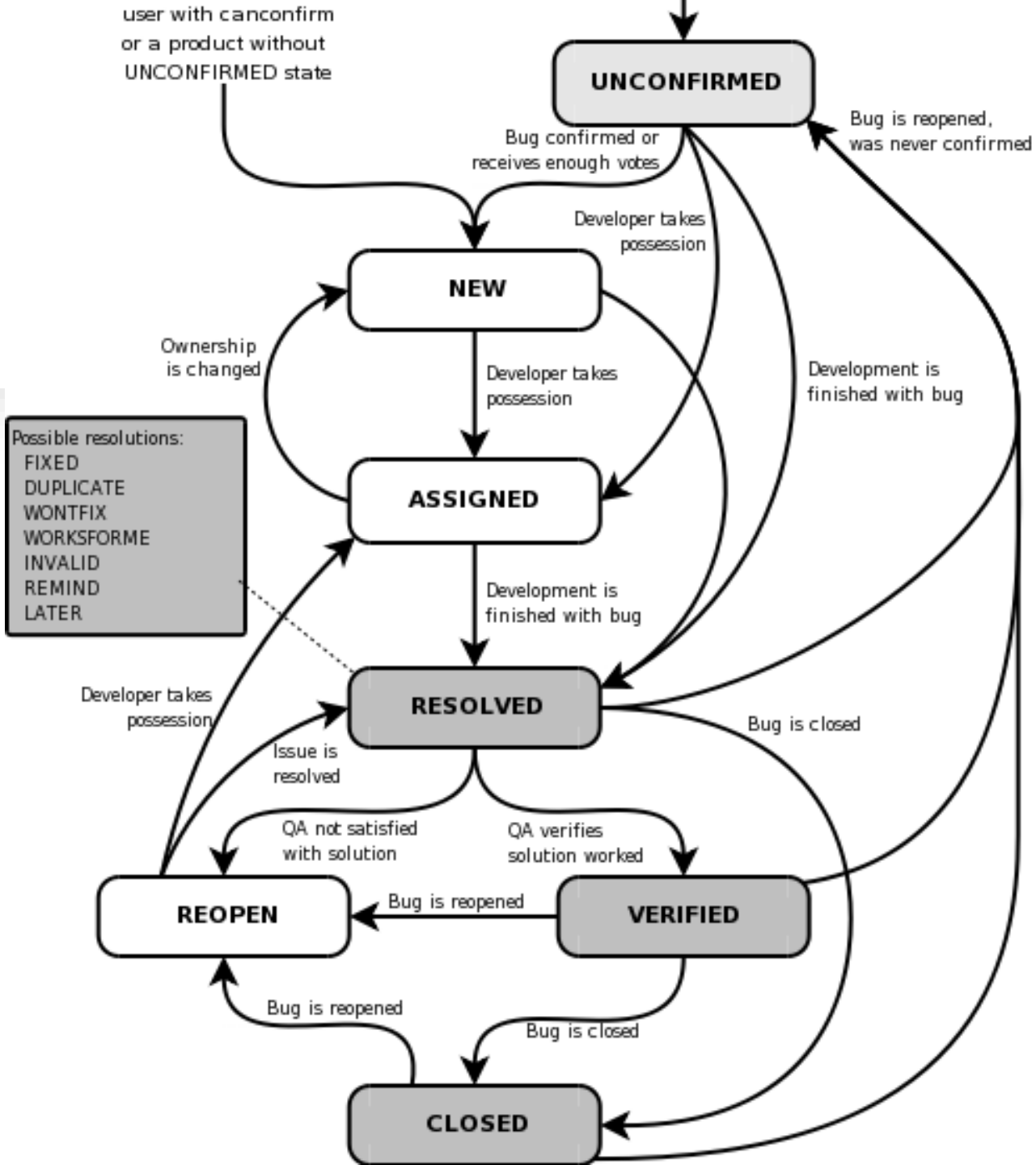


Figure 29.5 from [3]

Example: Lifecycle of a Bugzilla Bug

- https://www.bugzilla.org/docs/2.20/html/lifecycle.html

Bugzilla bug states



Activity: Create a State Machine diagram for one of the following systems

- Bug tracker
  - Bug life-cycle (Domain Model)
- Help desk ticket system
  - Ticket life-cycle (Domain Model)
- Delivery notification system
  - Package life-cycle (Domain Model)
- Scrum Task Manager
  - Task life-cycle (Domain Model)
  - User story life-cycle (Domain Model)

Actividad

- Quiz E09-4 Repaso de diagramas de estado

Clase 10

Outline

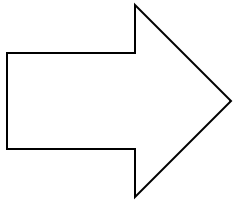
- UML for analysis
  - Use case diagrams
  - System Sequence Diagrams
  - Domain models
  - State diagrams
  - Activity diagrams

Activity diagrams

Activity diagrams

- They “show a sequence of activities [1]”

Activity diagram notation



Activity diagram notation

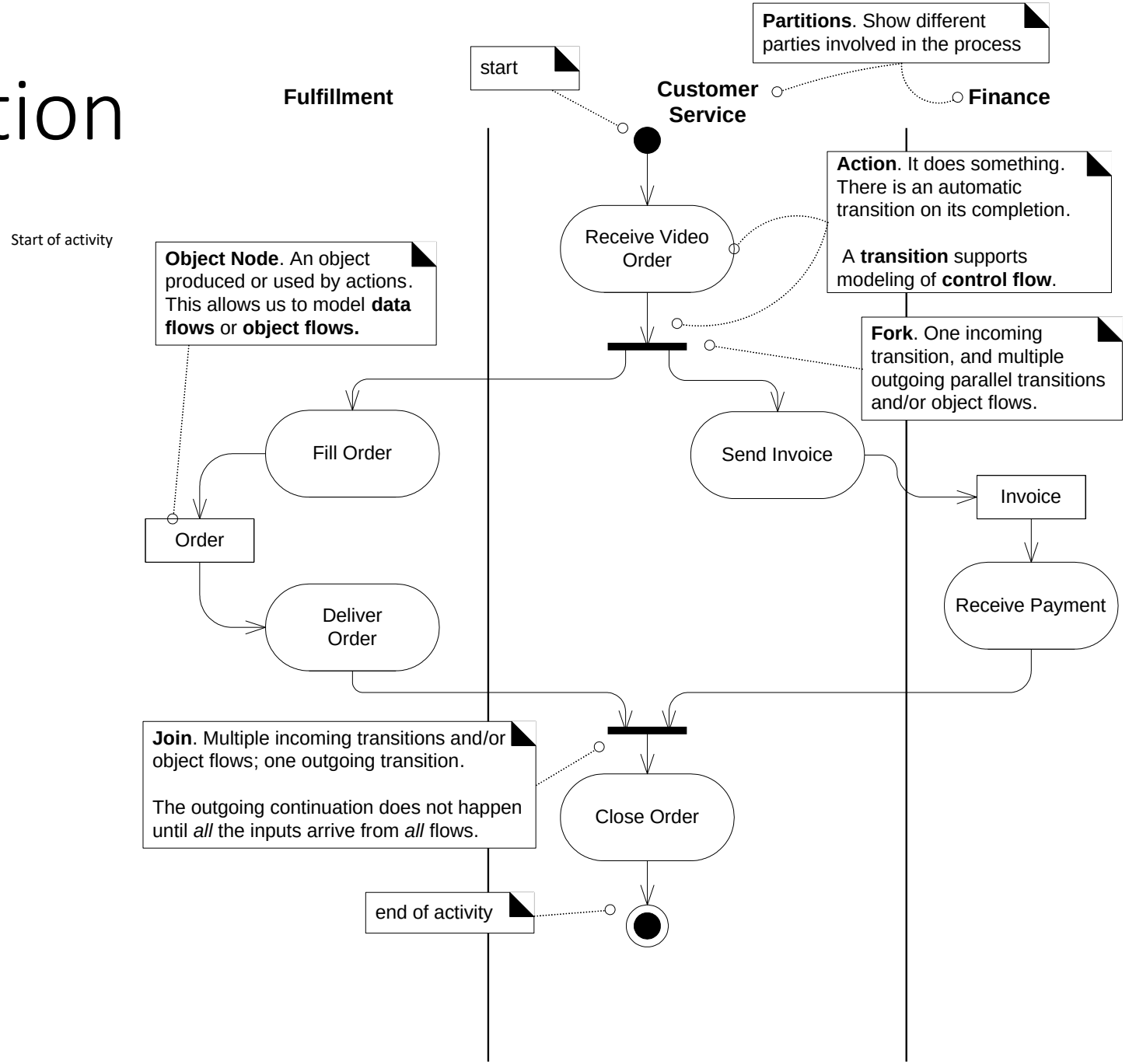


Figure 28.1 from [3]

