The 10 Don'ts (and Dos) of Modeling Use Cases

Meilir Page-Jones

Level-Setting Preamble

- **Examples of use cases**
- **What is a use case?**
- **Use case model deliverables**
- **Simplified example of a use case narrative**
- **Why use cases?**
- **Who produces use cases and from what?**

Examples of use cases

- Open new customer account
- Download e-mail
- Add new customer
- Admit patient to hospital
- Place customer order
- Change customer profile
- Respond to billing inquiry
- Accept auction item for listing
- Authorize merchandise return

What is a use case?

A defined sequence of interactions between a system and an actor playing a specific role, conducted towards a goal of value to the actor and / or business

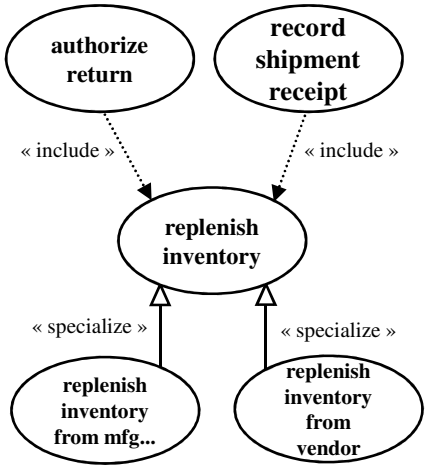
- **Defined**
- **Actor**
- **Role**
- **System**
- **Goal (or purpose)**

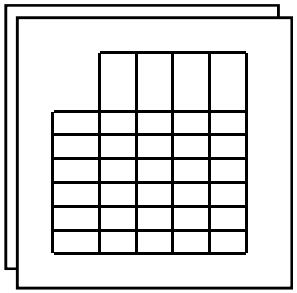
Use case model deliverables

- **Use case list**
- **Use case map**
- **Use case narratives**
- **Use case realizations**
- **Use case matrices**
  - **Relating roles, classes etc.**

Use cases

1. Place order
2. Cancel order
3. Apply for credit
4. Inquire on order
5. Record customer payment
6. Authorize return





Use case template

- filled out!

- **ID**
- **Name**
- **Purpose / goal**
- **Narrative**
- **Realization**
- **...**

Simplified example of use case narrative

Use case

Suspend telephone service

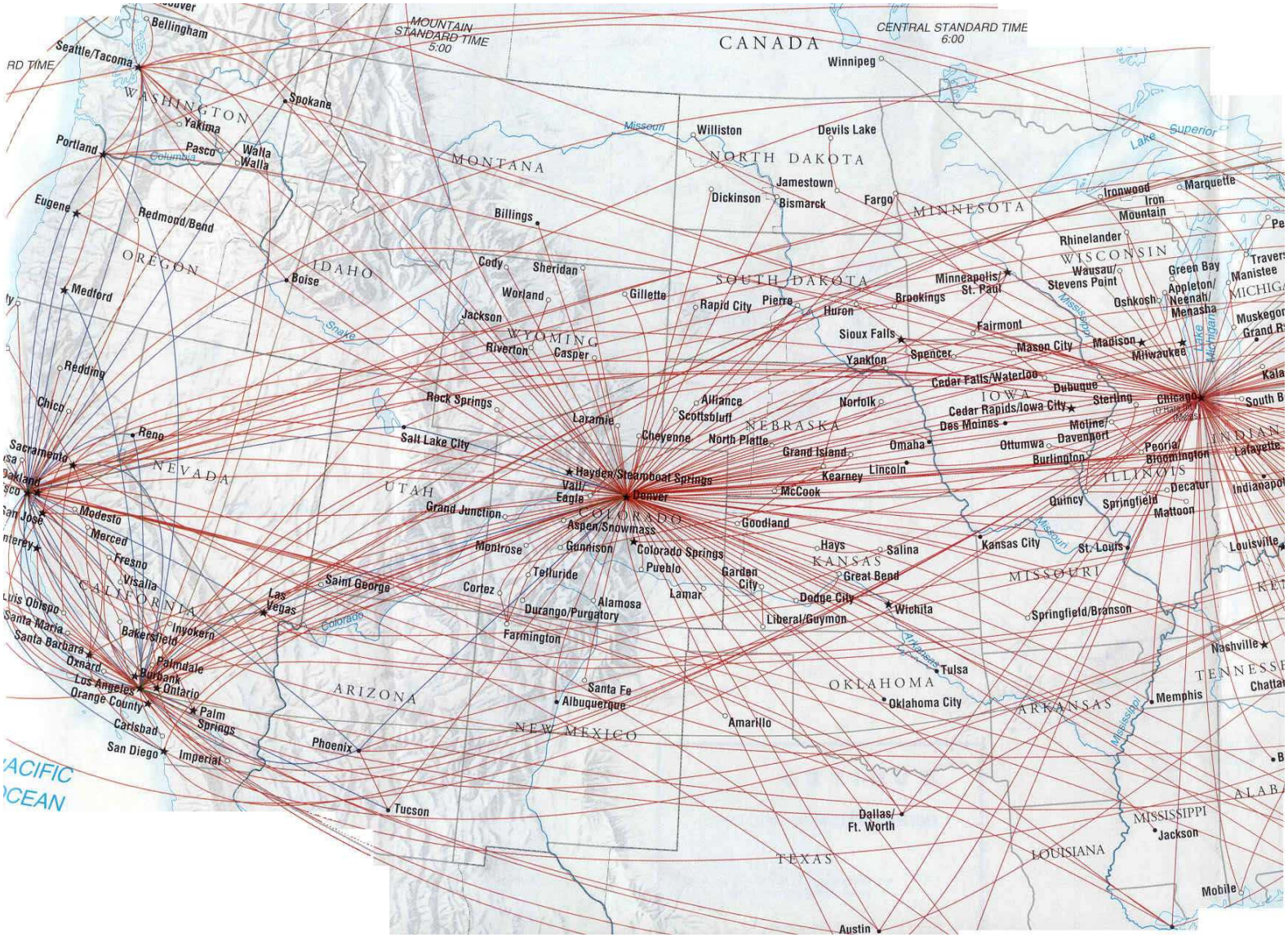
Purpose

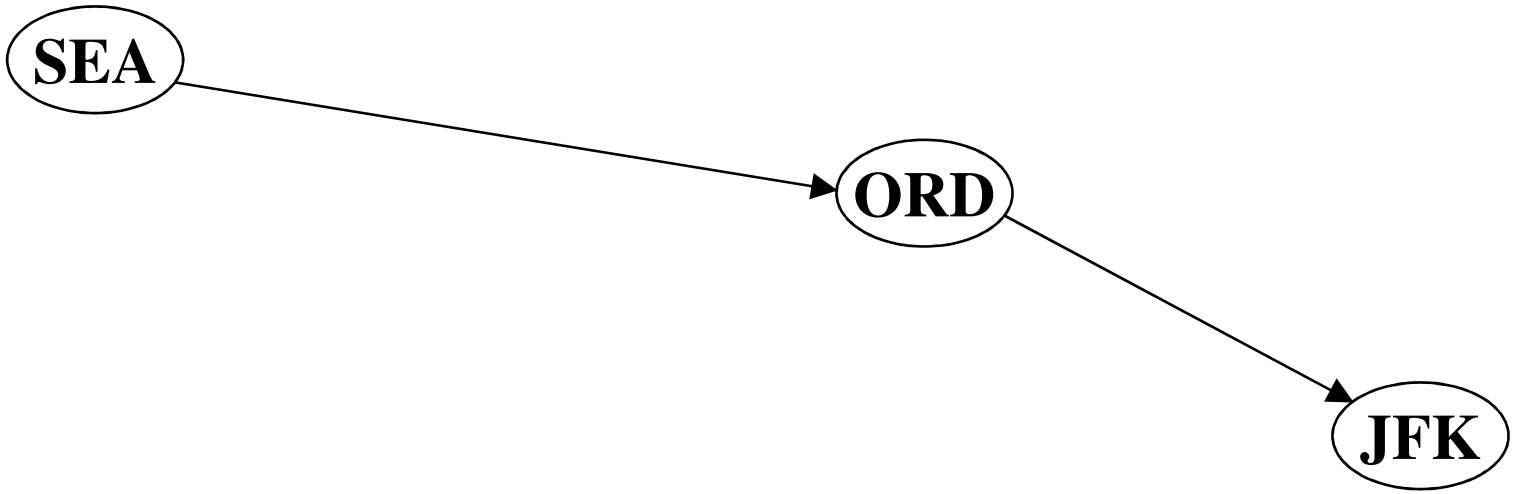
To suspend (voluntarily, temporarily) a customer's telephone service

Narrative

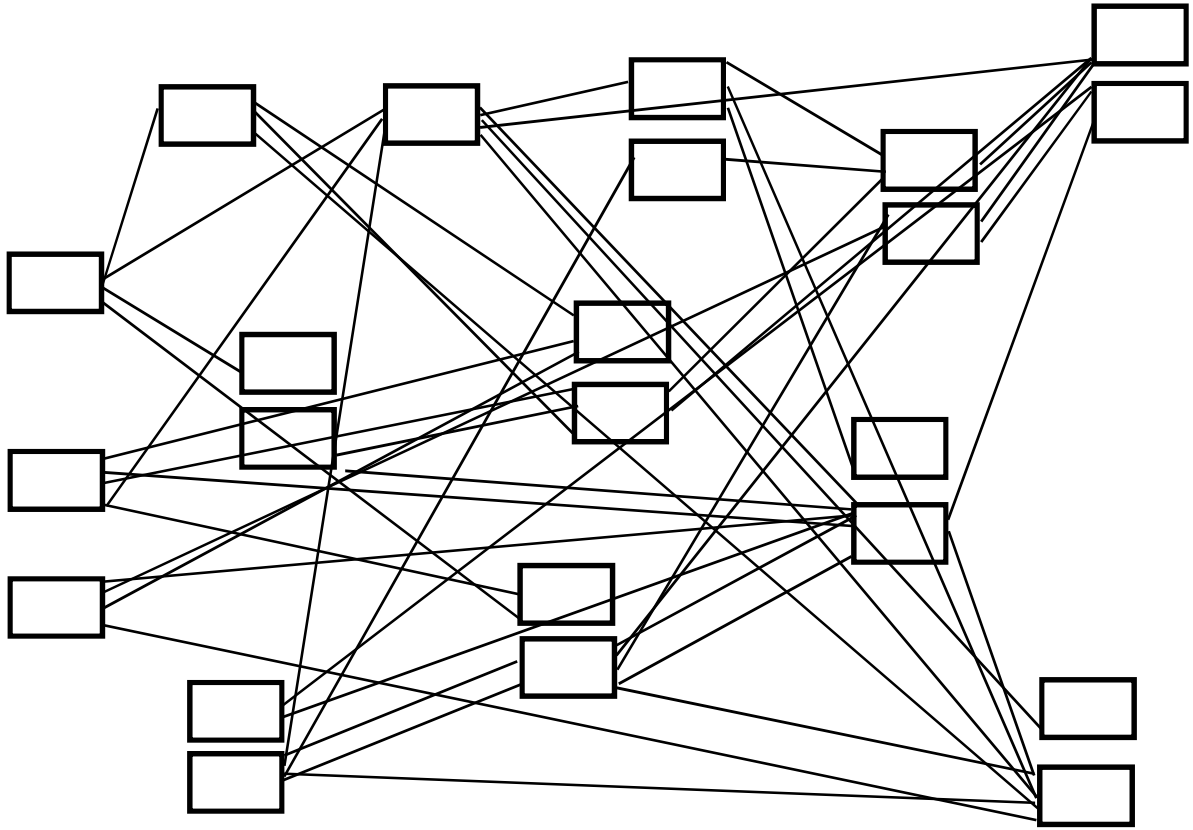
Actor	System
select subscription	
	display service details
select service(s) to suspend	
provide suspend / reinstate dates	
	validate dates
	calculate fees and savings
approve service suspensions	
	apply service suspensions

