INTRODUCTION TO BEHAVIOURAL MODELLING

DR. VIVEK NALLUR

VIVEK.NALLUR@SCSS.TCD.IE

OUTLINE OF THIS TALK

- Sequence Diagrams
- Activity Diagrams

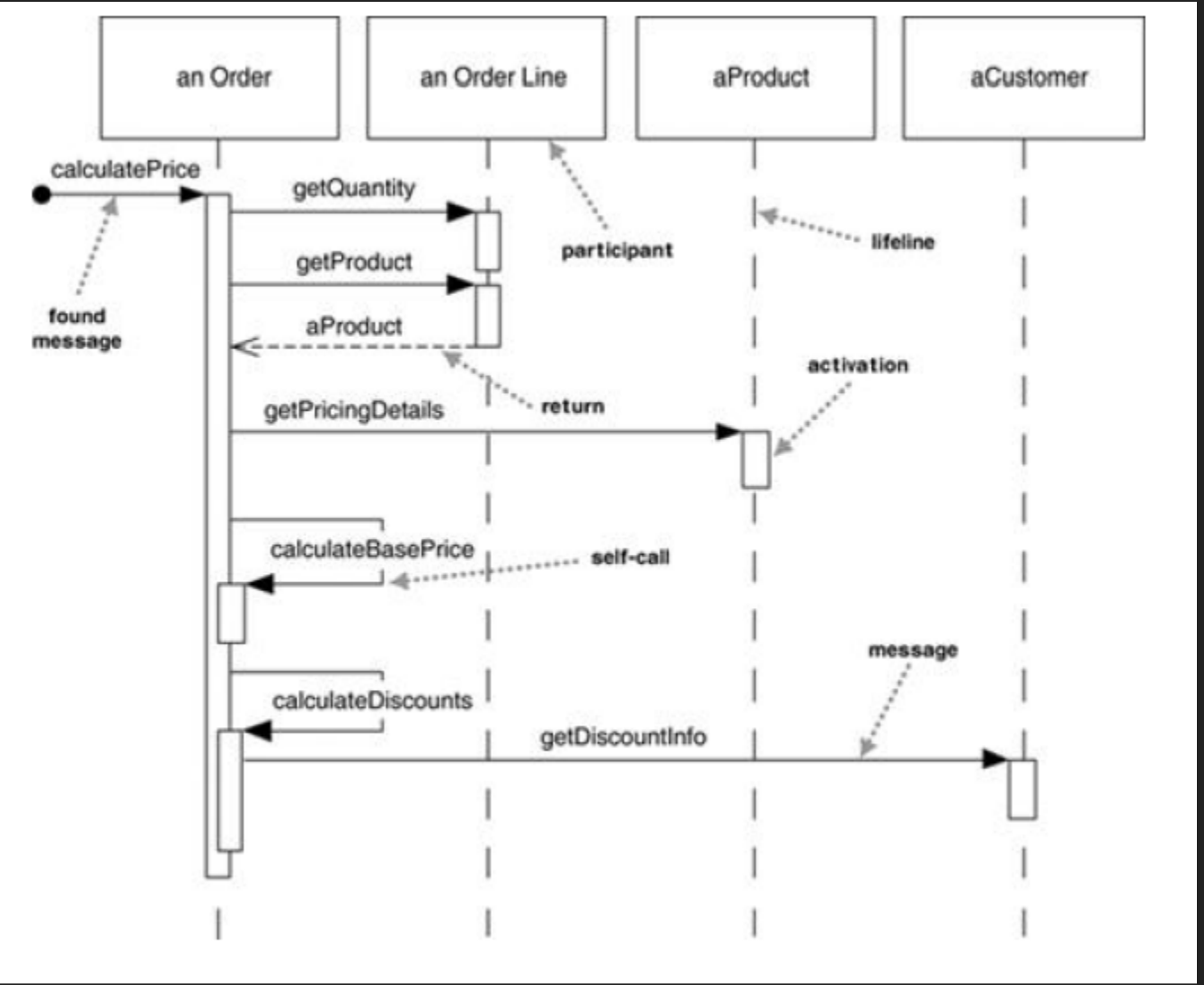
WHAT IS AN SEQUENCE DIAGRAM?

- Most common form of documenting interactions in UML
- Typically captures the behaviour of a single scenario/usecase
- Shows participating objects and messages passed between them

CONSIDER A USECASE

- We have an order and need to calculate the total price
  - Look at all the line-items in the order
  - Look up the price of each line-item
  - Calculate the sum
  - Calculate discount, if applicable