Activity diagram notation

"Once an action is finished, there is an automatic outgoing transition. [1]"

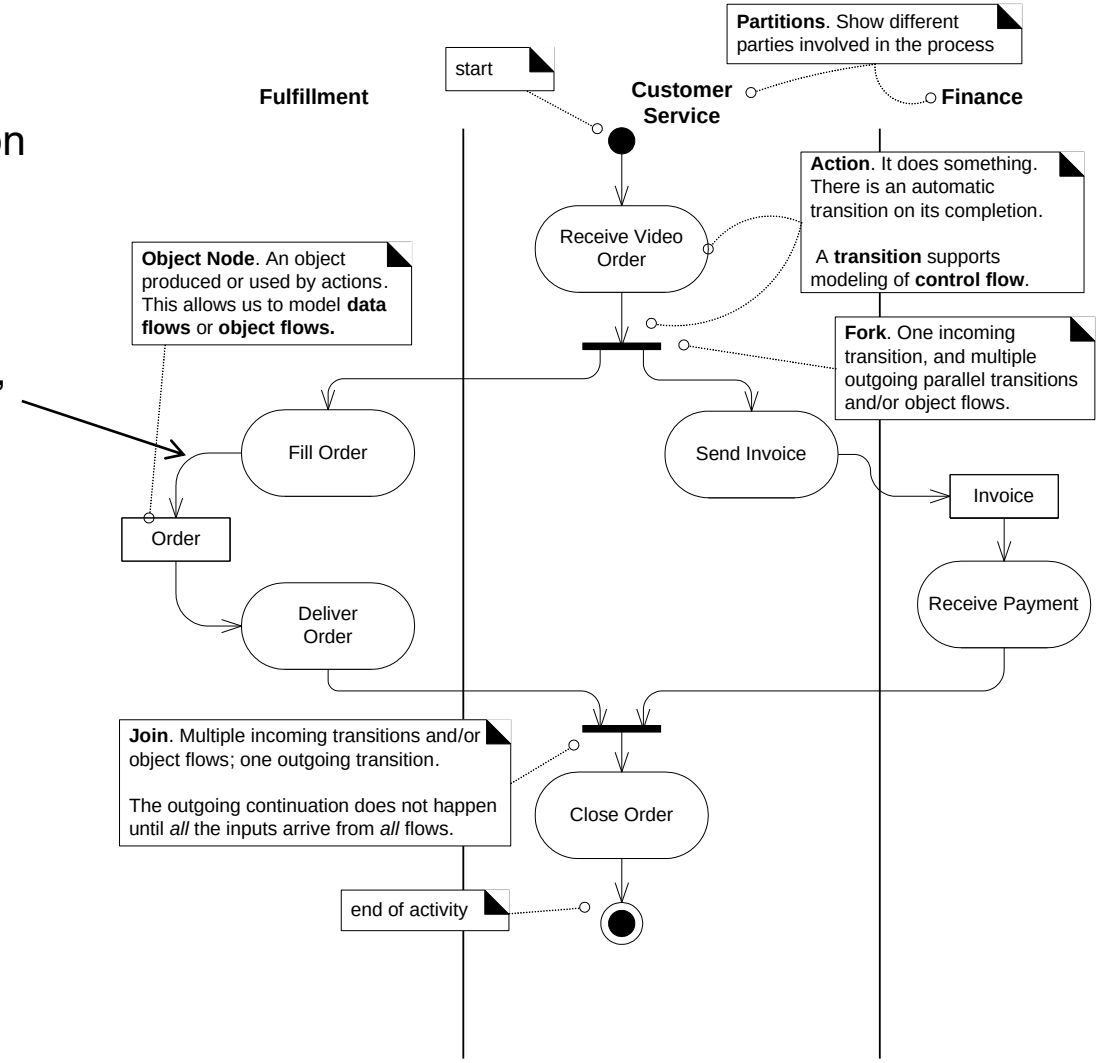


Figure 28.1 from [3]

Activity diagram notation

"The diagram can show both control flow and data flow. [1]"

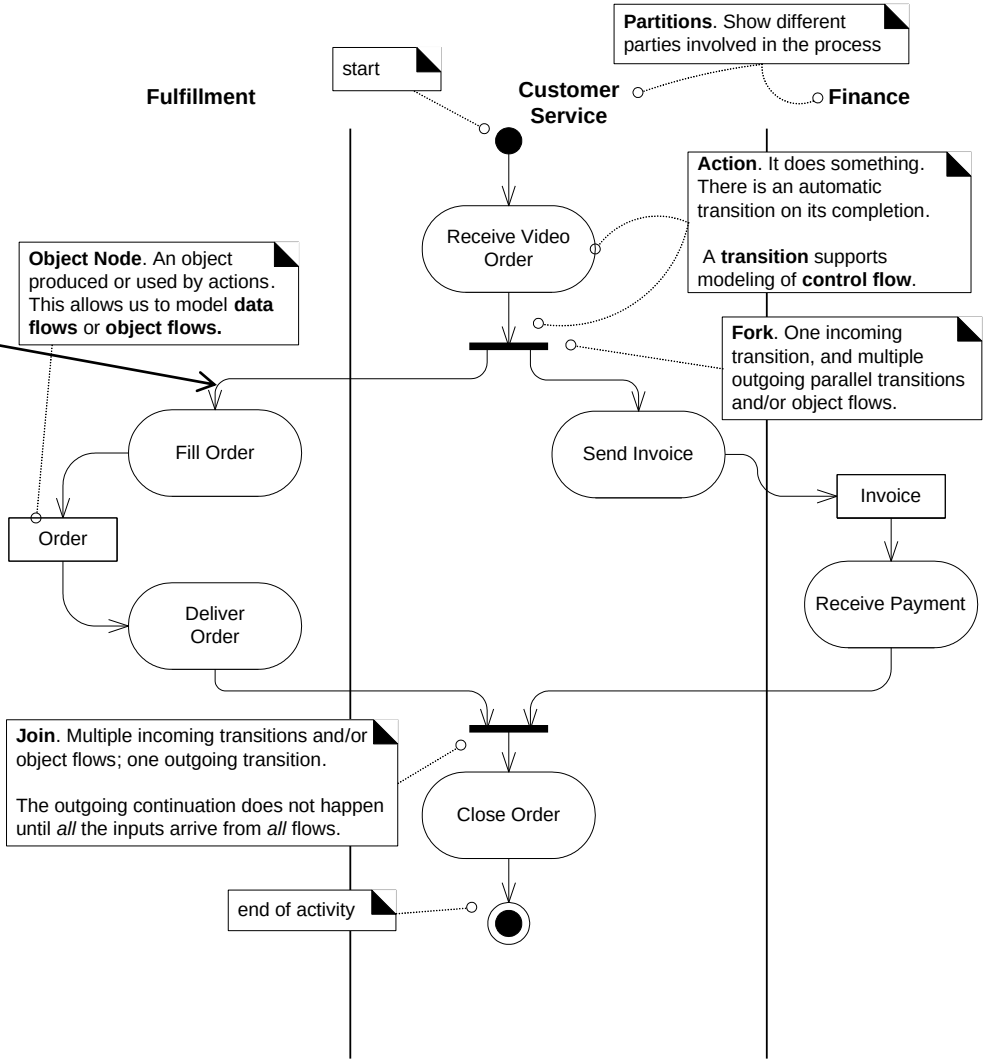


Figure 28.1 from [3]