Use cases for partitioning

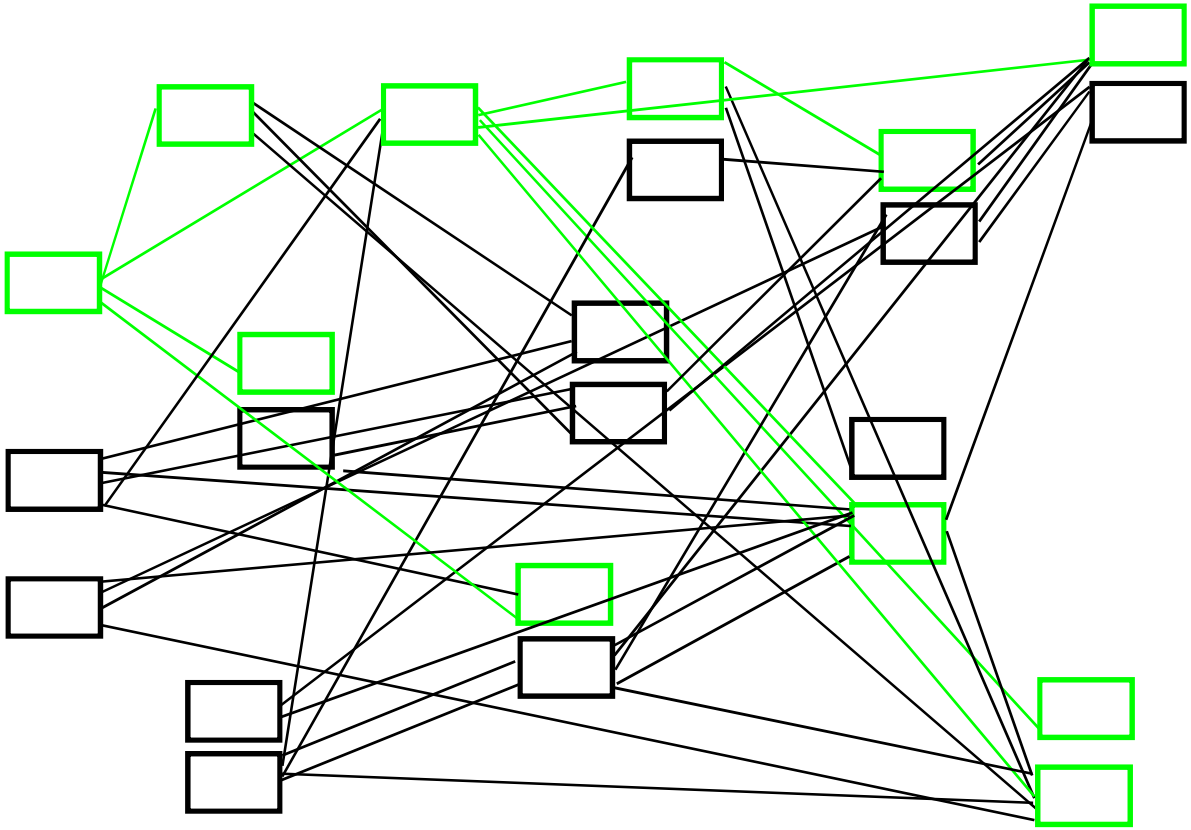




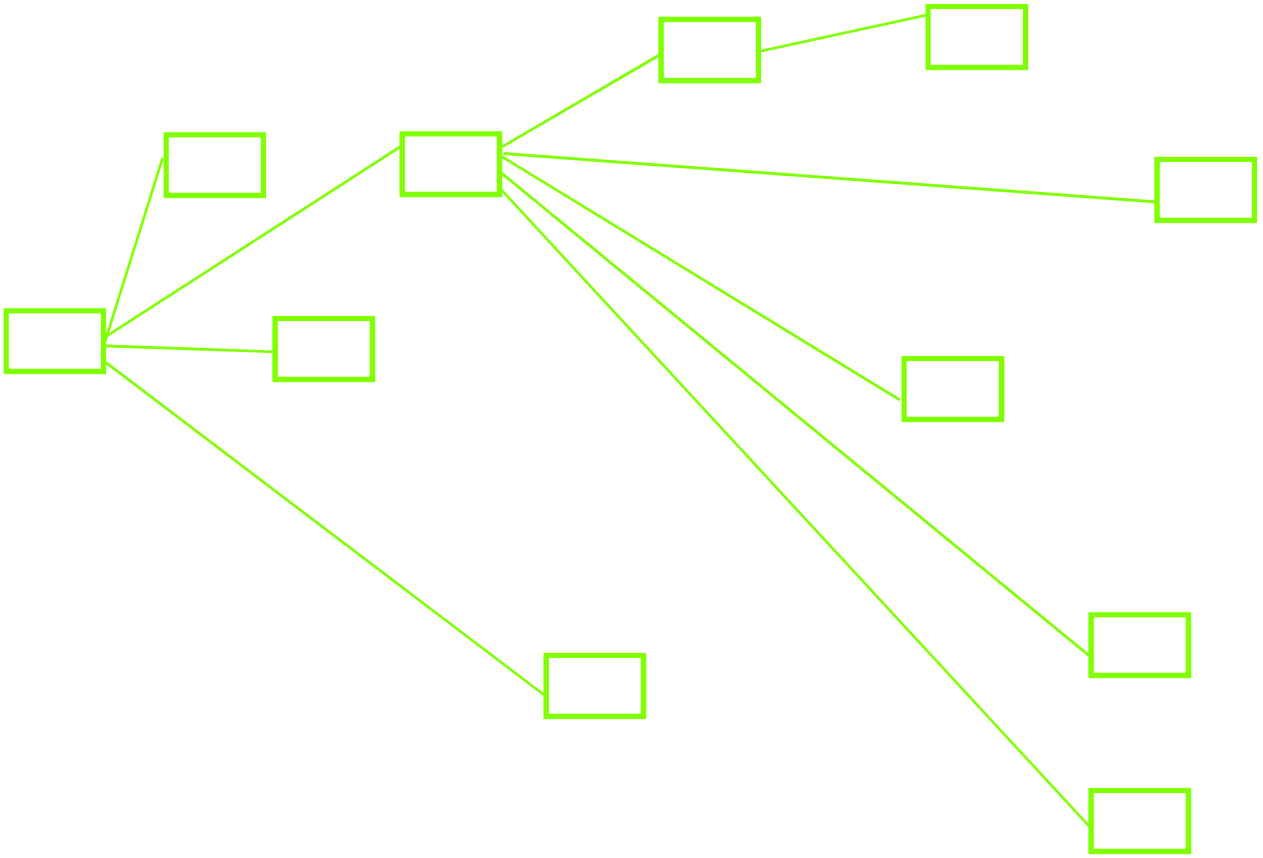
- **Object-collaboration diagram of whole system**
  - **A small system !**



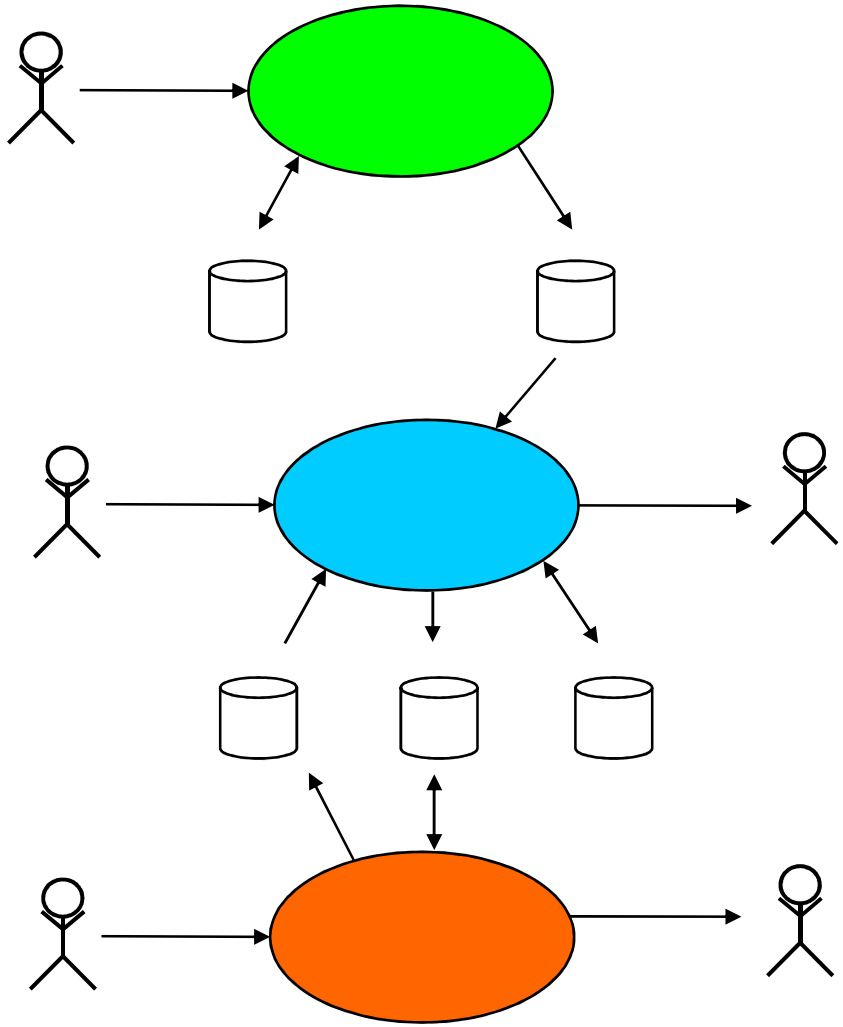
- **Trace of a single use case**



- **Single use case isolated**

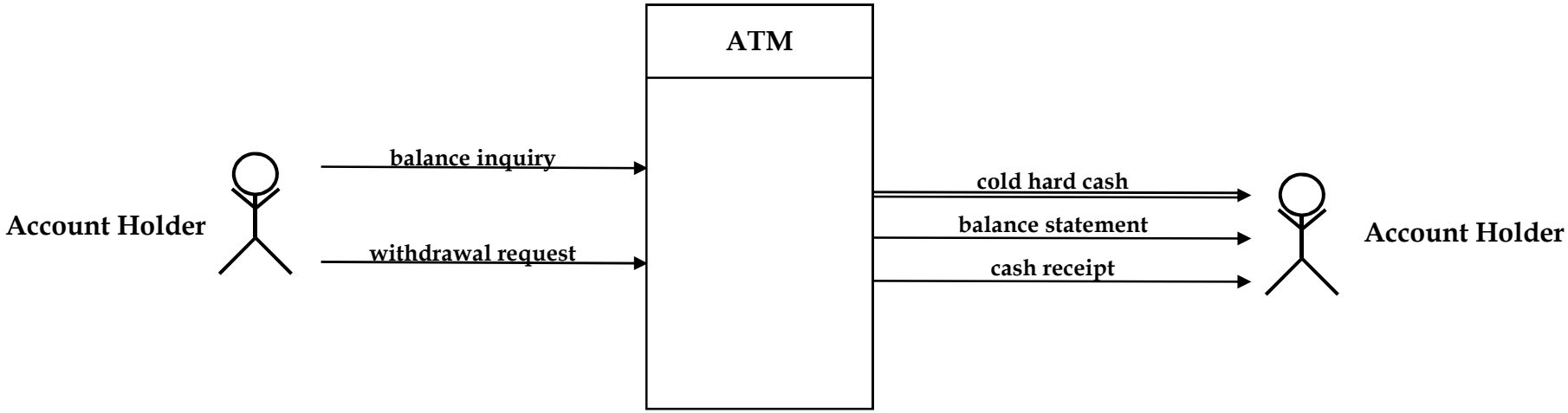


- **How use cases typically interact**
  - **Or don't!**



Use cases as definition of behavior

- For the system
  - And the actors
- Use cases also relate user actions and system actions
  - Example