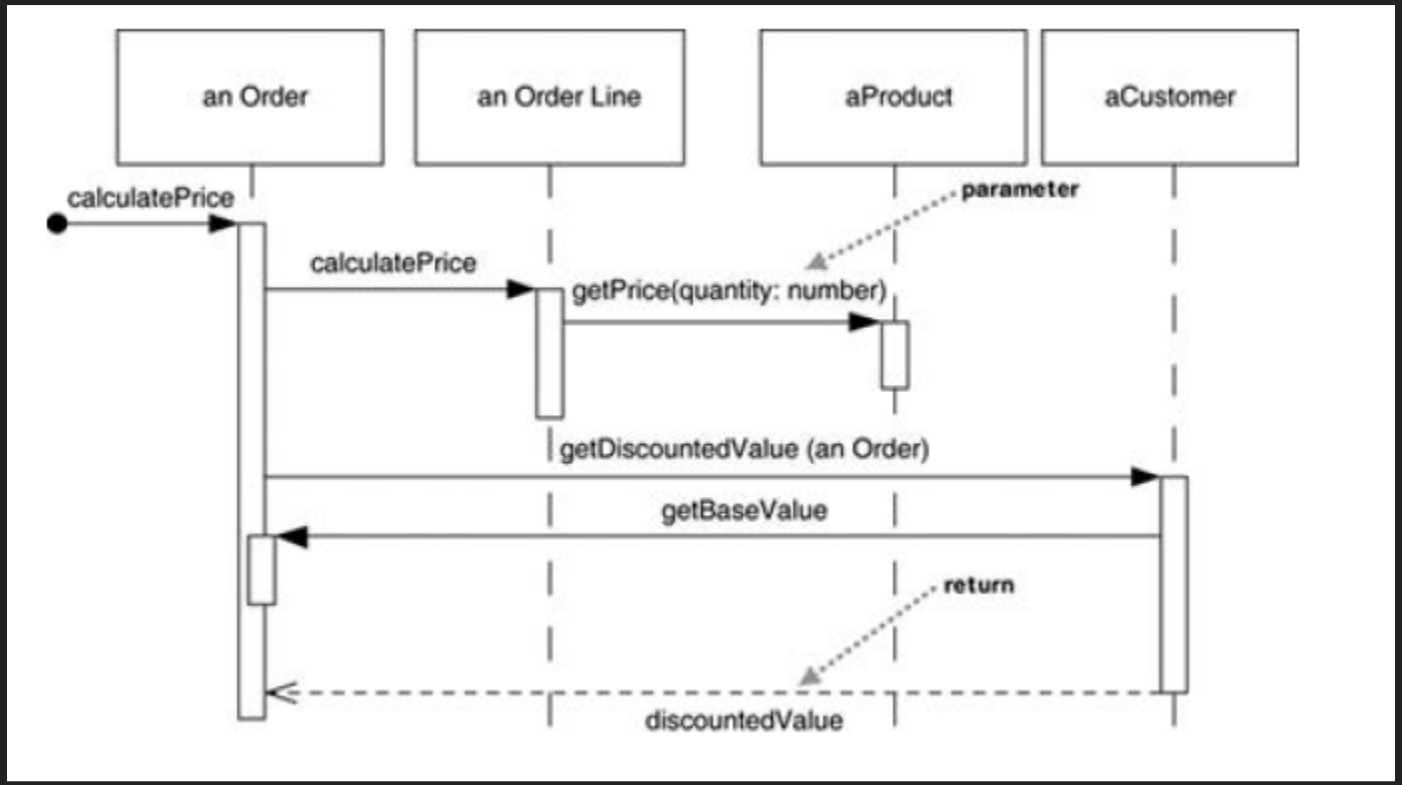
A POSSIBLE SEQUENCE DIAGRAM



THINGS TO NOTE

- The full syntax is - `name: class`, but clarity is most important
- Each lifeline has an activation bar
- Return arrows are optional for methods
- The first message has no participant, hence it is called a *found message*

A SLIGHTLY DIFFERENT APPROACH

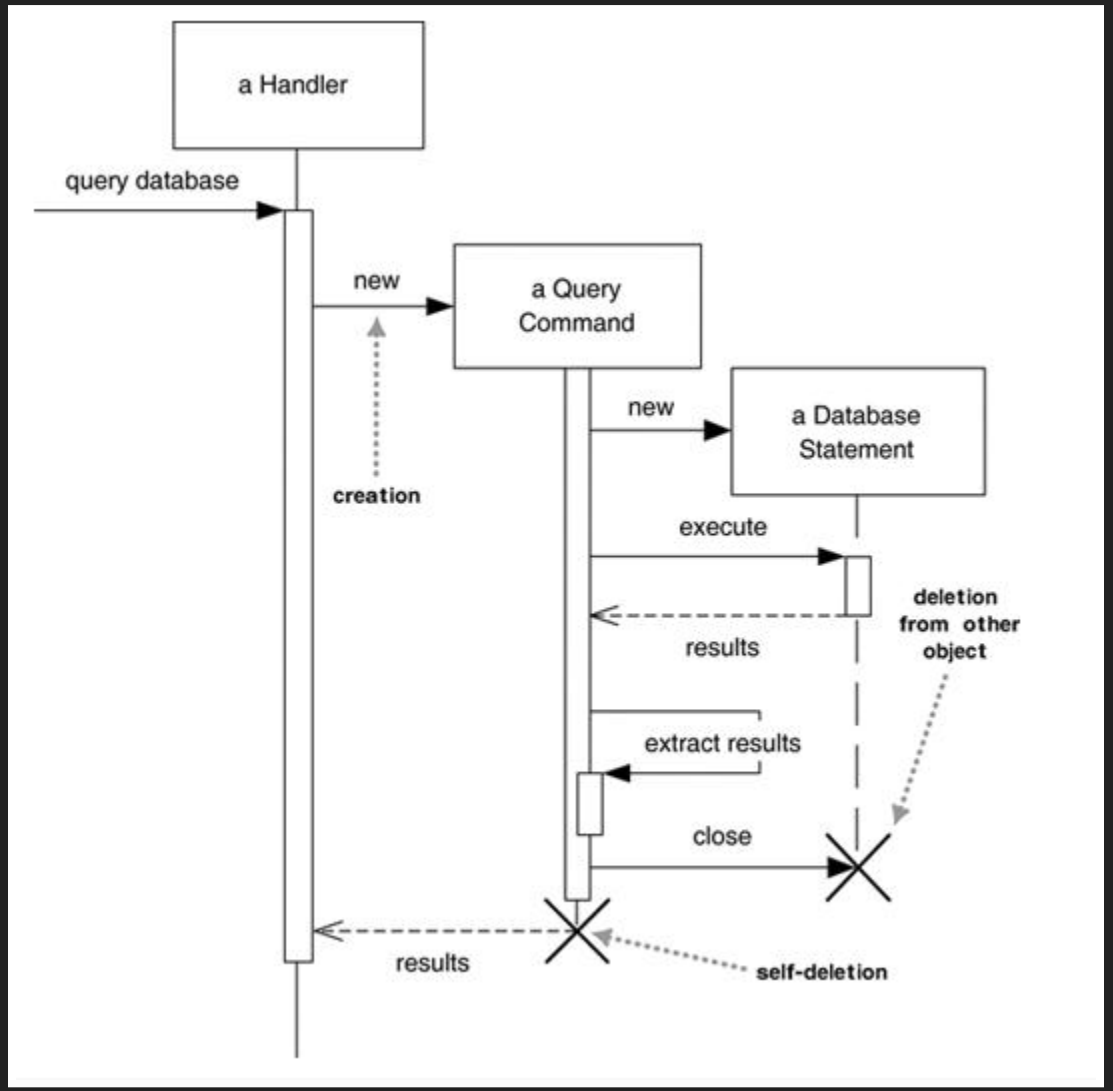


What's the difference between this and the previous sequence diagram?