Partitions

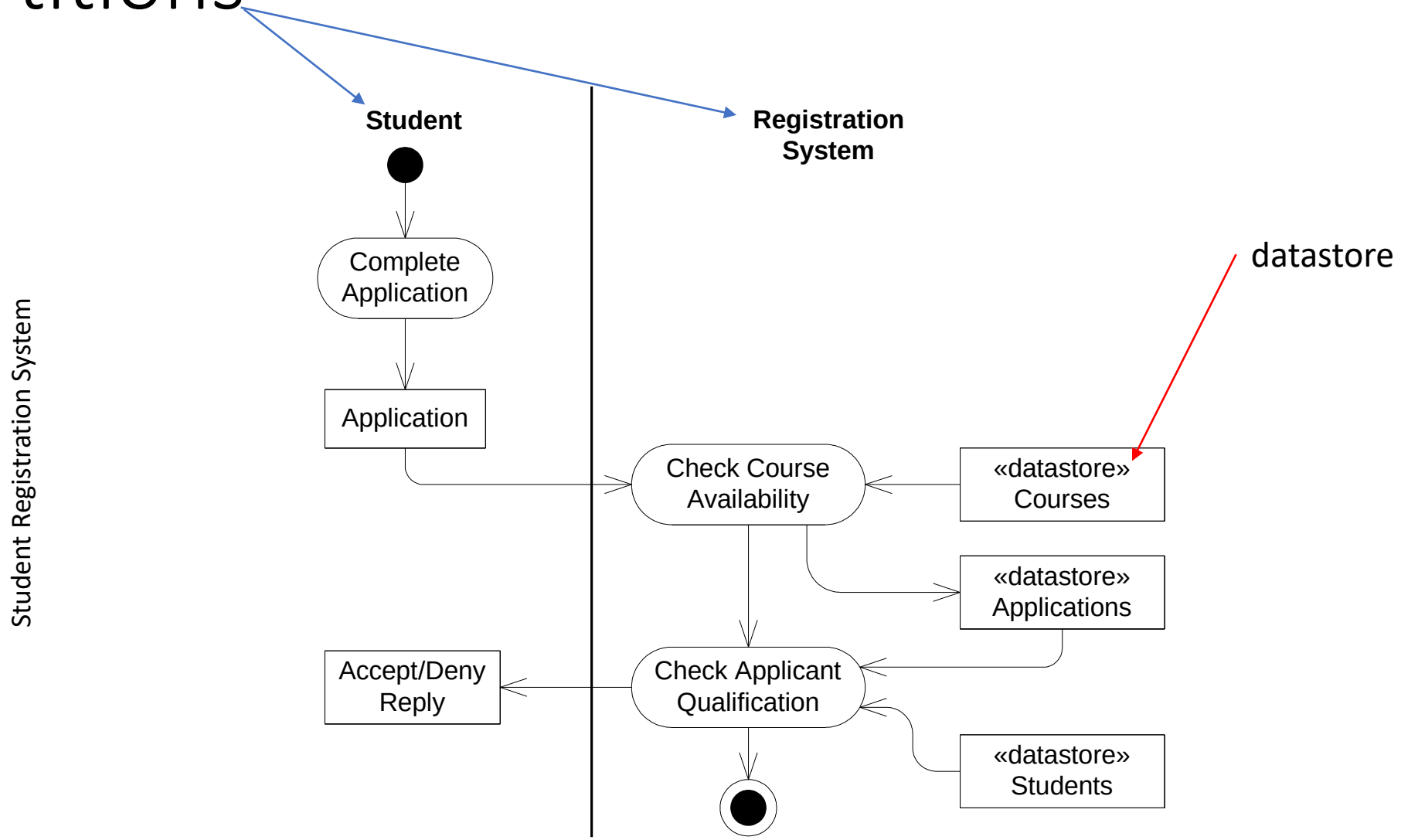


Figure 28.3 from [3]

Process Sales of the POS system

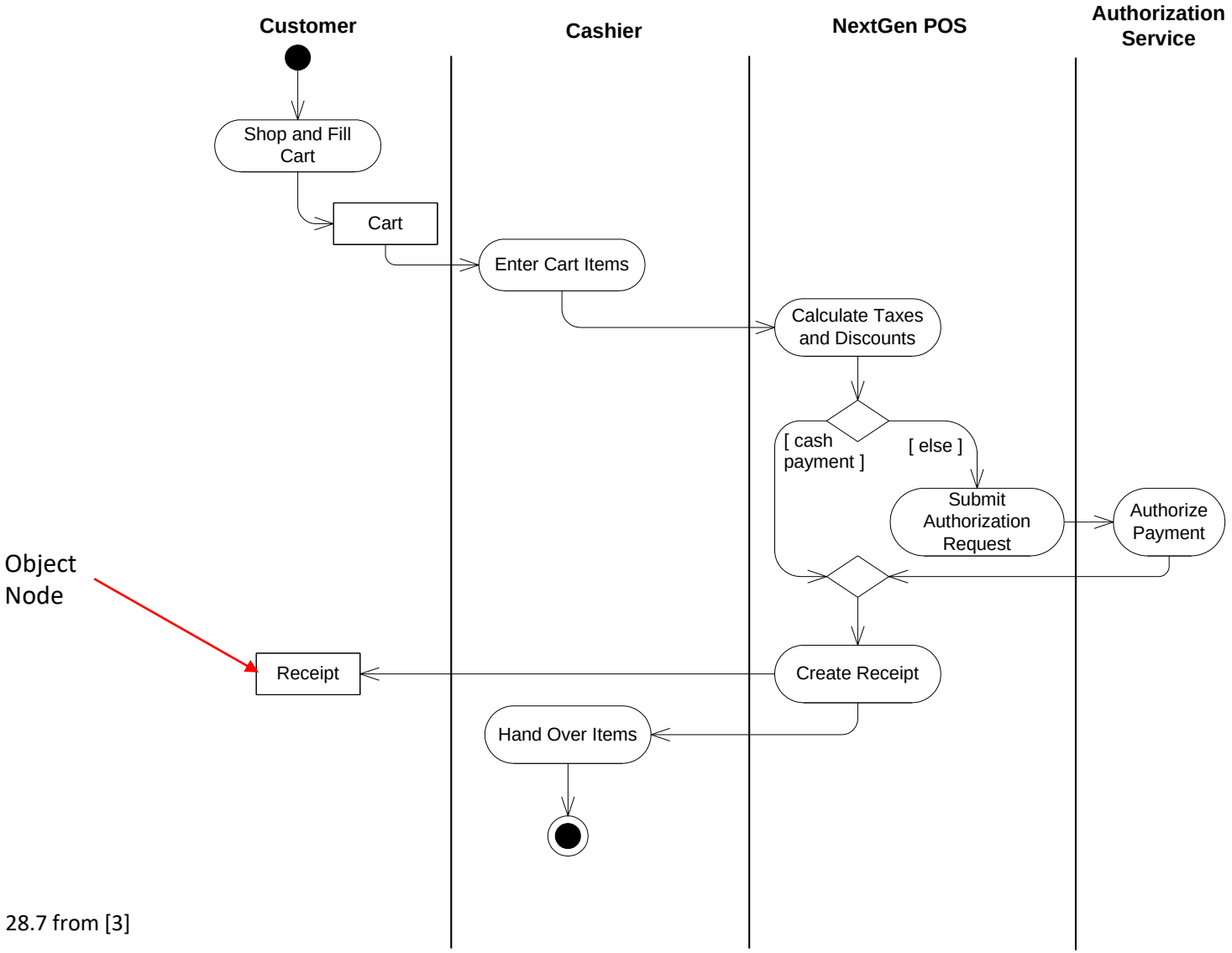
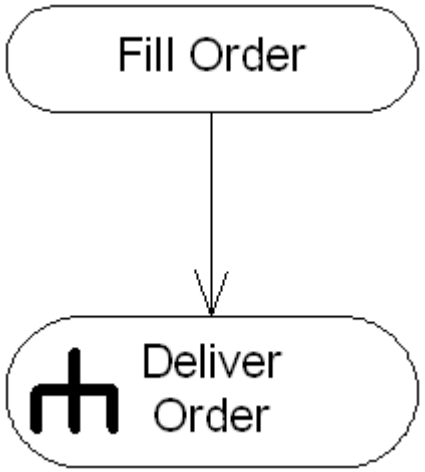