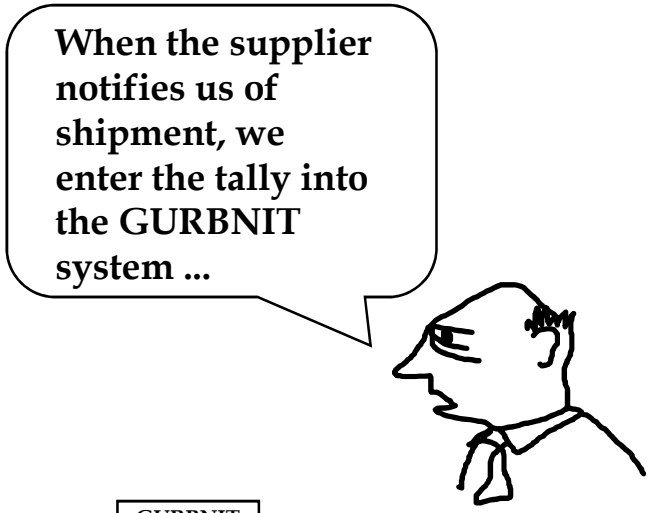
- Other models don't express behavioral aspects

Some benefits of use cases

- Can be worked on separately
  - Because they're "relatively independent"
- Are easy for users to understand and validate
- Provide a test for completion of analysis
- Provide the rationale for interface design
- Form the basis for future test scripts
- Form the "organizational backbone" for the project
  - Design
  - Construction
  - Delivery
  - Etc.

Who produces use cases - and from what?

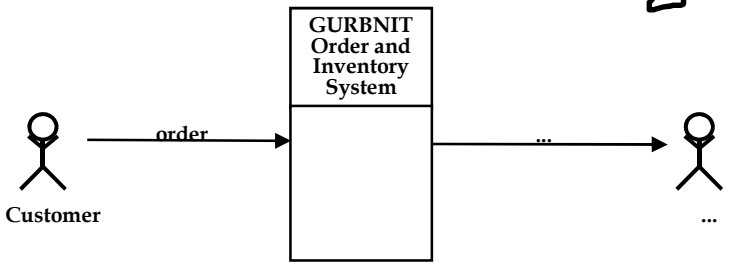
- **Requirements analysts**
- **Business folks**

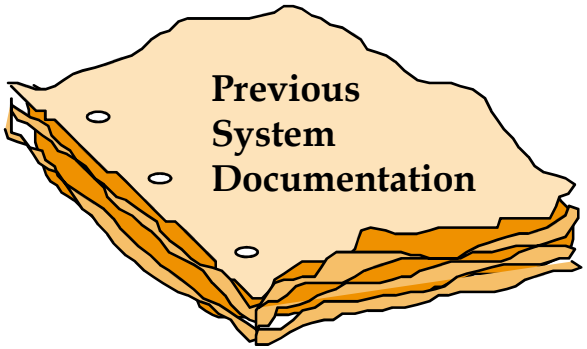


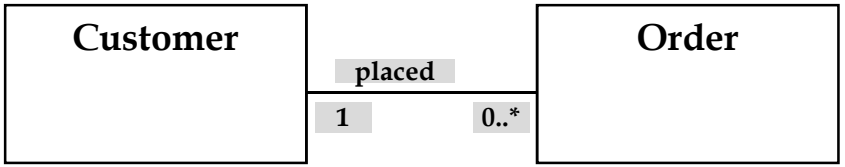
THE CHARTER

The system shall ...

•Issue an acknowledgement for each customer order







The 10 Don'ts (and Dos) of Modeling Use Cases

Meilir Page-Jones

Here are the Don’ts and Dos

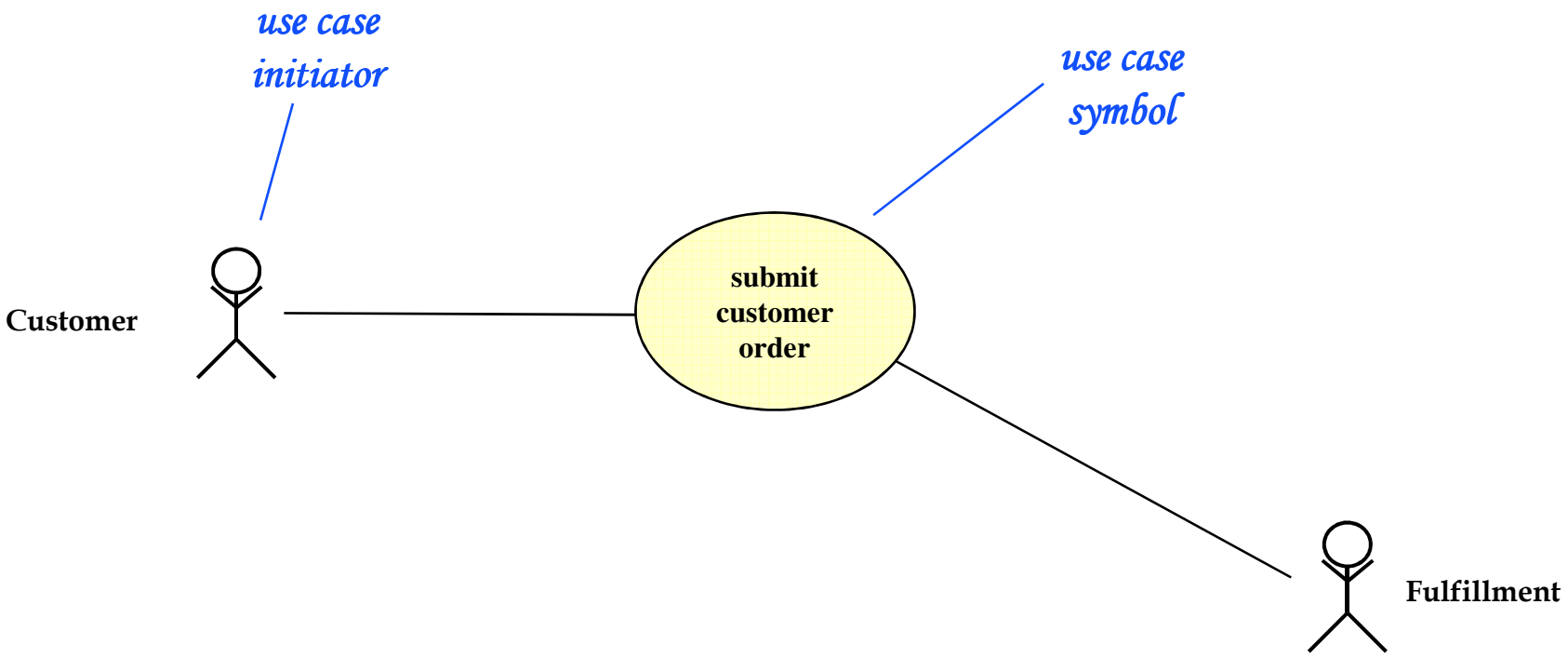
- **Don’t sit still for the puny official UML notation for use cases / actors**
  - **Do enhance the official standard notation**
- **Don’t specify implementation and UI details in the business use case**
  - **Do specify the "essential"**
- **Don’t specify use cases in an unreadable way**
  - **Do use structured English, ADs or SDs**
- **Don’t allow anarchy in use case style**
  - **Do use a standard template and style guides**

- **Don't try for a rigorous top-down hierarchy of use cases**
  - **Do focus on each use case is an individual piece of behavior**
- **Don't model use cases unnecessarily**
  - **Do use <<include>> or <<extend>> if appropriate**
- **Don't fight over <<include>> v. <<extend>>**
  - **Do pick something you can live with, regardless of the text books**

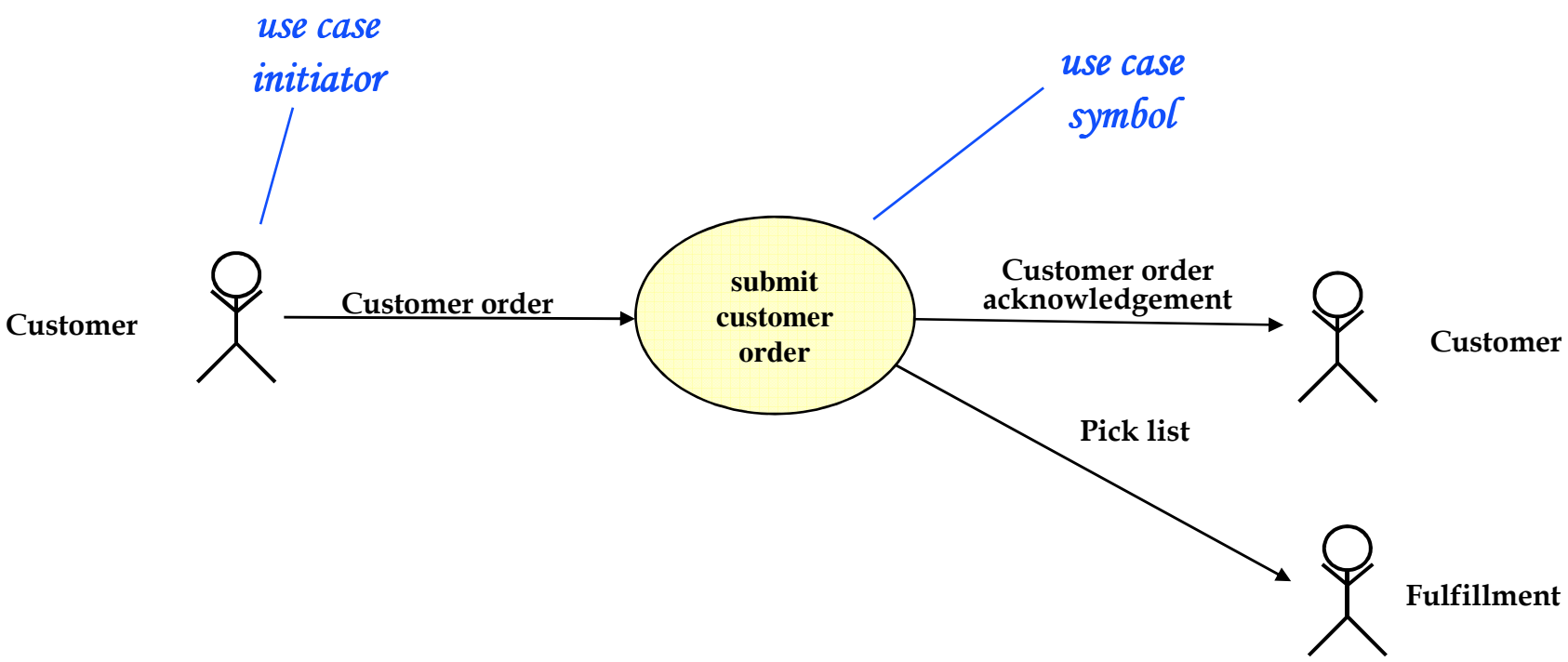
- **Don’t make use cases the only pillar of your business specification**
  - **Do make sure you have a class [E/R] diagram**
  - **Do use state diagrams, UI prototypes**
- **Don’t assume that “the system” is obvious to everybody**
  - **Do know the boundary of your system before you begin use cases**
- **Don’t rely only on brainstorming to come up with use cases**
  - **Do model events in order to get a complete list of use cases**