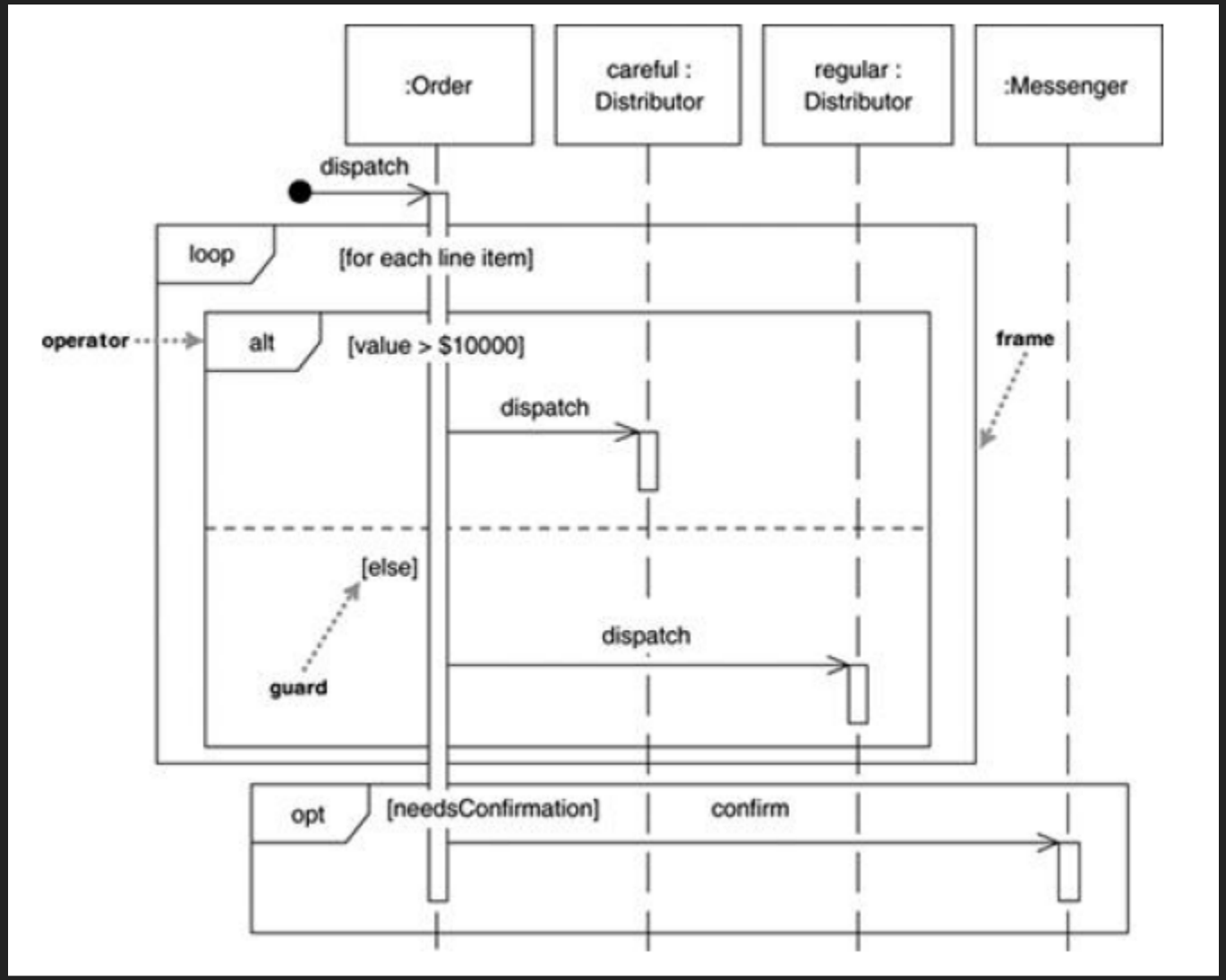
LOCUS OF CONTROL

- Note how clearly the difference between centralized and de-centralized control is shown
- Multiple objects with many little methods is considered better than one object with a huge method
- In general, more opportunities for de-coupled code and more polymorphism

CREATION AND DELETION OF PARTICIPANTS



LOOPS AND CONDITIONALS



MINUTIAE

- No mechanism to show data being passed.
- Hack! Use parameter names and return arrows
- Synchronous calls are shown using filled arrowheads
- Asynchronous calls are shown using stick arrowheads

USAGE OF SEQUENCE DIAGRAMS

Sequence diagrams are good at showing interactions between multiple classes for a *single* usecase