Figure 28.7 from [3]

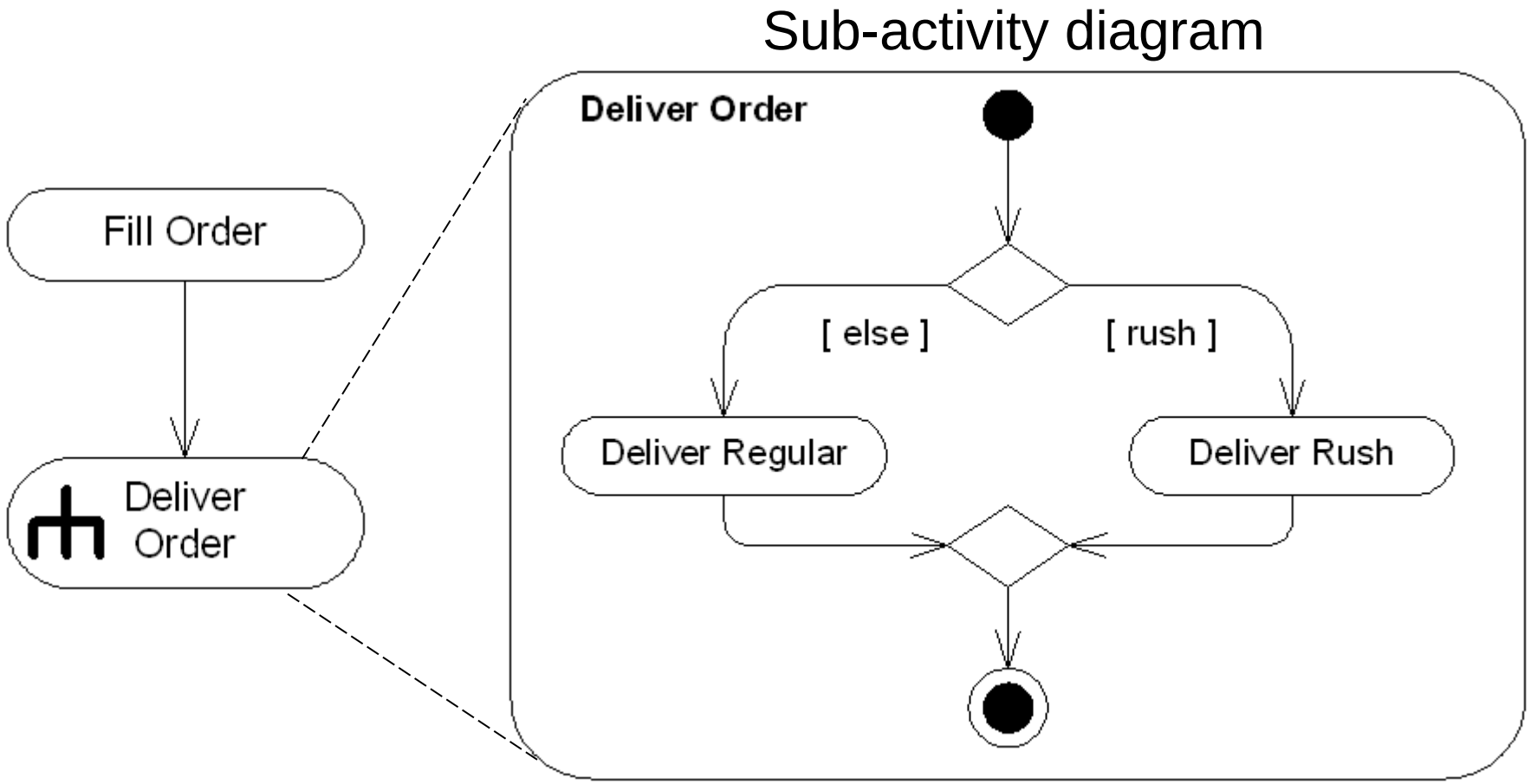
More notation

- The rake symbol (⫛) used to convey hierarchy



More notation

- The rake symbol used to convey hierarchy



Expansion of action with 

the “rake” symbol (which represents a hierarchy) indicates this activity is expanded in a sub-activity diagram