1. Don’t sit still for the puny official UML notation for use cases / actors

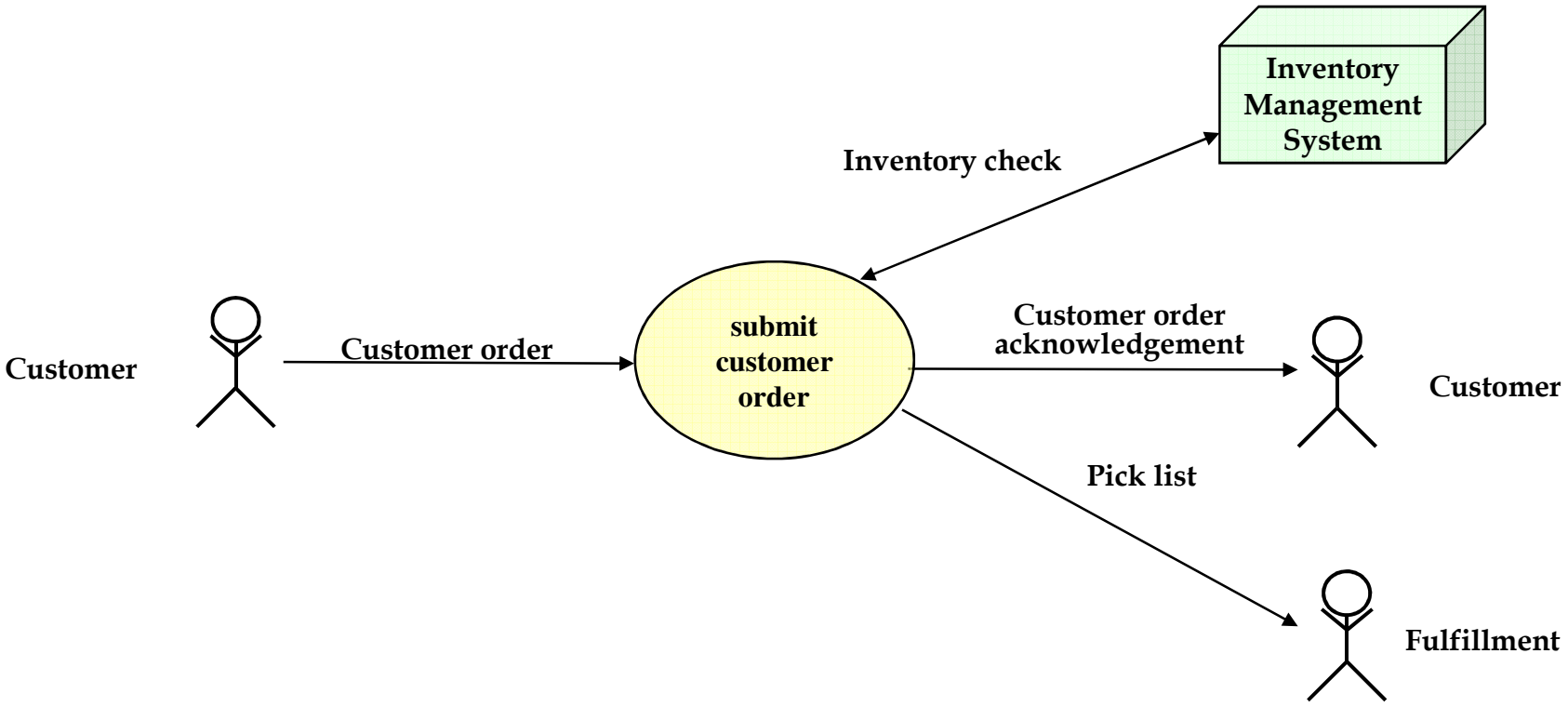
- Do enhance the official standard notation
- Standard UML for a use case



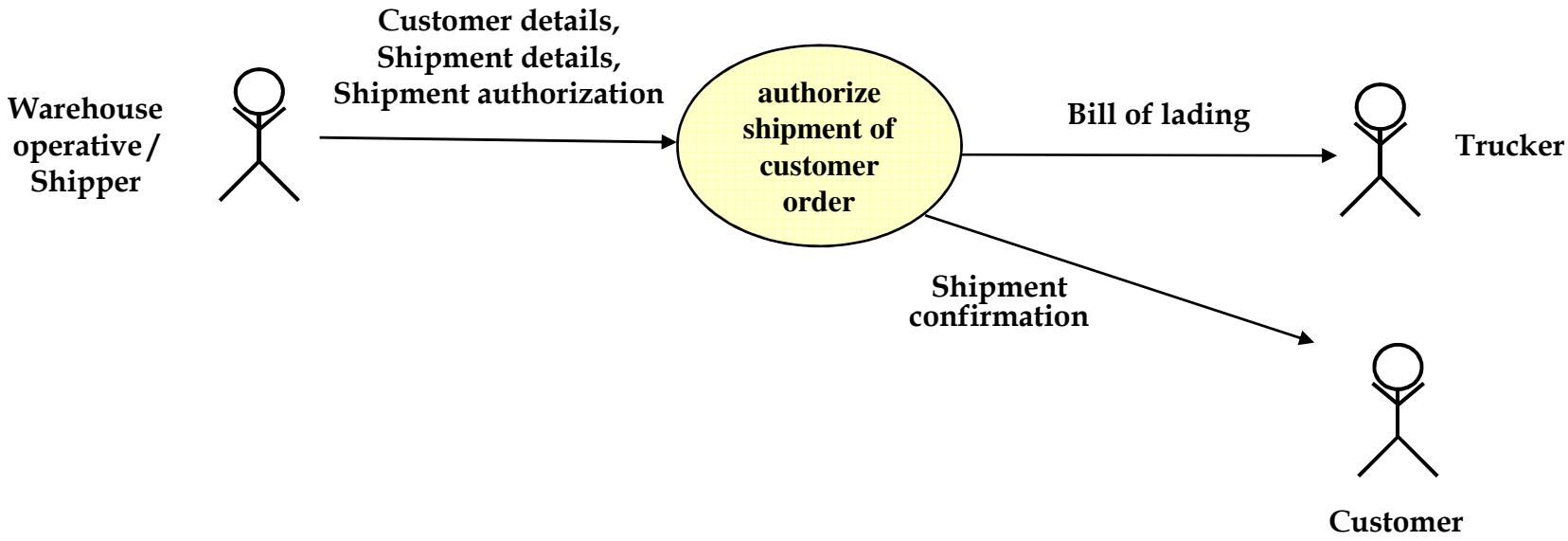
A small change will make a profound difference



- **And don't feel tied to stick figures**



- **Another example of a use case**



• Example of a use-case narrative

The warehouse operative clicks the left mouse button on the SEE-FILLED-ORDERS command button with focus on a particular customer. Then the system displays the first 10 orders for that customer that are filled and so are ready for shipping. If the warehouse operative wants to see more orders that are ready for shipping, then he/she clicks the left mouse button on the SEE-FILLED-ORDERS command button again. If there are no filled orders for that customer, then the system pops up a "NO FILLED ORDERS" window and the use case stops.

The warehouse operative then double-clicks the left mouse button on the particular order that he/she wants to authorize for shipment and the system displays the details of that order on a pop-up window named ORDER-DETAILS-WIN. (But if the customer is on the "no ship" list, then the system pops up a "CANNOT SHIP TO THIS CUSTOMER" window and the use case stops.)

This is the window where the warehouse operative can put in relevant shipment information such as the date and time of shipment, the trucking company, the truck ID and so on. The system validates this information and pops up a message if there are any problems with it. The warehouse operative has to re-enter the shipment information either until it's valid or until the warehouse operative gives up.

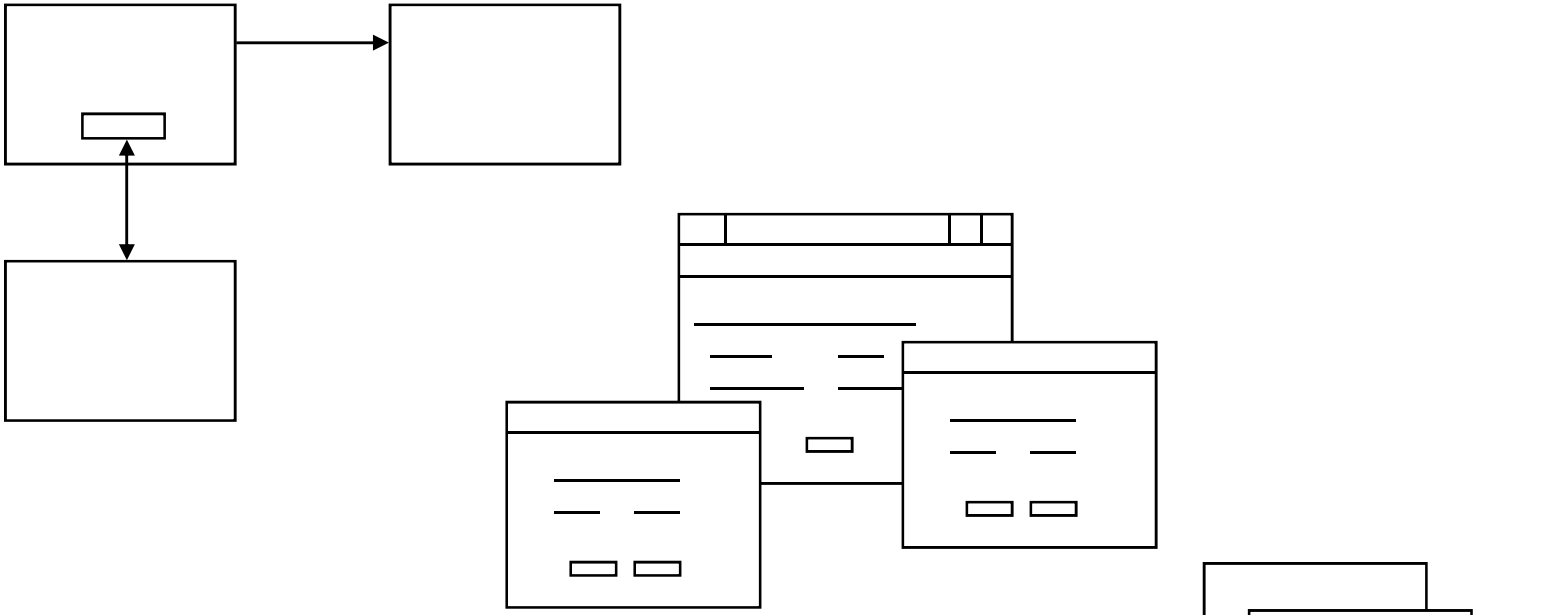
Then the warehouse operative clicks on a command button named SHIP-ORDER and the system updates tables in the database, such as X103-CUST-SHIPM and X117-CUST-SHIPM-ITEM , it e-mails a shipment confirmation to the customer and it prints out a bill of lading on the laser printer. He/she can then close this window.