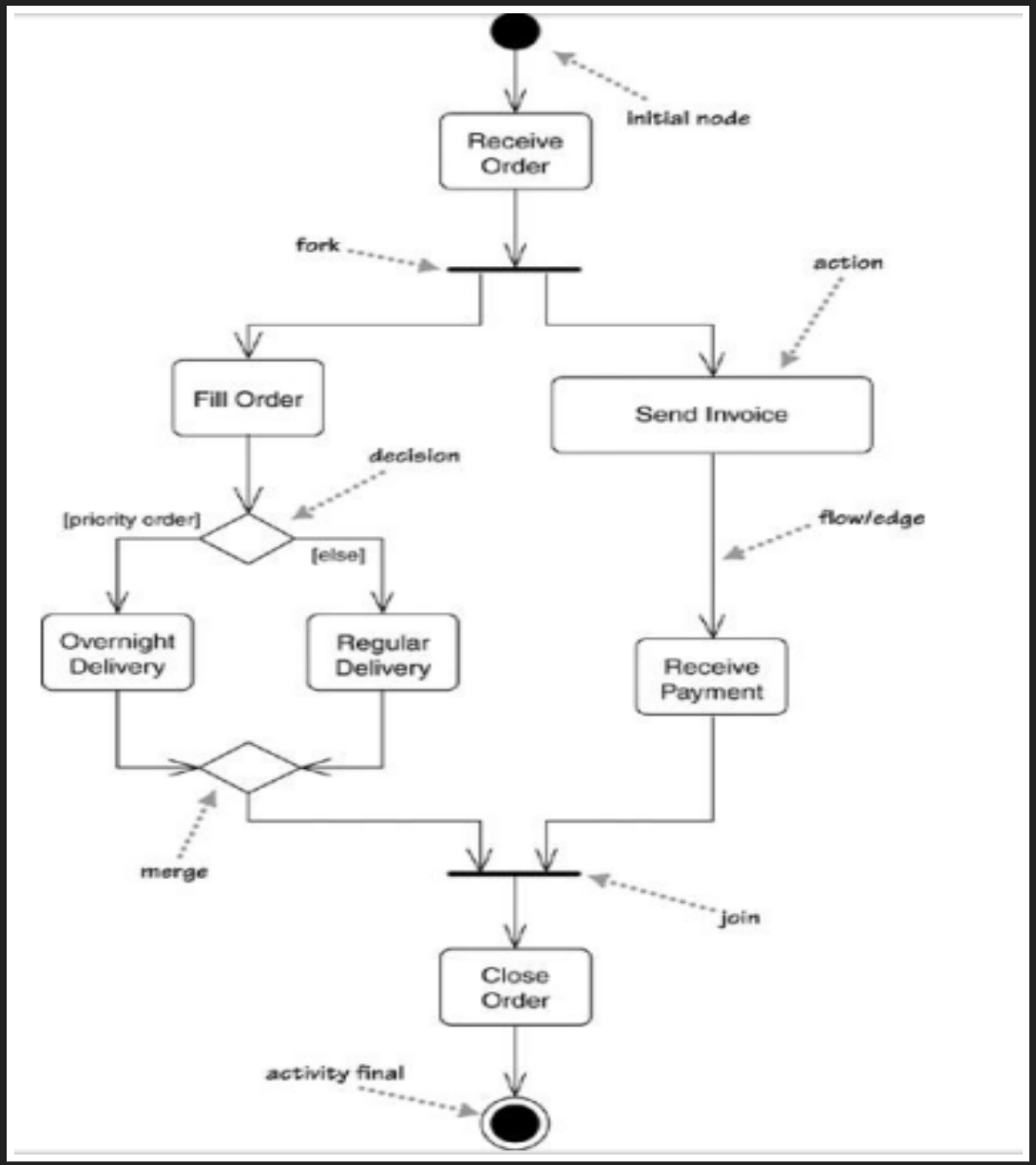
They're not very good at showing precise algorithms