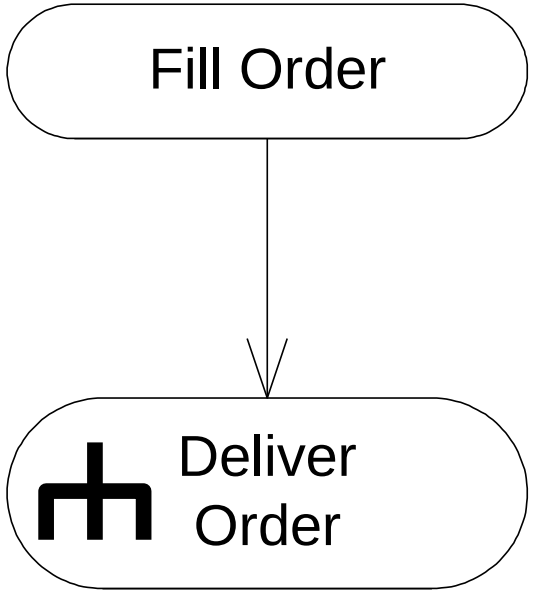
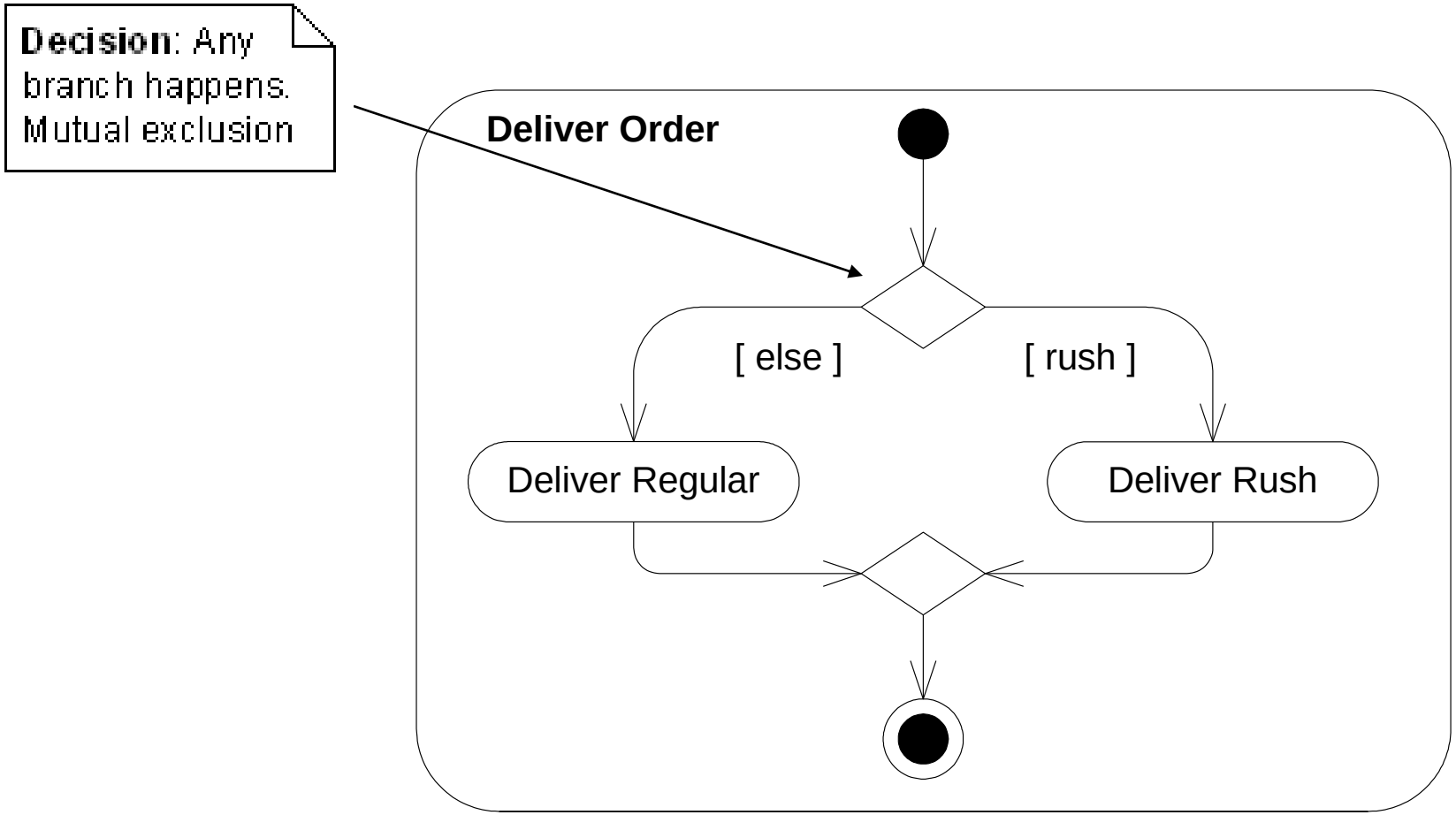


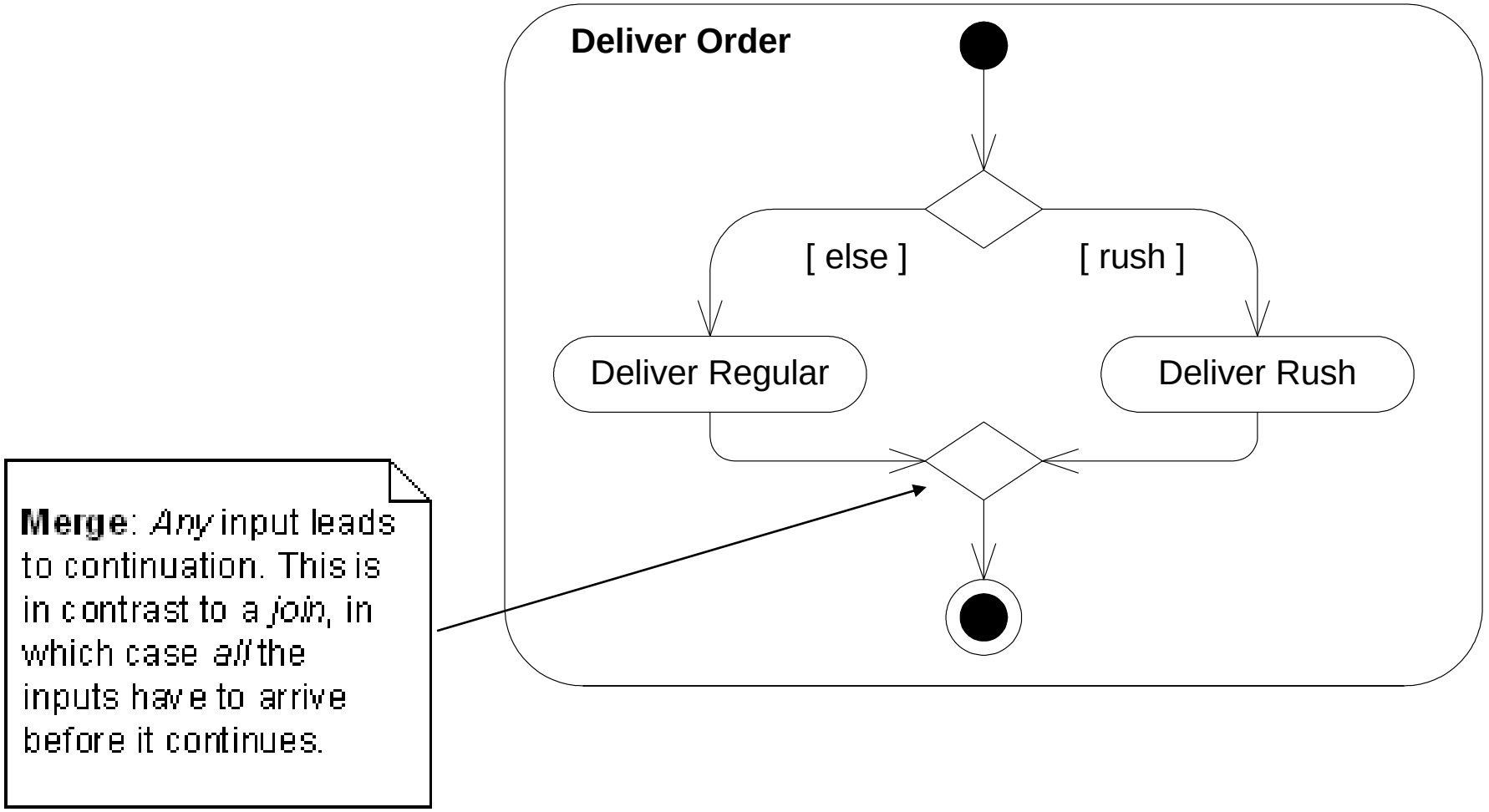
Figure 28.4 from [3]

Decision and Merge symbols



Adapted from Figure 28.5 from [3]

Decision and Merge symbols

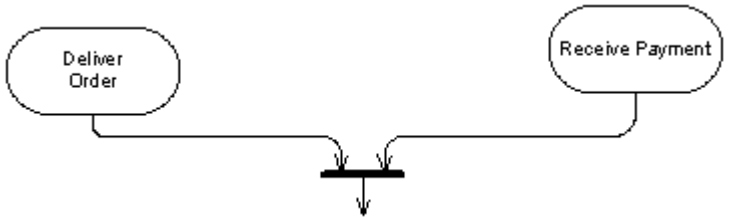


Adapted from Figure 28.5 from [3]

Comparison

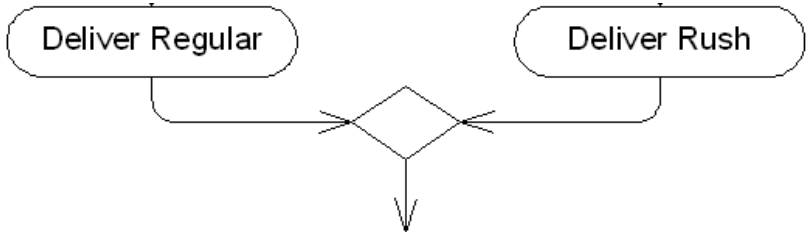
Join

- Can’t continue unless all inputs have arrived



Merge

- “Any input leads to continuation [1]”