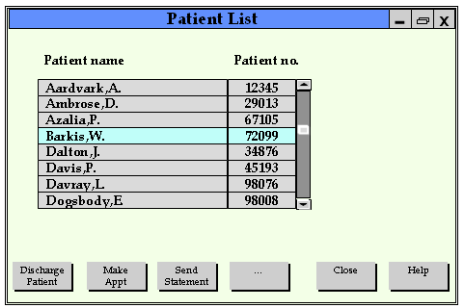
The warehouse operative takes the bill of lading off the printer and hands it to the trucker, who can then drive off with the customer order. The warehouse operative may repeat this procedure for each of the customer's filled orders.

2. Don’t specify implementation and UI details in the business use case

- **Where are such details in the example?**

- **Do develop the UI specification in a separate, but related, model**





- **Related topic … what's the distinction lurking in this group of candidate use cases**
  - **Synchronize distributed server files**
  - **Place customer order**
  - **Record receipt of vendor shipment**
  - **Archive ancient orders**
- **Specify only the "business-essence" use cases**
  - **Perfect technology!**
- **OK to keep a separate list of implementational use cases**

3. Don't specify use cases in an unreadable way

- Don't use acres of deathly prose
- Do use
  - An activity diagram, or
  - A sequence diagram, or
  - Structured English
- Don't get wrapped around business alternatives
  - And beware of separating alternatives in unreadable or unmaintainable ways

Two-column style

Warehouse operative	Application
1. Request filled customer orders for a particular customer	
	2. Display list of filled customer orders for that customer
3. Select customer order	
	4. Display details of selected customer order
5. Enter shipment information	
	6. Validate shipment information
7. Authorize shipment of customer order	
	8. Record customer order as shipped
	9. Update customer order database
	10. Issue order shipment confirmation
	11. Create bill of lading
	12. Issue bill of lading

Alternatives in two-column style

[Customer is on “no ship” list]

After step 2:

Display “cannot ship to this customer” message

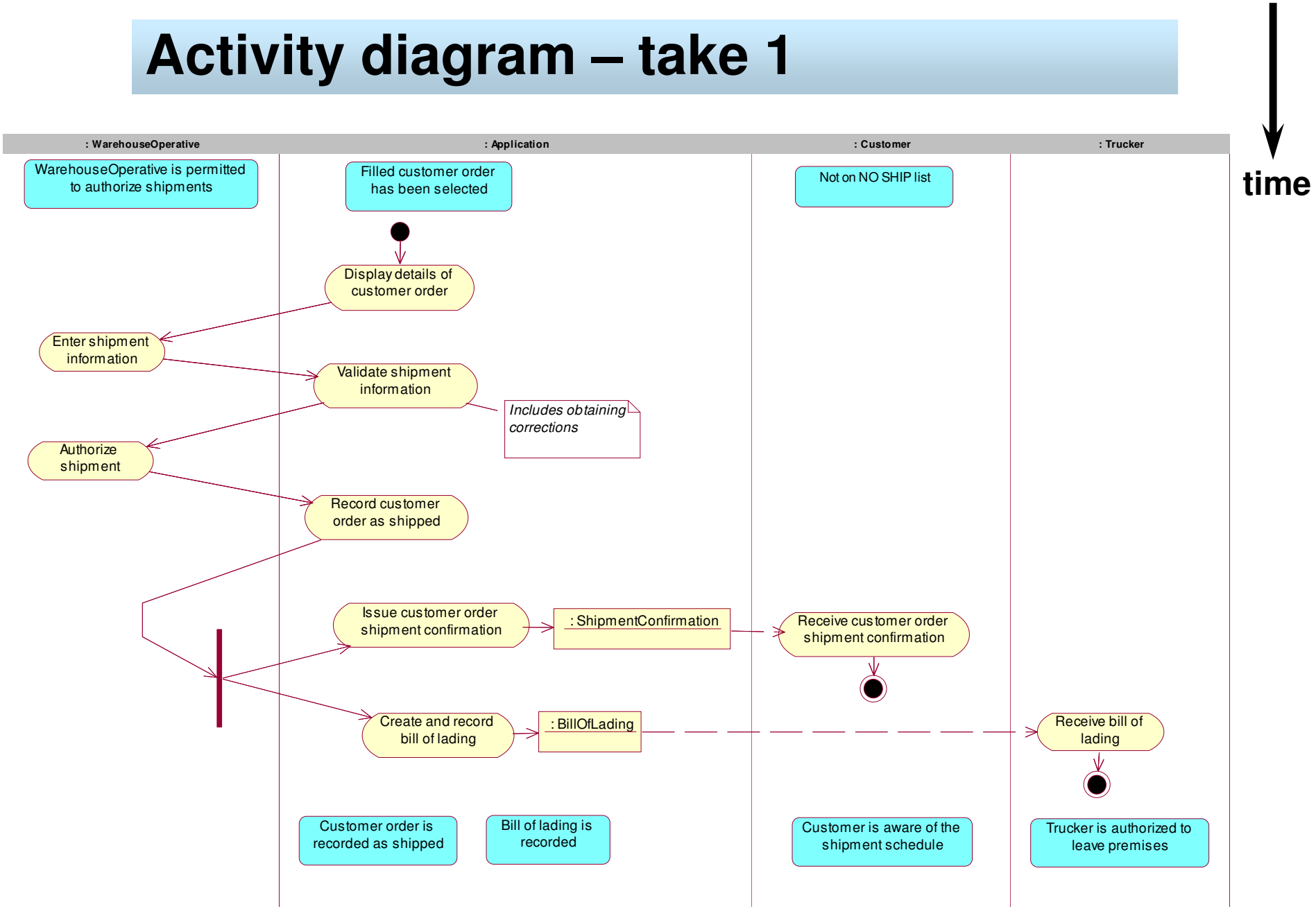
END

[Customer has multiple filled orders]