ACTIVITY DIAGRAMS

Showing interactions across *multiple* usecases

A technique to show procedural logic or business workflow

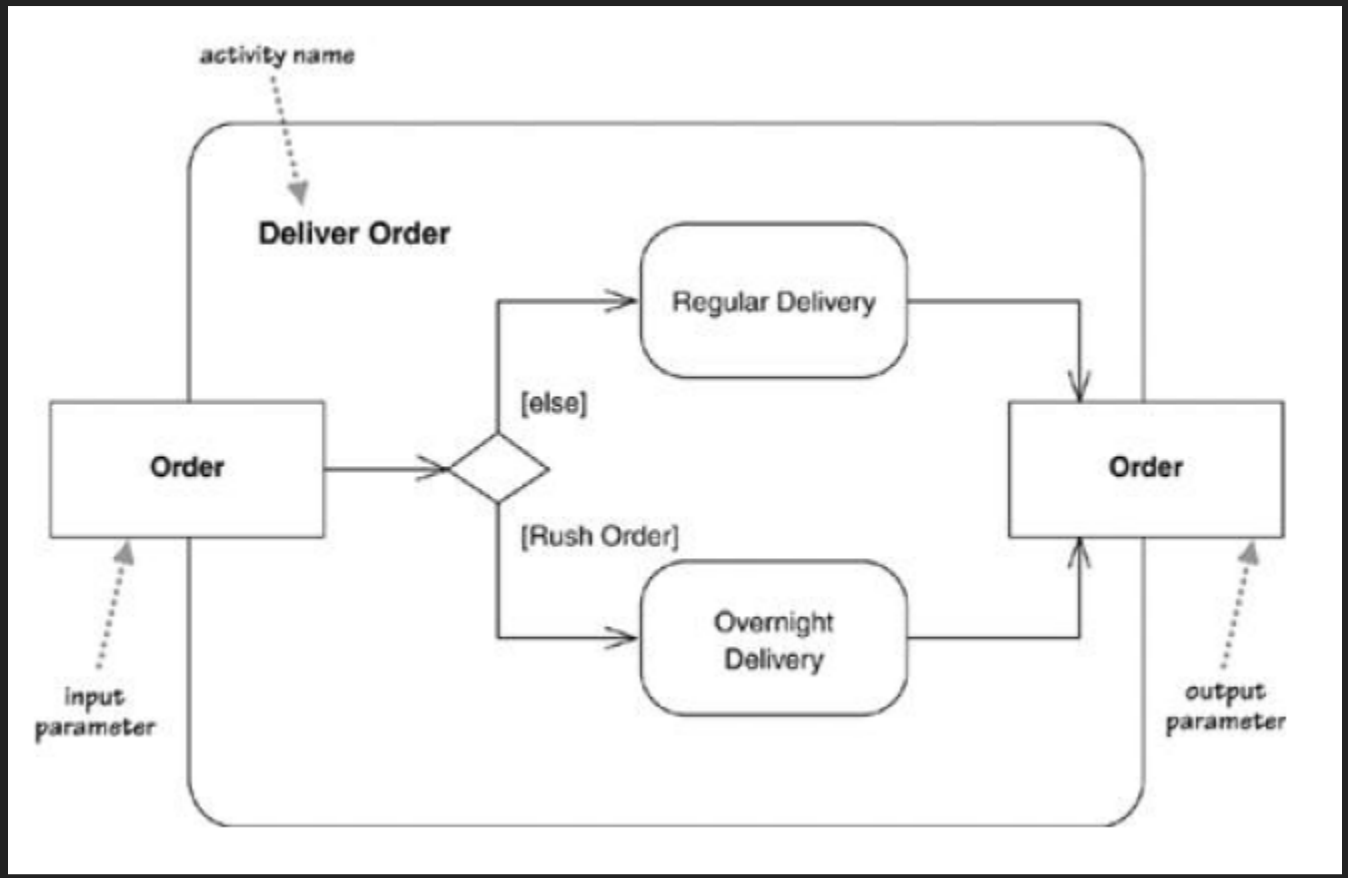
A SIMPLE ACTIVITY DIAGRAM



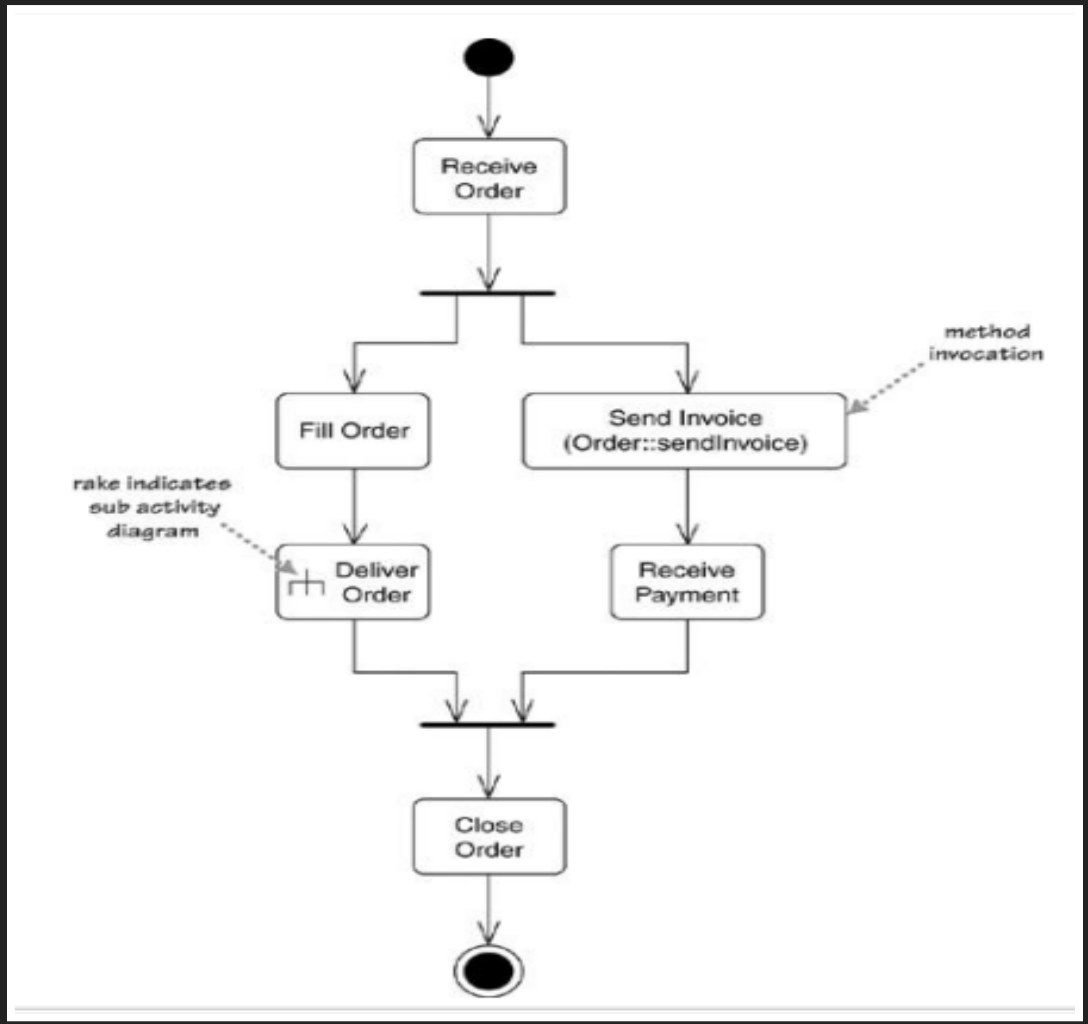
THINGS TO NOTE

- Synchronous processing needs a *join*
- Conditionals are shown through: *decisions* and *merges*
  - A *decision* has a single incoming flow and multiple outgoing flows
  - A *merge* has multiple incoming flows and single incoming flow