Send signal notation

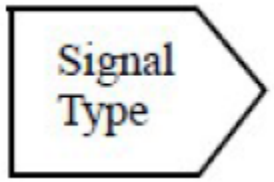
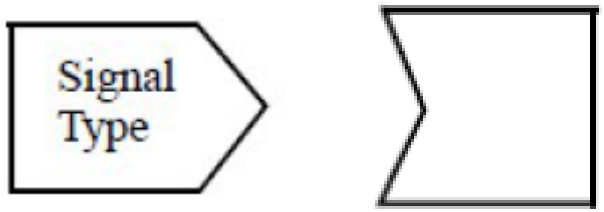


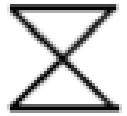
Figure 16.40 from [6]

Accept event action notations

Also: Accept signal



Accept event action



Accept time event action

An accept event action
with a time trigger

Figure 16.40 from [6]

Receive video order

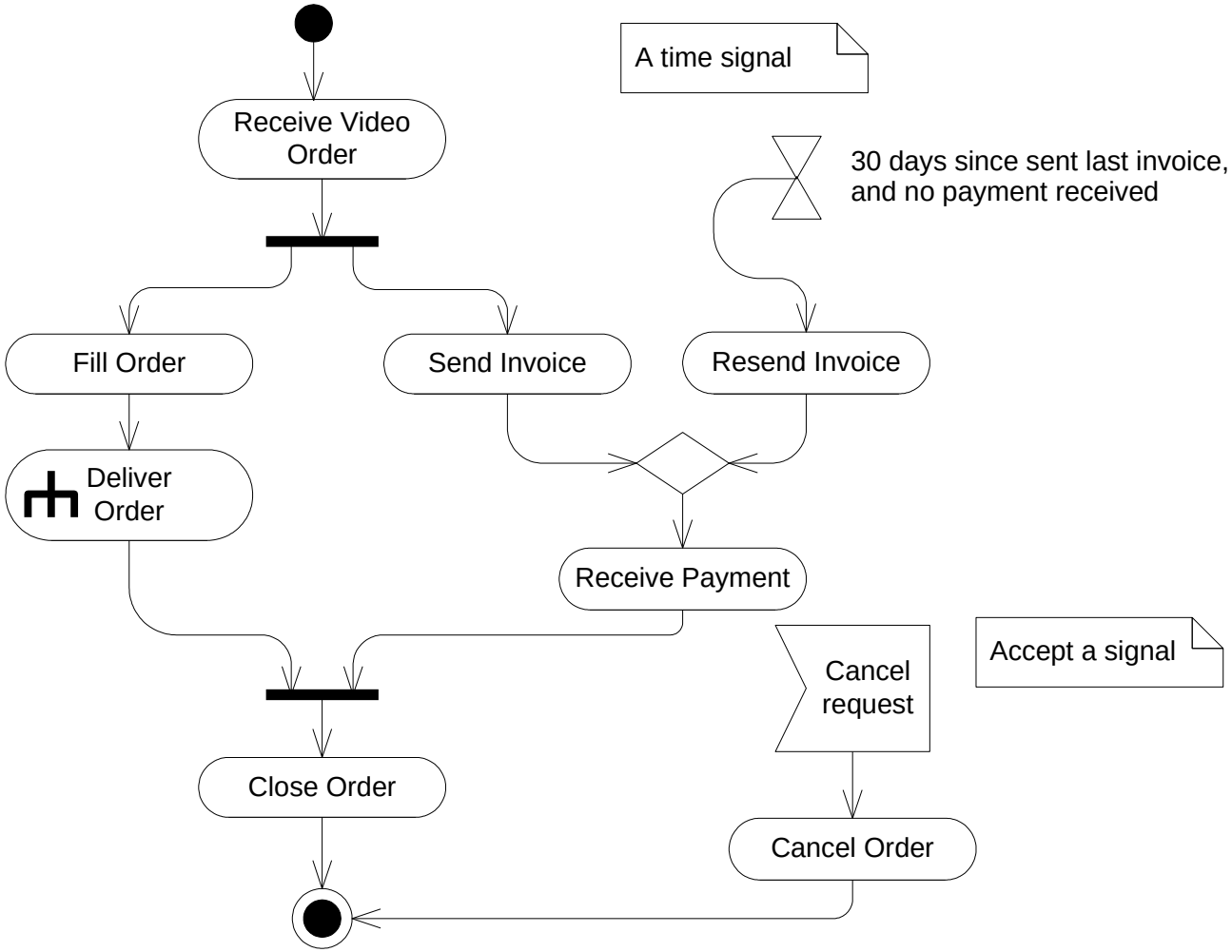


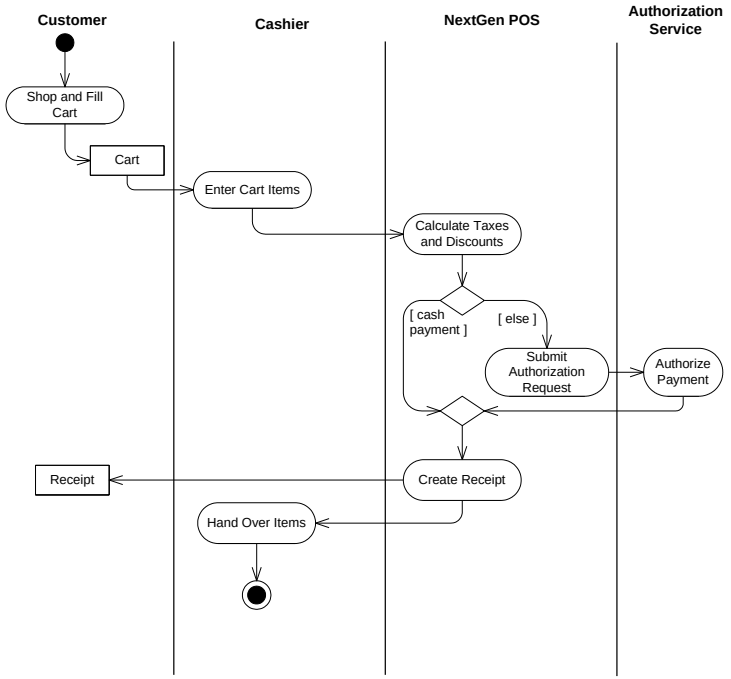
Figure 28.6 from [3]

Uses of activity diagrams

- Modeling:
  - Use cases [5]
    - SSDs
  - Business processes
  - Data flows
  - Concurrent programming and parallel algorithms
  - Visualizing the steps of an algorithm [5]

What is a business process?

- “A **business process** or **business method** is a collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers. [2]”



Concurrency

- Use of partitions for different processes
- Forks for starting processes

Activity (Optional)

- Create an activity diagram that illustrates the process when a campaign is created to the moment it is shared to customers