Repeat steps 3 thru 12 for each filled order

or until WO quits the use case

Activity diagram – take 1



- **Barebones**

WO: Request details of a single order

APP: Display order details

WO: Enter shipment information

APP: Validate shipment information

WO: Authorize order shipment

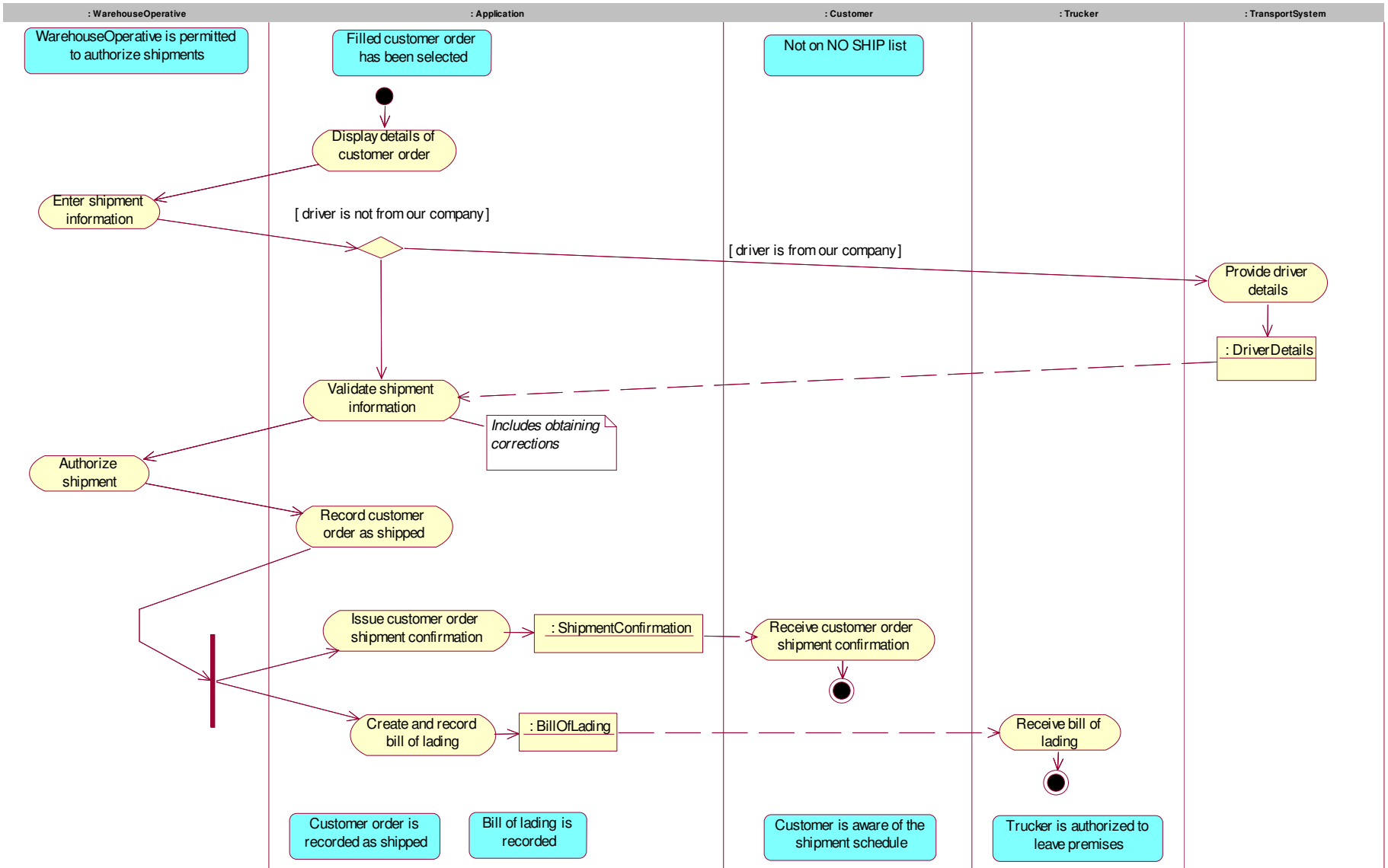
APP: Update database

APP: Issue shipment confirmation

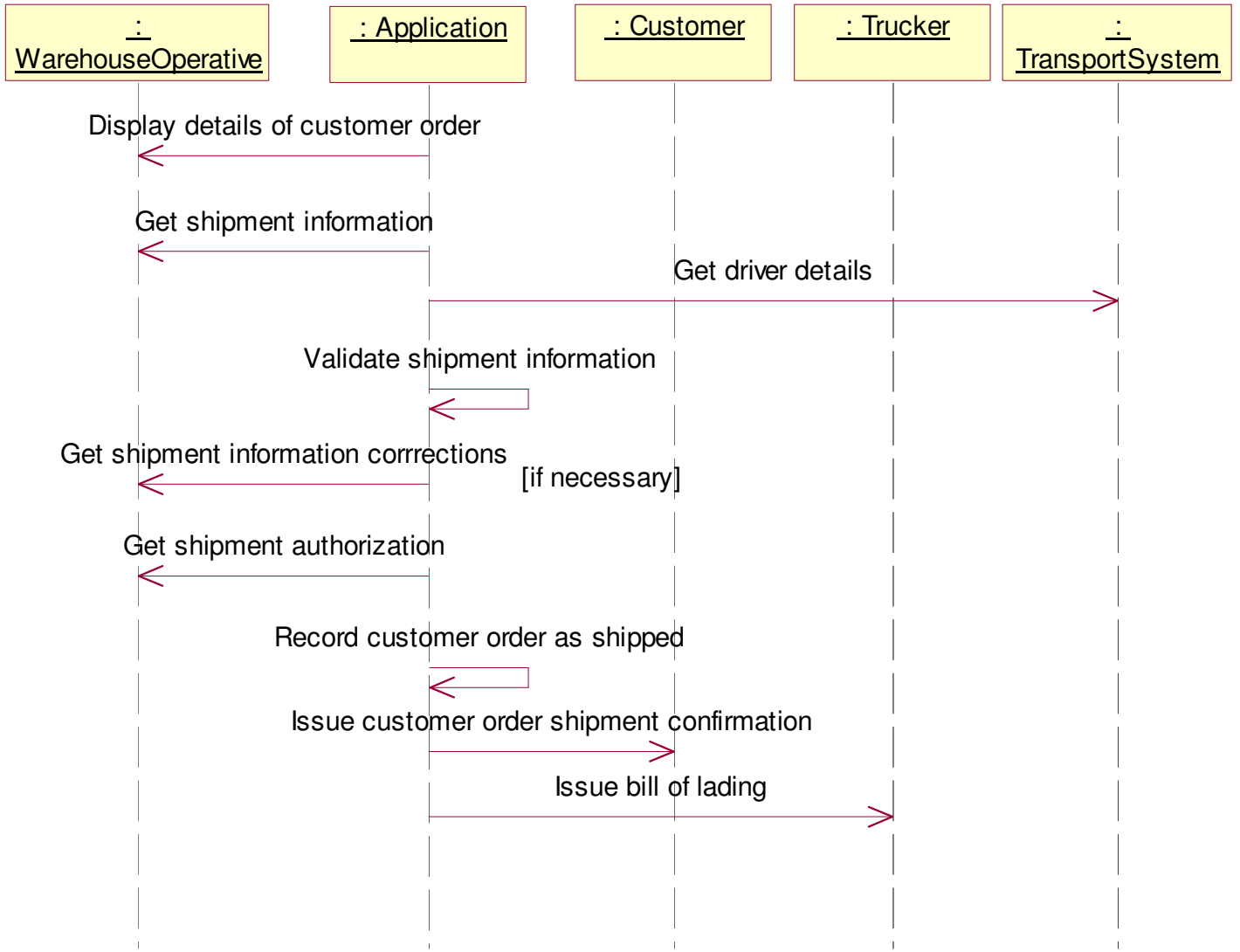
APP: Create bill of lading

APP: Issue bill of lading

Activity diagram – take 2



Sequence diagram



Structured English

Input from WO

Customer order details
- Customer order id

Shipment details
- Shipment date
- Shipment time
- Trucking company id
- Truck id
- If our trucking company:
  Driver id, Truck plate num
- If not our trucking company:
  Driver id, Driver name, Truck plate num

Shipment authorization
- …

Precondition

Filled customer order has been selected (by WO)
Customer is not on the NO SHIP list

Output

Bill of lading
Shipment confirmation

Postcondition

Customer order is recorded as shipped

Narrative steps

display details of selected customer order

WO enters shipment details

If truck is from our company
Then retrieve Driver details from TransportSystem
Endif

validate shipment details

WO enters shipment authorization

record customer order as shipped

issue shipment confirmation to customer

create and record bill of lading

issue bill of lading to Trucker

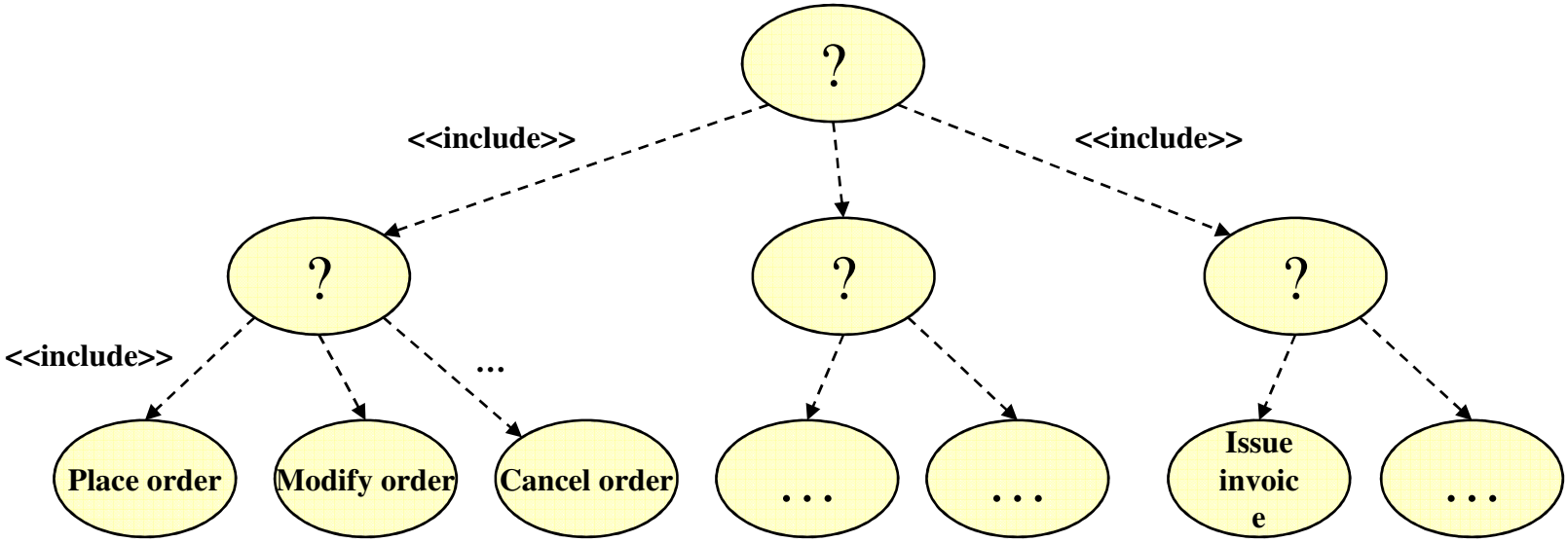
4. Don’t allow anarchy in use case style

Use case ID	UC#150
Use case name	Authorize shipment of customer order
Use case purpose	To notify the system that a vehicle from a transport company is ready to embark with the goods for a customer’s order. The system changes the status of the order and produces a trucker’s bill of lading to accompany the shipment.
Planned release	1.0
Initiating actor [role]	Warehouse operative [Shipper]
Triggering business event	Warehouse operative [Shipper] ships customer order.
Precondition	Filled customer order has been selected (by WO) Customer is not on the NO SHIP list
Postcondition	Customer order is recorded as shipped.
Business effect	Vehicle may leave the premises (once the trucker has the Bill of lading). Customer is informed of order shipment.
Input (stimulus)	Customer details, Shipment details, Shipment authorization
Output (response)	Bill of lading, Shipment confirmation
Assumptions	Each order is shipped in its entirety.
Other business rules	Shipment date must not be in the future.
QoS constraints	Must be available at least 16 hrs / day
Other requirements	Must be usable in high-pressure, distracting environment
Use case narrative	*[See below]*
Use case realization	*[See below]*
Use case issues	Are there any special considerations for shipments with an ultimate foreign destination?

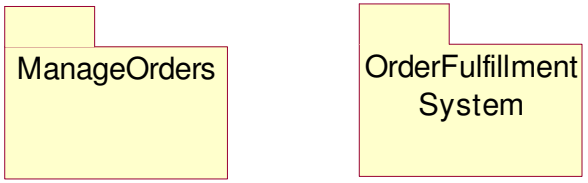
- **Do use a standard template**
- **Do pick a default approach to use case narrative**
  - **But allow flexibility for special cases**
- **Do establish style guides**
  - **Needless variation can be distracting**
  - **Styles apply to diagrams as well as text**

- **Does a single style fit all use cases?**
- **Is natural language better than a more structured form?**
- **Is a diagram superior to text?**
- **Are numbers for steps a good idea?**
- **How best to distinguish alternatives?**
  - **Is it enough simply to list them, just so they're not overlooked?**
  - **Are if and for-each a good idea?**
- **Should the narrative contain *repeats***
  - **To handle multiple items?**
  - **To deal with retries?**
- **What about validation steps?**
  - **And should early termination of the use case be noted?**

5. Don't try for a rigorous top-down hierarchy of use cases

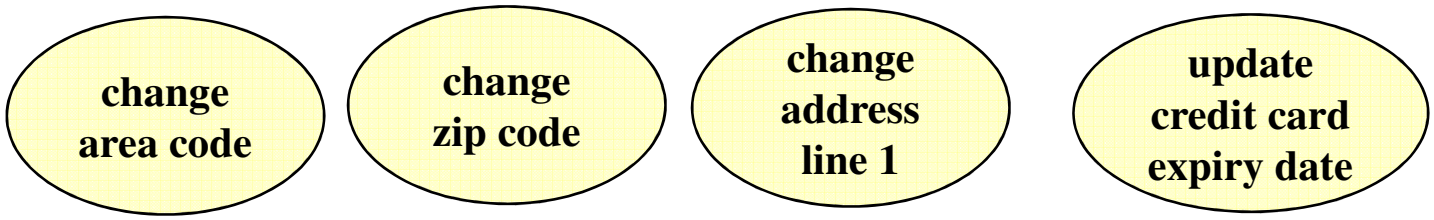


- **Do focus on each use case is an individual piece of behavior**
  - **Packages of use cases are fine**



6. Don't model use cases unnecessarily

- What do you think of these use cases?

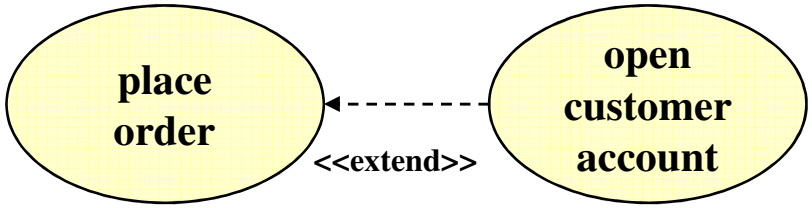


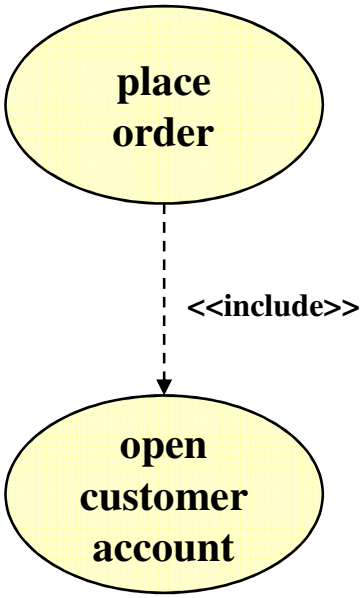
- Do use <<include>> or <<extend>>
  - But only where appropriate!