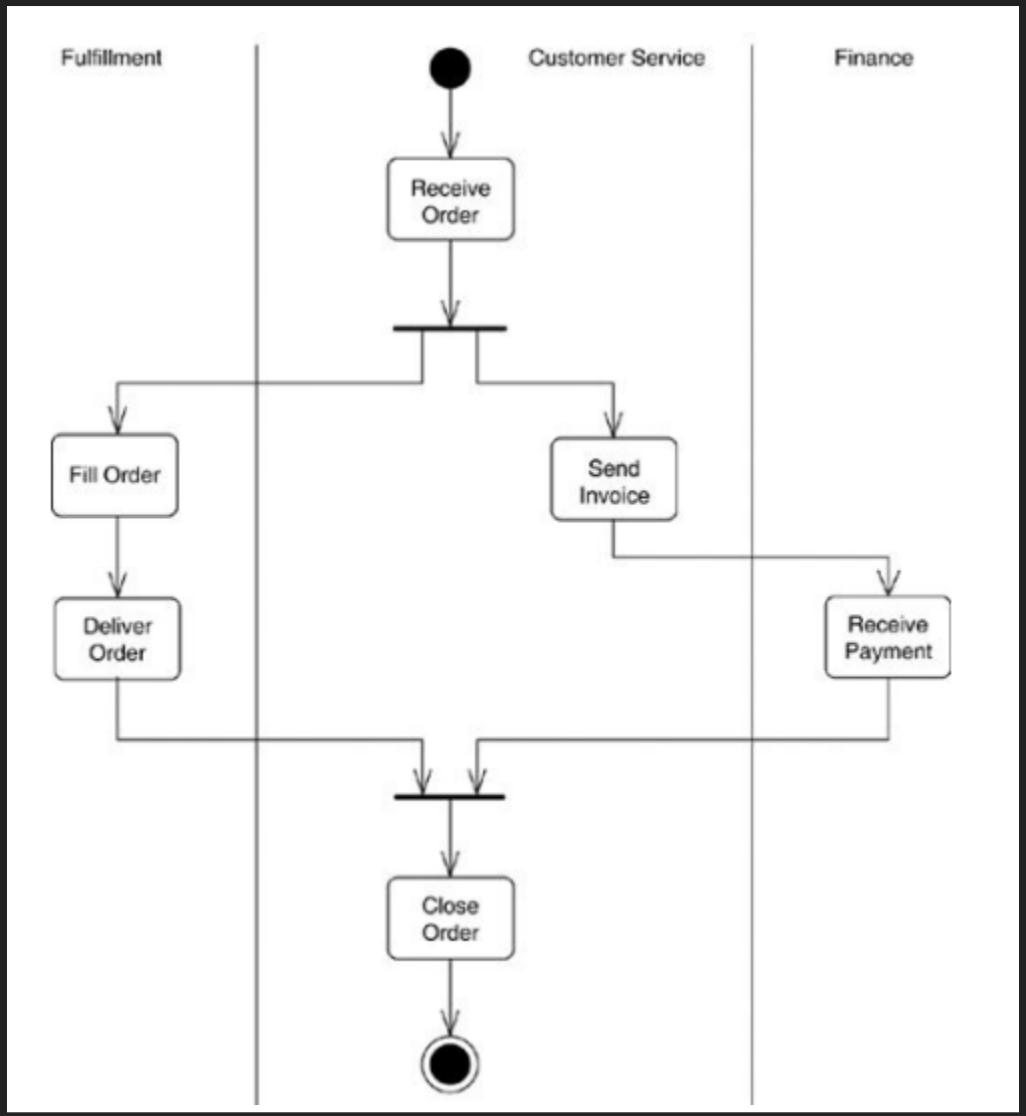
DECOMPOSING ACTIONS



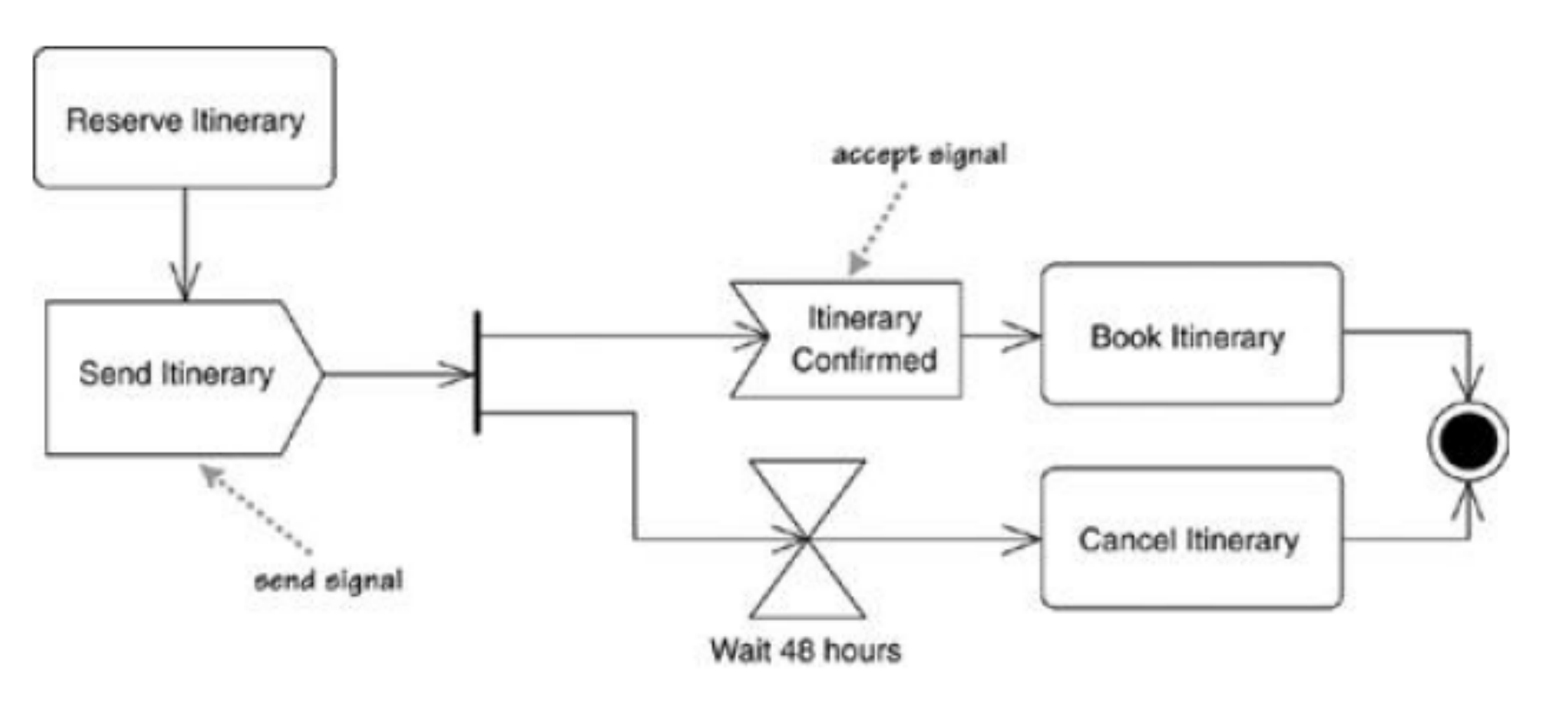
USING A SUBSIDIARY DIAGRAM



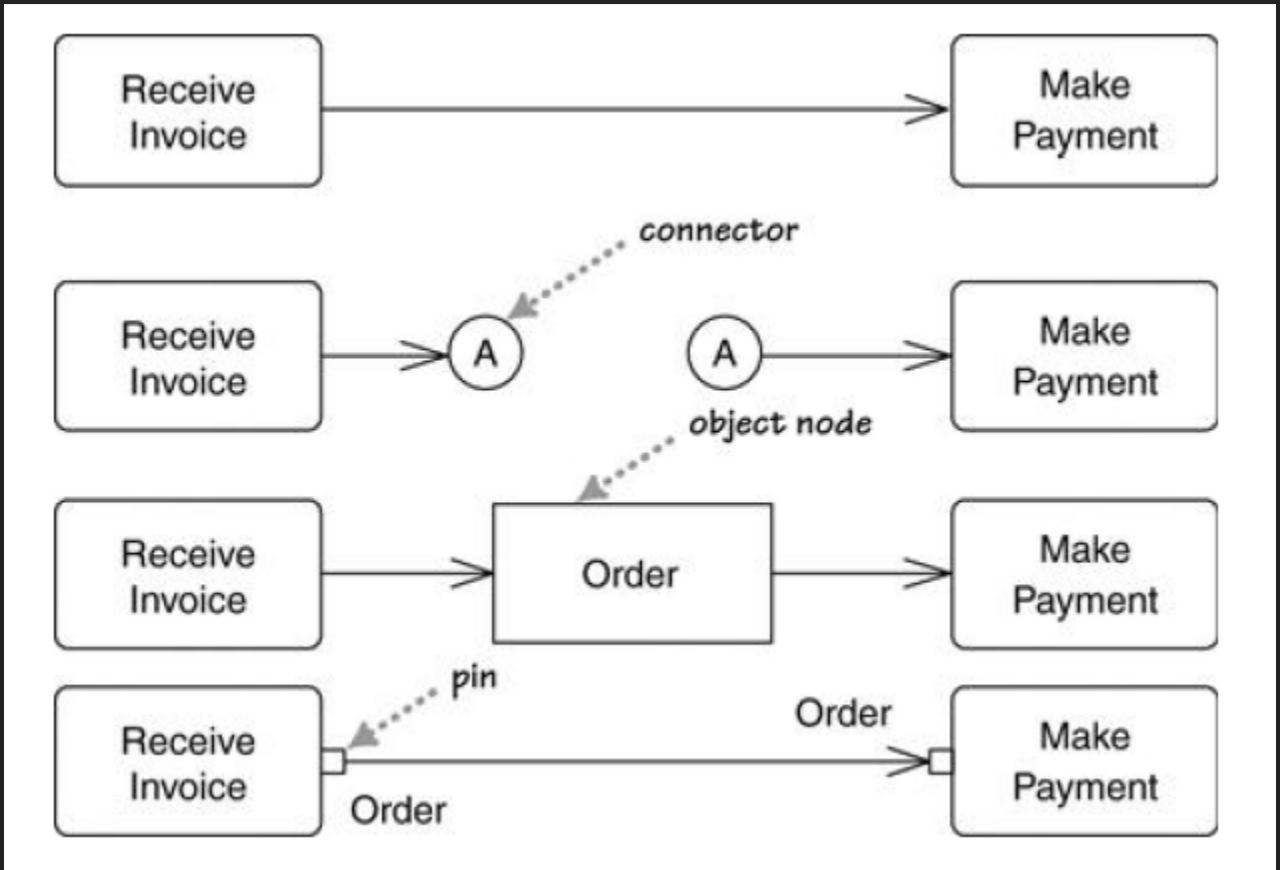
PARTITIONS


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

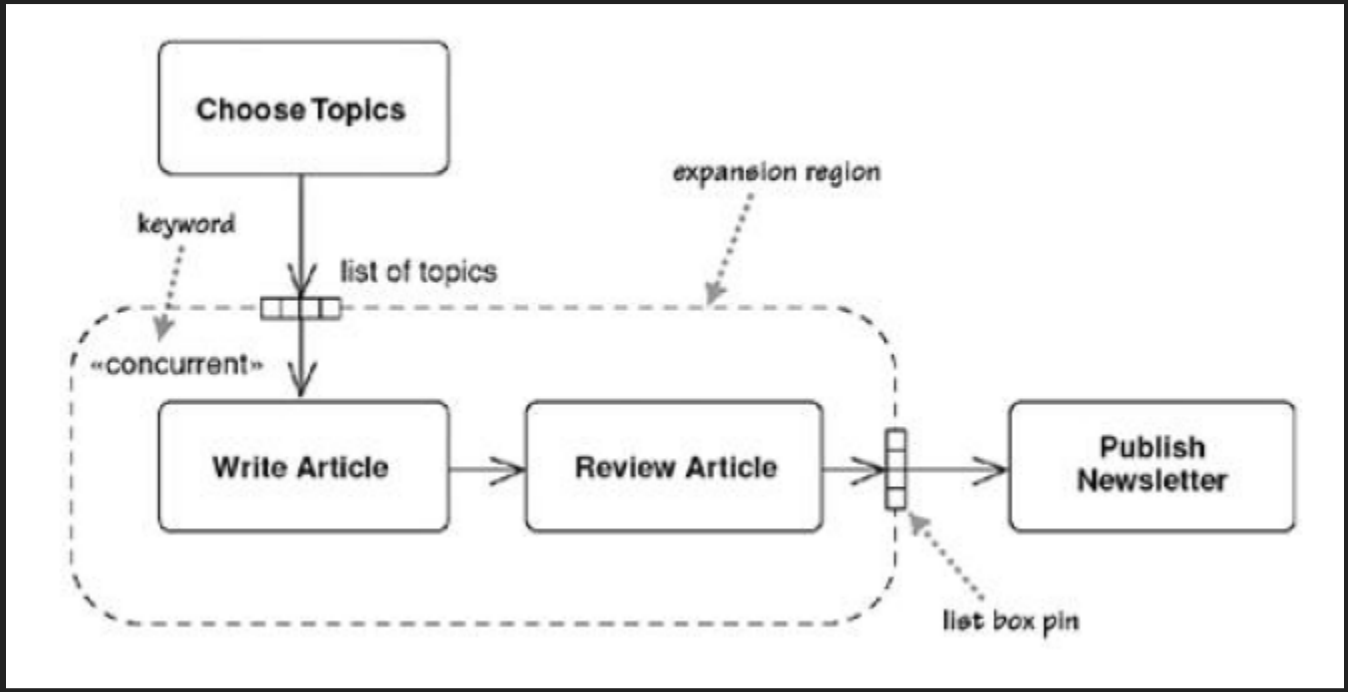
SIGNALS


Reserve Itinerary
accept signal
Send Itinerary
Itinerary Confirmed
Book Itinerary
send signal
Cancel Itinerary