Actividad

- Quiz E10-1 Repaso de diagrama de actividades

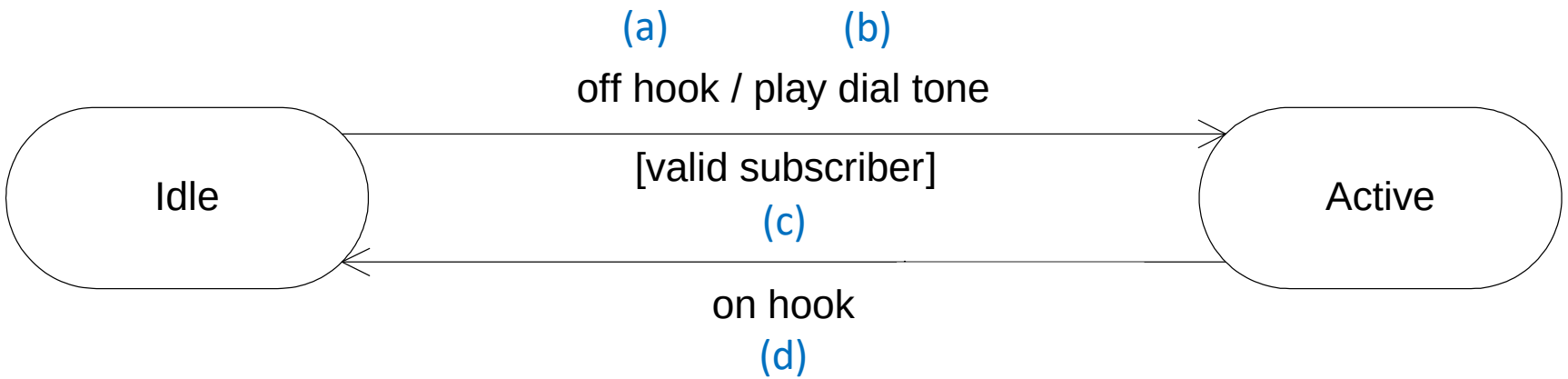
References

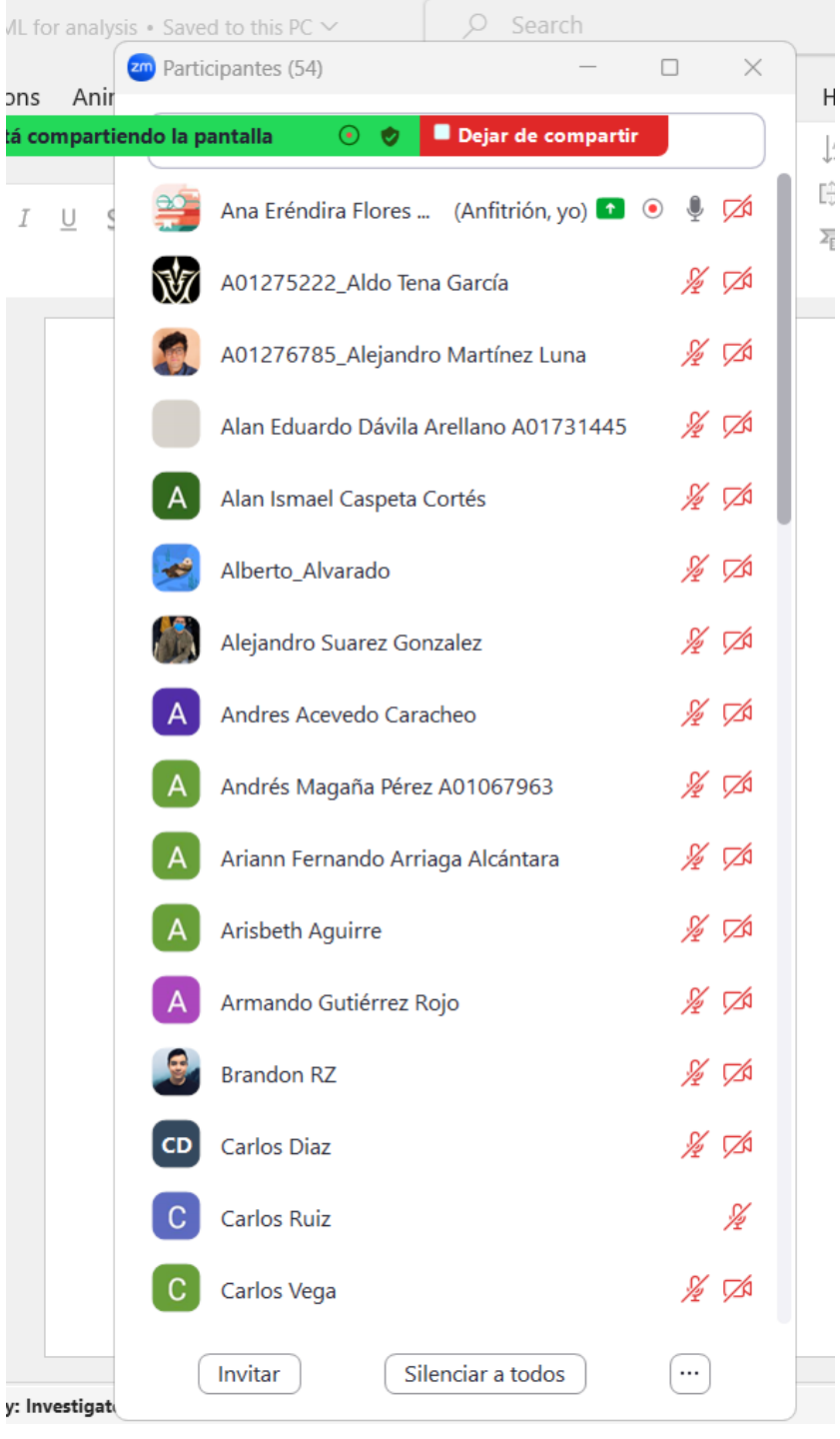
- 1. Larman, Craig (2005). *Applying UML and Patterns*. Prentice Hall PTR, 3rd ed.
- 2. Larman, C. (2004). UML y patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado, Pearson Educación, 2ª ed., España.
- 3. Larman, C (?). Educator Resources, URL: https://www.craiglarman.com/wiki/index.php?title=Educator_Resources#Applying_UML_and_Patterns, as of 9/4/20.
- 4. Valtech (1999). Chapter 6: Creating a Conceptual Model
- 5. Fowler Martin (1996). Analysis Patterns: Reusable Object Models, 1st Edition, Addison-Wesley.
- 6. Hay David C. (1996). Data Model Patterns: Conventions of Thought, Dorset House eBooks.