- Don't create elaborate use cases for straightforward reports

7. Don't fight over <<include>> versus <<extend>>

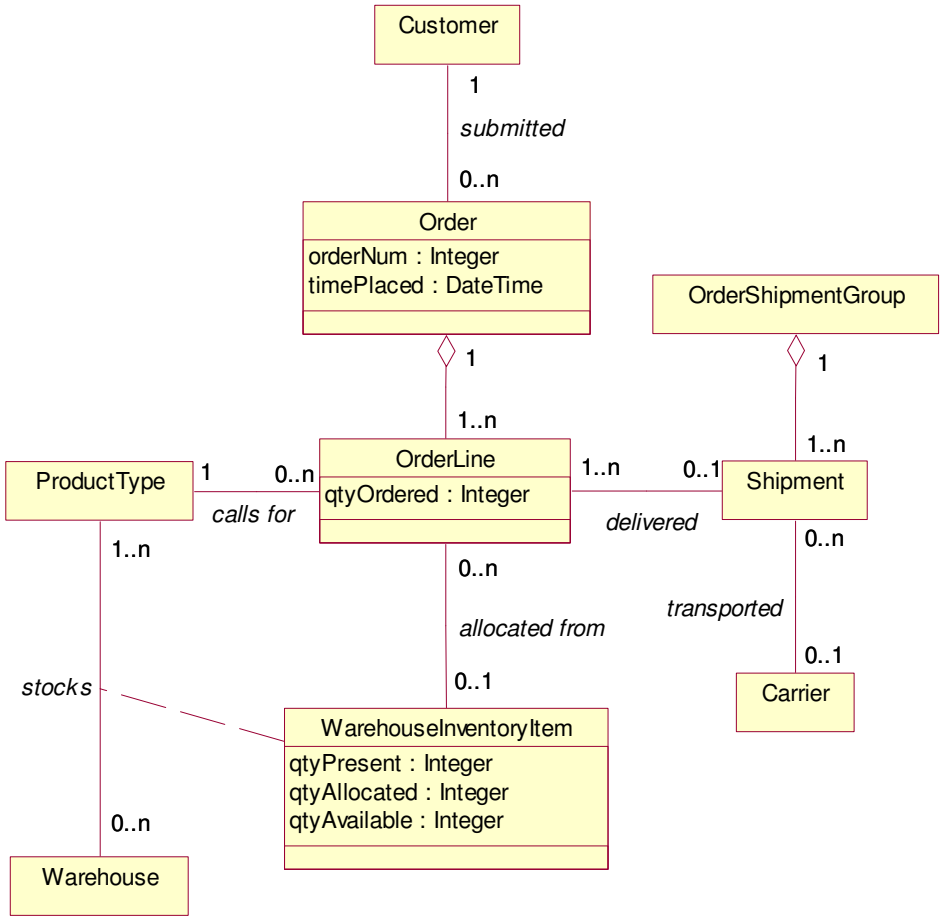
- **Do pick something you can all live with, regardless of the text books**





8. Don’t make use cases the only pillar of your business specification

- **Do make sure you have a class [E/R] diagram**

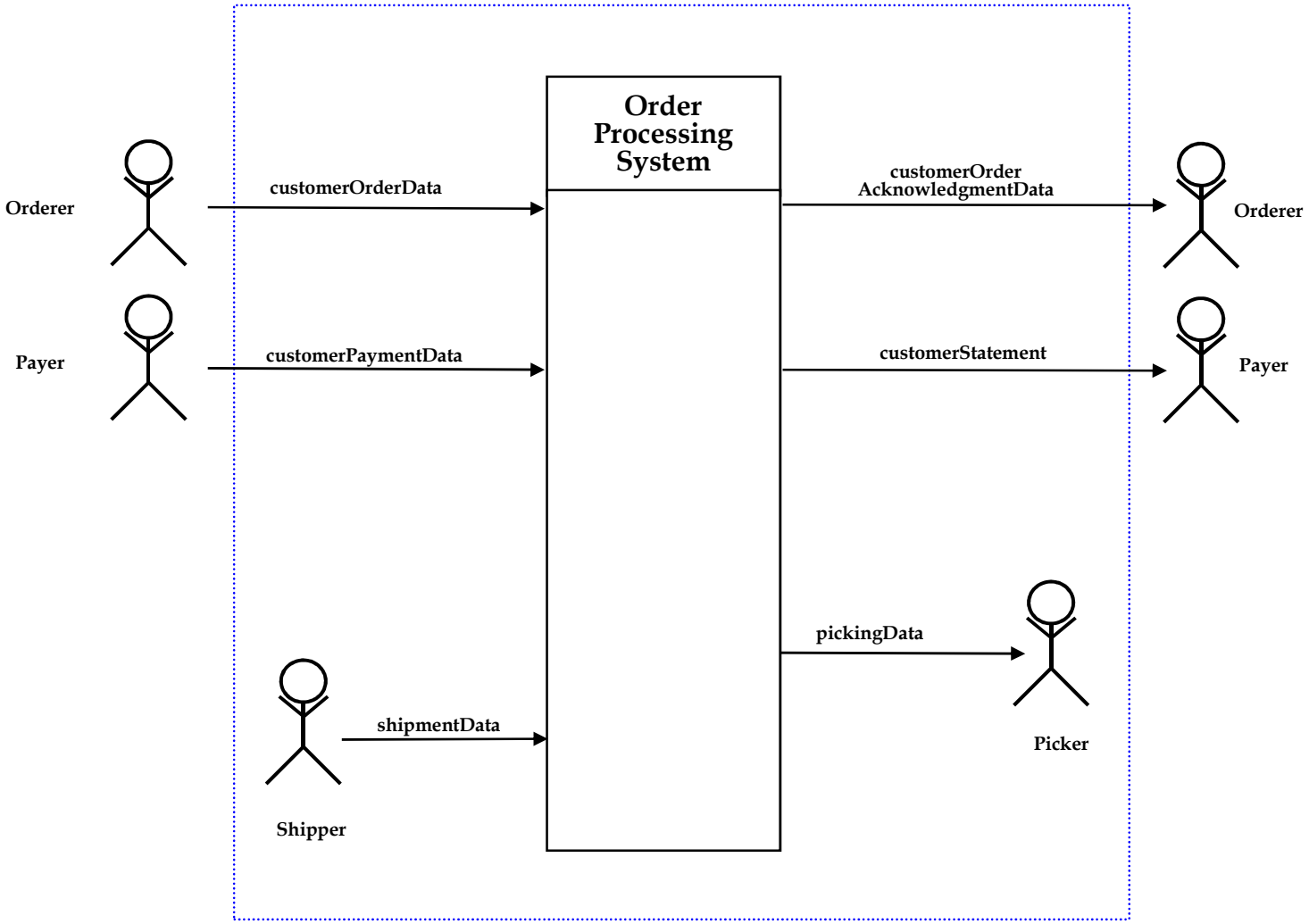


- **Also use state diagrams, UI prototypes**

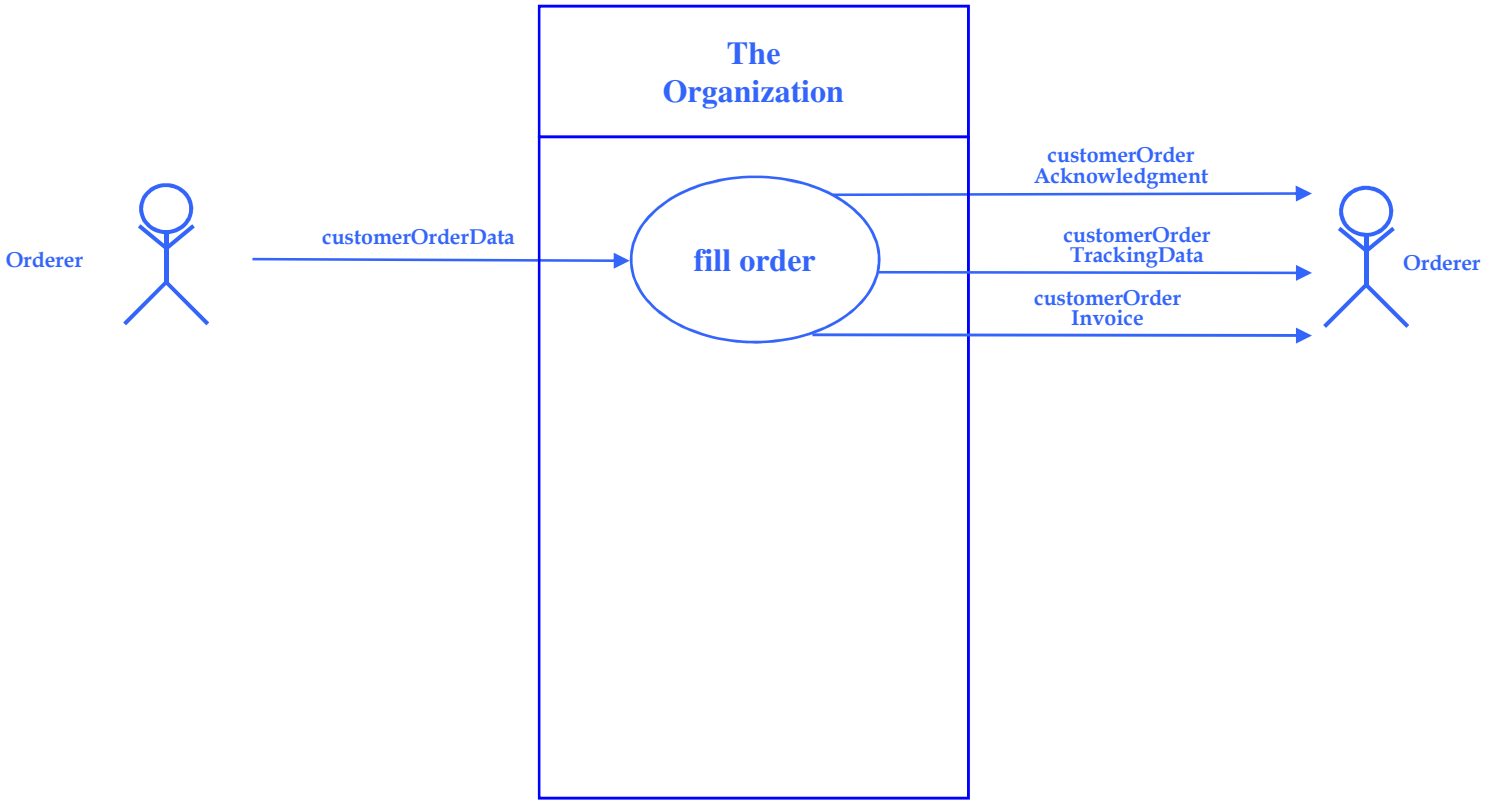
9. Don’t assume that “the system” is obvious to everybody

- **Do know the boundary of your system before you begin use cases**
  - **Is each use case a business or a system use case?**