Wait 48 hours

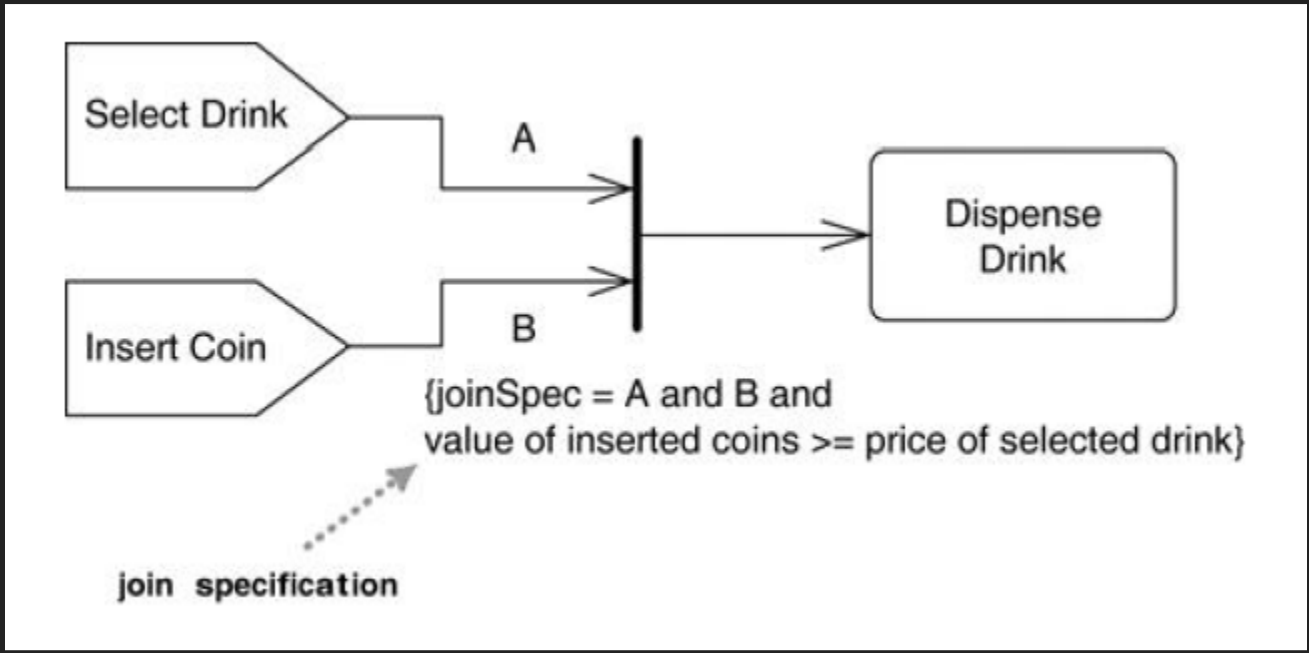
FLOWS AND EDGES



EXPANSION REGIONS



JOIN SPECIFICATION



USING ACTIVITY DIAGRAMS

- Great tool for modelling process flow and behavioural logic
- UML-compliant flowcharts

WHAT'S THE DIFFERENCE?

- Activity diagrams show a workflow
  - starting point
  - decisions, actions, splits, joins
  - termination conditions
- Sequence diagrams show interactions:
  - Between actors and objects
  - Between objects and methods
  - Parameters, return types, creation/deletion of objects

BEFORE WE GO...

Assignment announcements

- New EngineersLog.csv in your directory ("2" for this week)
- Undergraduate programming centre - revamped/fresh/new-and-improved https://www.scss.tcd.ie/misc/psc/

THAT'S ALL, FOLKS!

Questions? Comments?