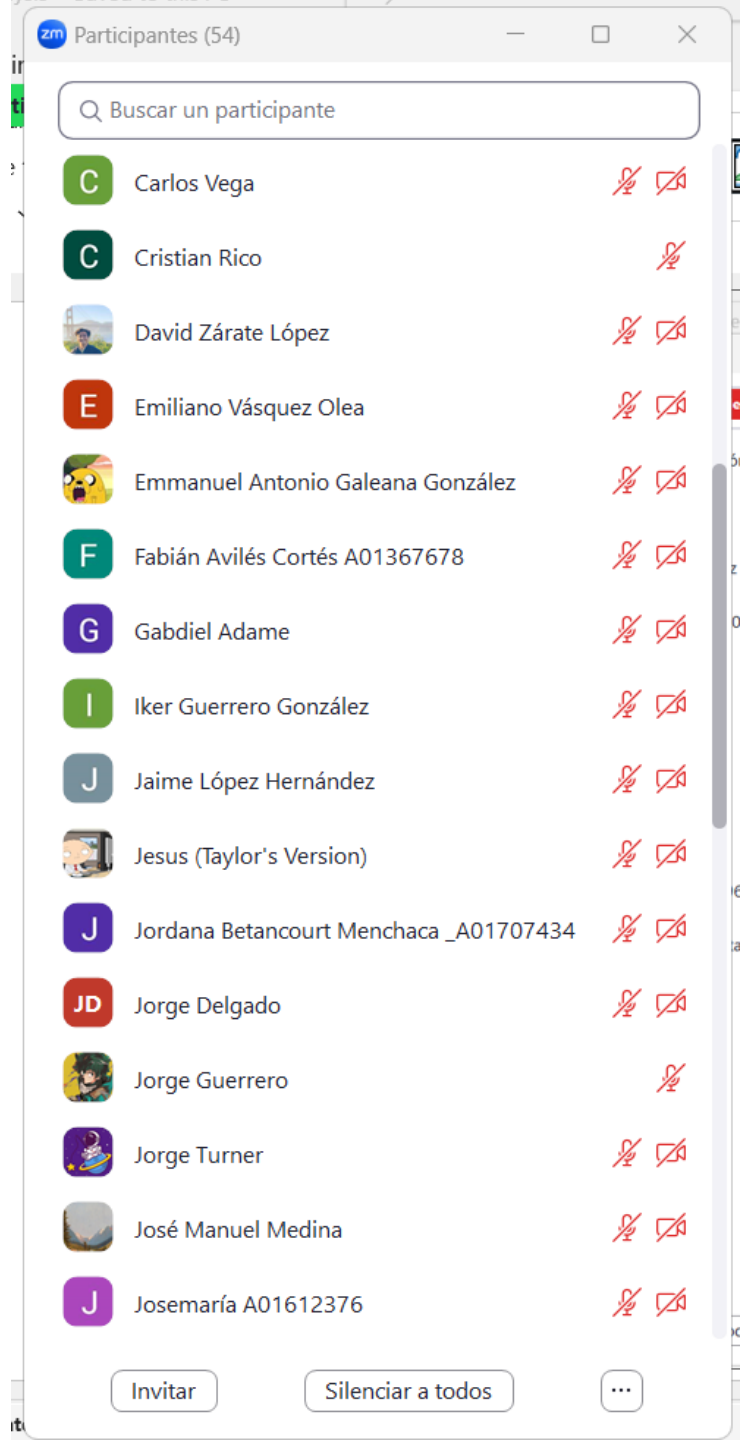
Appendix

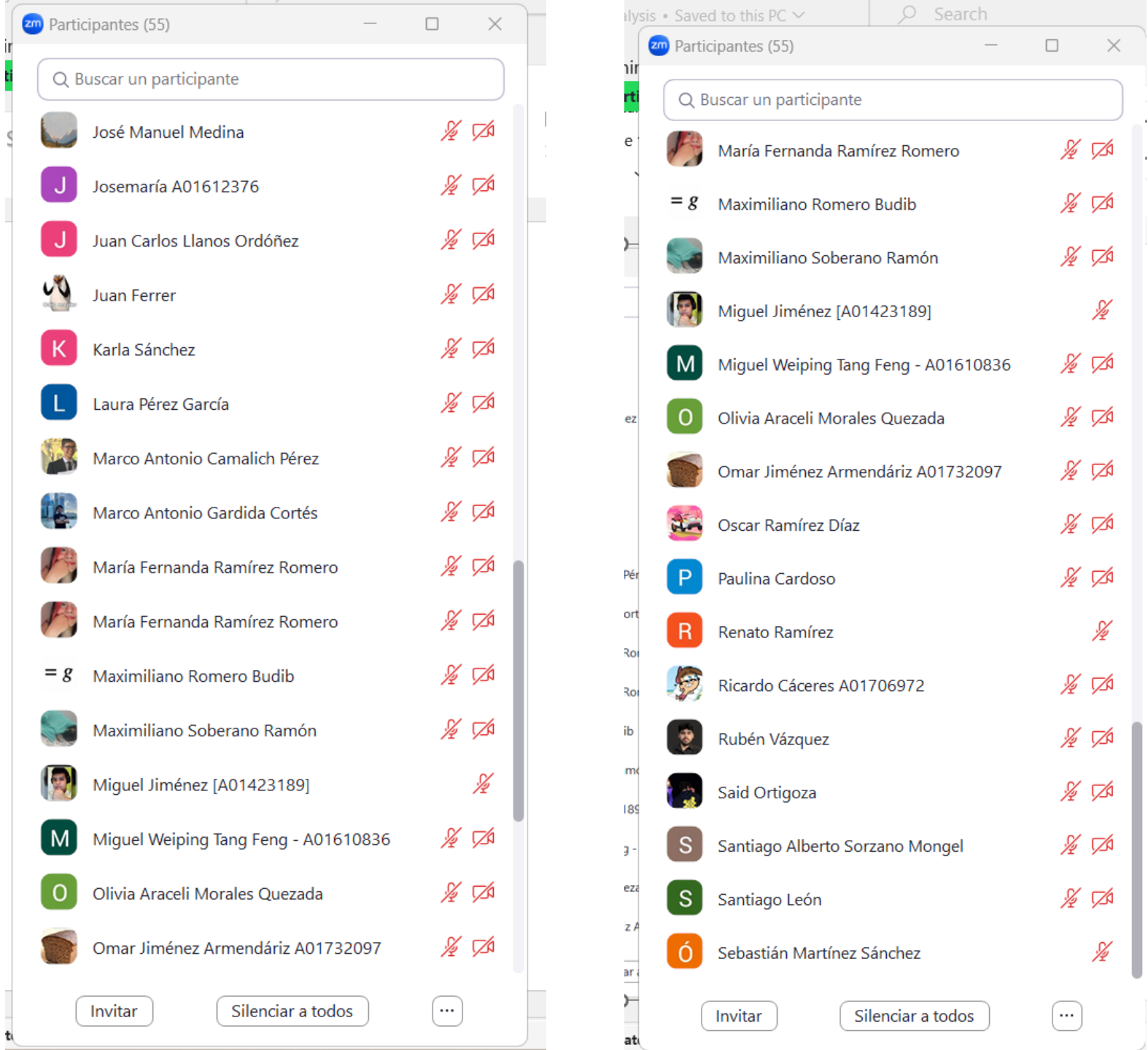
(a)

(b)


(a)
(b)
off hook / play dial tone
Idle
[valid subscriber]
(c)
Active
on hook
(d)


Participantes (54)
Dejar de compartir
Ana Eréndira Flores ... (Anfitrión, yo)
A01275222_Aldo Tena García
A01276785_Alejandro Martínez Luna
Alan Eduardo Dávila Arellano A01731445
Alan Ismael Caspeta Cortés
Alberto_Alvarado
Alejandro Suarez Gonzalez
Andres Acevedo Caracheo
Andrés Magaña Pérez A01067963
Ariann Fernando Arriaga Alcántara
Arisbeth Aguirre
Armando Gutiérrez Rojo
Brandon RZ
Carlos Diaz
Carlos Ruiz
Carlos Vega
Invitar
Silenciar a todos


Participantes (54)
Buscar un participante
Carlos Vega
Cristian Rico
David Zárate López
Emiliano Vásquez Olea
Emmanuel Antonio Galeana González
Fabián Avilés Cortés A01367678
Gabdiel Adame
Iker Guerrero González
Jaime López Hernández
Jesus (Taylor's Version)
Jordana Betancourt Menchaca _A01707434
Jorge Delgado
Jorge Guerrero
Jorge Turner
José Manuel Medina
Josemaría A01612376
Invitar
Silenciar a todos


Participantes (55)
Buscar un participante
José Manuel Medina
Josemaría A01612376
Juan Carlos Llanos Ordóñez
Juan Ferrer
Karla Sánchez
Laura Pérez García
Marco Antonio Camalich Pérez
Marco Antonio Gardida Cortés
María Fernanda Ramírez Romero
María Fernanda Ramírez Romero
Maximiliano Romero Budib
Maximiliano Soberano Ramón
Miguel Jiménez [A01423189]
Miguel Weiping Tang Feng - A01610836
Olivia Araceli Morales Quezada
Omar Jiménez Armendáriz A01732097
Invitar
Silenciar a todos
Participantes (55)
Buscar un participante
María Fernanda Ramírez Romero
Maximiliano Romero Budib
Maximiliano Soberano Ramón
Miguel Jiménez [A01423189]
Miguel Weiping Tang Feng - A01610836
Olivia Araceli Morales Quezada
Omar Jiménez Armendáriz A01732097
Oscar Ramírez Díaz
Paulina Cardoso
Renato Ramírez
Ricardo Cáceres A01706972
Rubén Vázquez
Said Ortigoza
Santiago Alberto Sorzano Mongel
Santiago León
Sebastián Martínez Sánchez
Invitar
Silenciar a todos