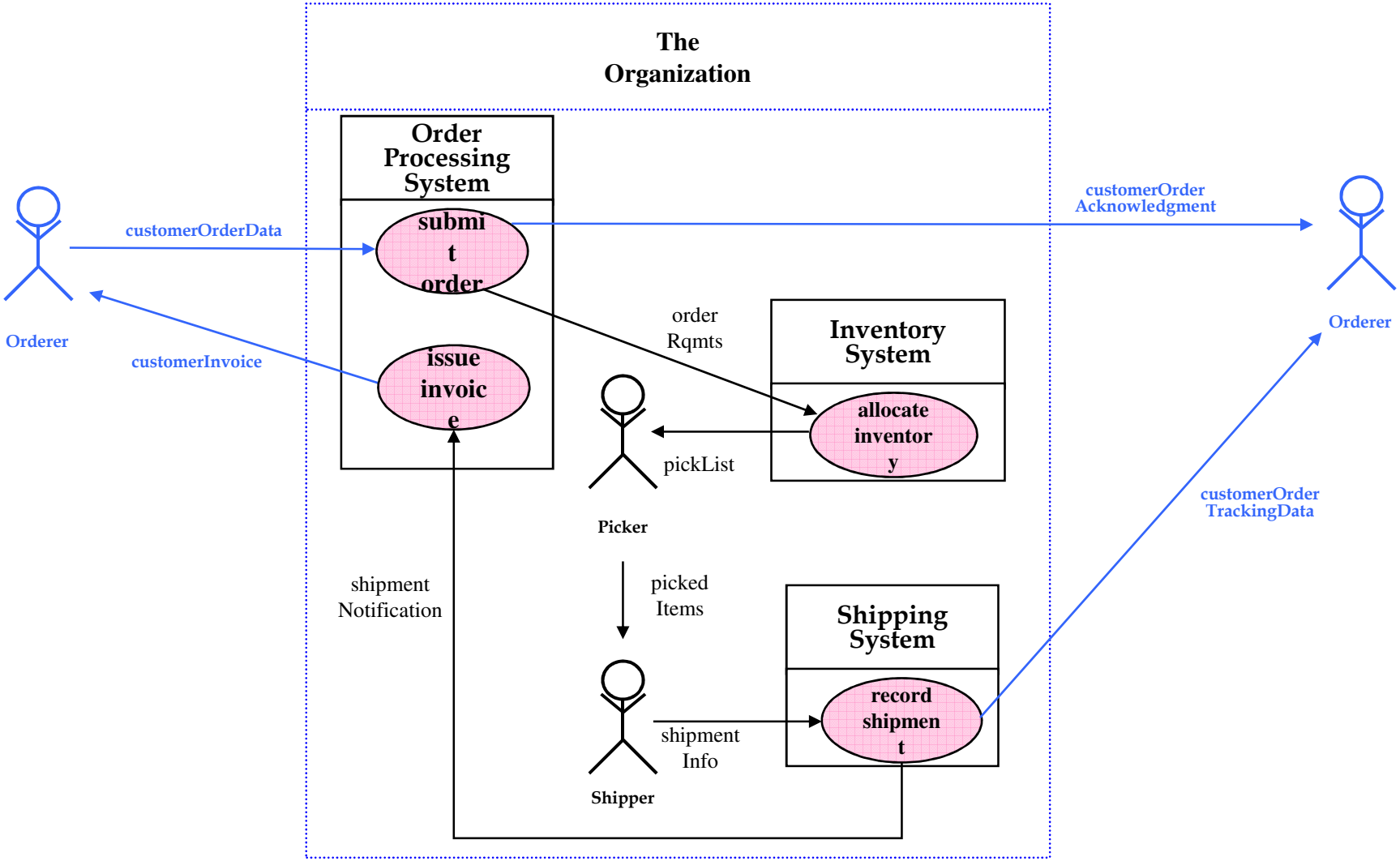
- **What is “the system”?**



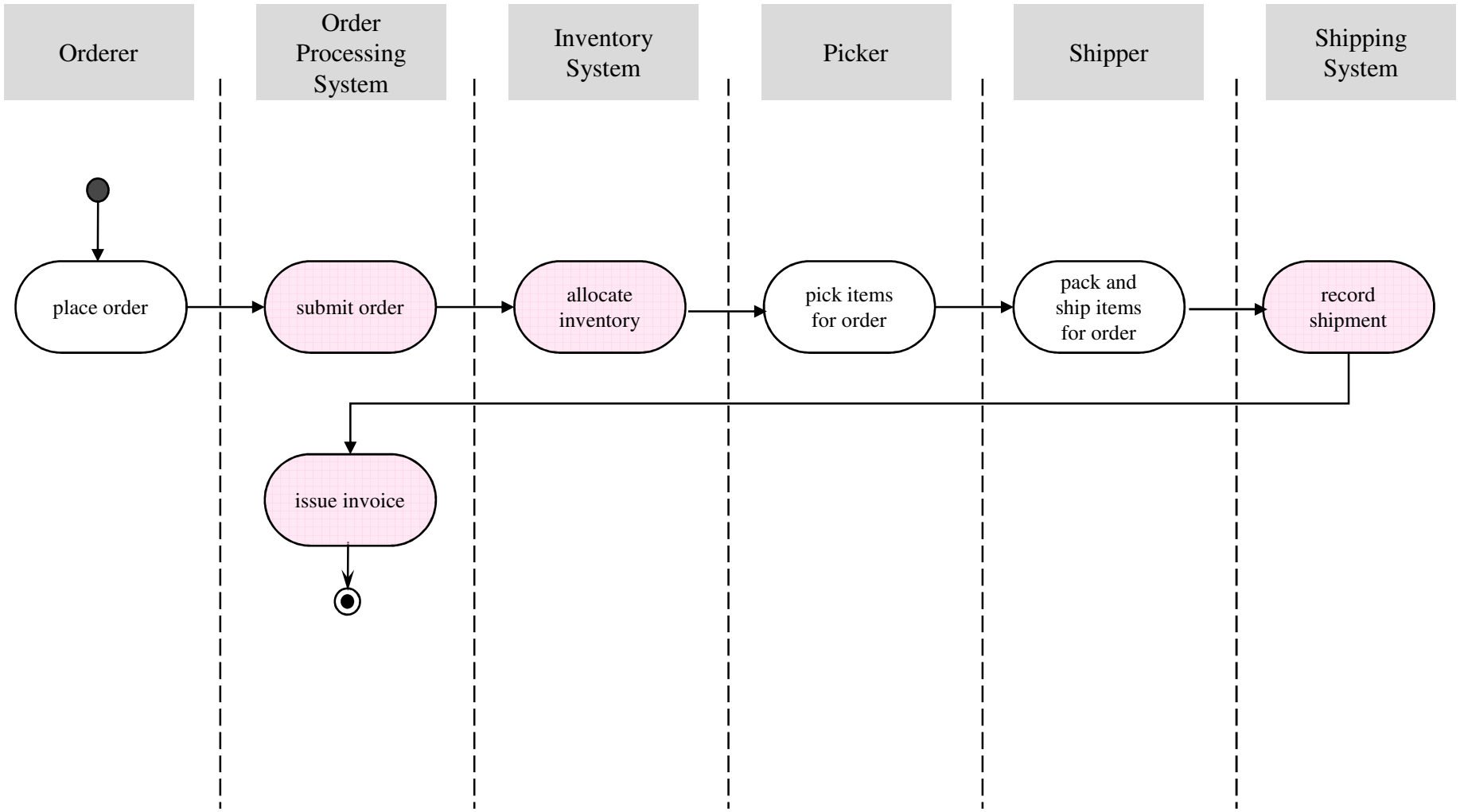
- **Business use case**



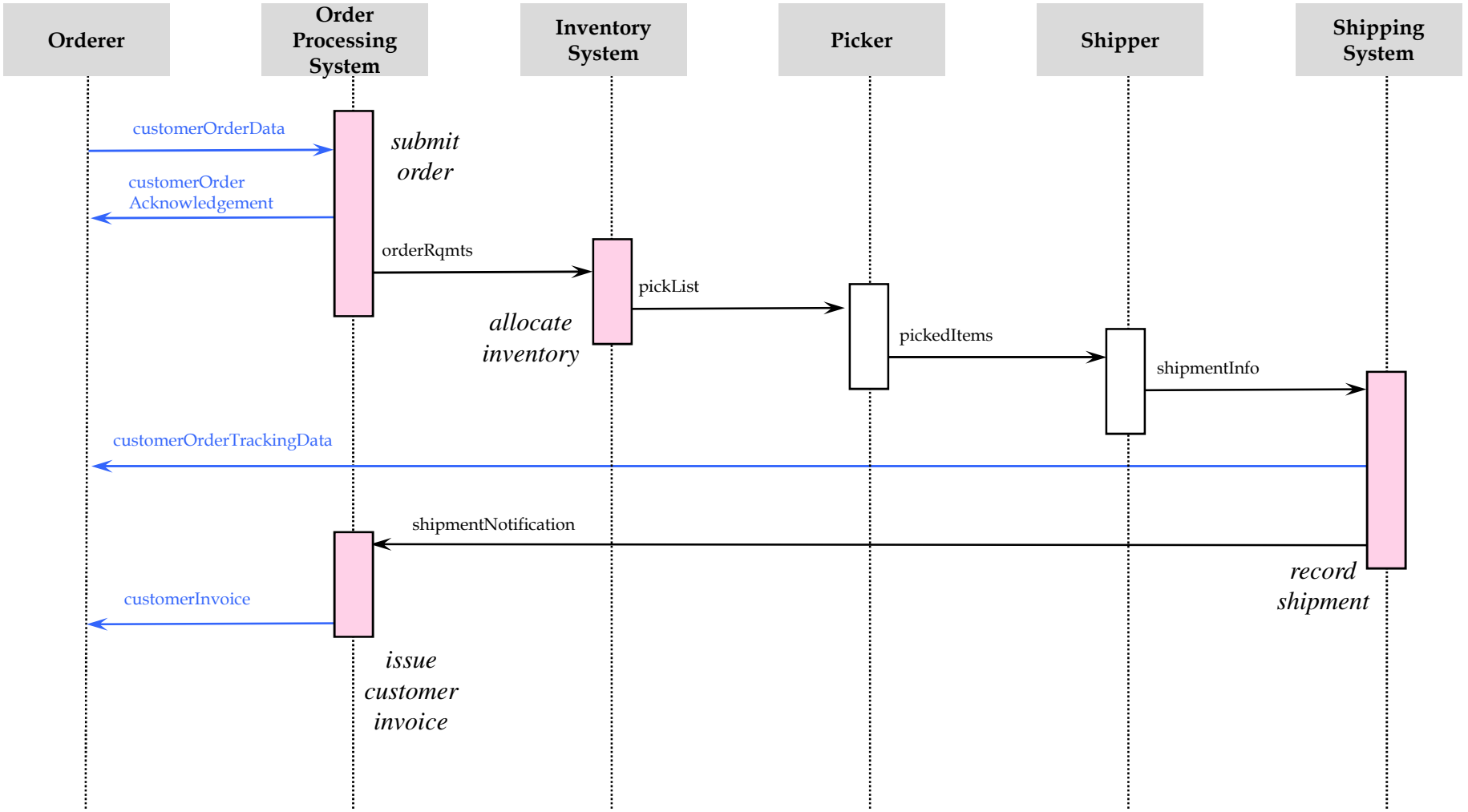
- **System use cases**



- **System use cases as an activity diagram**



- **System use cases as a sequence diagram**



10. Don’t rely only on brainstorming to come up with use cases

- Do model events in order to get a complete list of use cases

- True business events external to “our system”
  - Under control of the environment
  - Detectable by our system
  - Relevant to our system

- **Good examples**
  - Customer places phone call
  - … to issue monthly bill
  - Customer pays monthly bill
  - Customer … pay monthly bill
  - Customer changes primary residence

- **Bad examples**
  - Customer is a premium payer
  - Our system e-mails profile-change confirmation
  - Customer goes to sleep

- **Each event (including non-events!) leads to a use case**
  - **More or less**

Event	Use case
Customer completes phone call	Record call details
Time to issue monthly bill	Issue monthly bill
Customer pays monthly bill	Apply customer payment
Customer fails to pay monthly bill	Issue reminder notice
Customer changes primary residence	Update customer profile

- **Event modeling is**
  - **Fast**
  - **Quite precise**
  - **Freer of team wrangling than use case modeling**

- **Event modeling and use case modeling are nicely complementary**

Summary

- **Don’t sit still for the puny official UML notation for use cases / actors**
  - **Do enhance the official standard notation**
- **Don’t specify implementation and UI details in the business use case**
  - **Do specify the "essential"**
- **Don’t specify use cases in an unreadable way**
  - **Do use structured English, ADs or SDs**
- **Don’t allow anarchy in use case style**
  - **Do use a standard template and style guides**

- **Don't try for a rigorous top-down hierarchy of use cases**
  - **Do focus on each use case is an individual piece of behavior**
- **Don't model use cases unnecessarily**
  - **Do use <<include>> or <<extend>> if appropriate**
- **Don't fight over <<include>> v. <<extend>>**
  - **Do pick something you can live with, regardless of the text books**

- **Don't make use cases the only pillar of your business specification**
  - **Do make sure you have a class [E/R] diagram**
  - **Do use state diagrams, UI prototypes**
- **Don't assume that "the system" is obvious to everybody**
  - **Do know the boundary of your system before you begin use cases**
- **Don't rely only on brainstorming to come up with use cases**
  - **Do model events in order to get a complete list of use cases**