



# **more uml: sequence & use case diagrams**



- as a sketch:
  - very selective – informal and dynamic
  - forward engineering: describe some concept you need to implement
  - reverse engineering: explain how some part of a program works



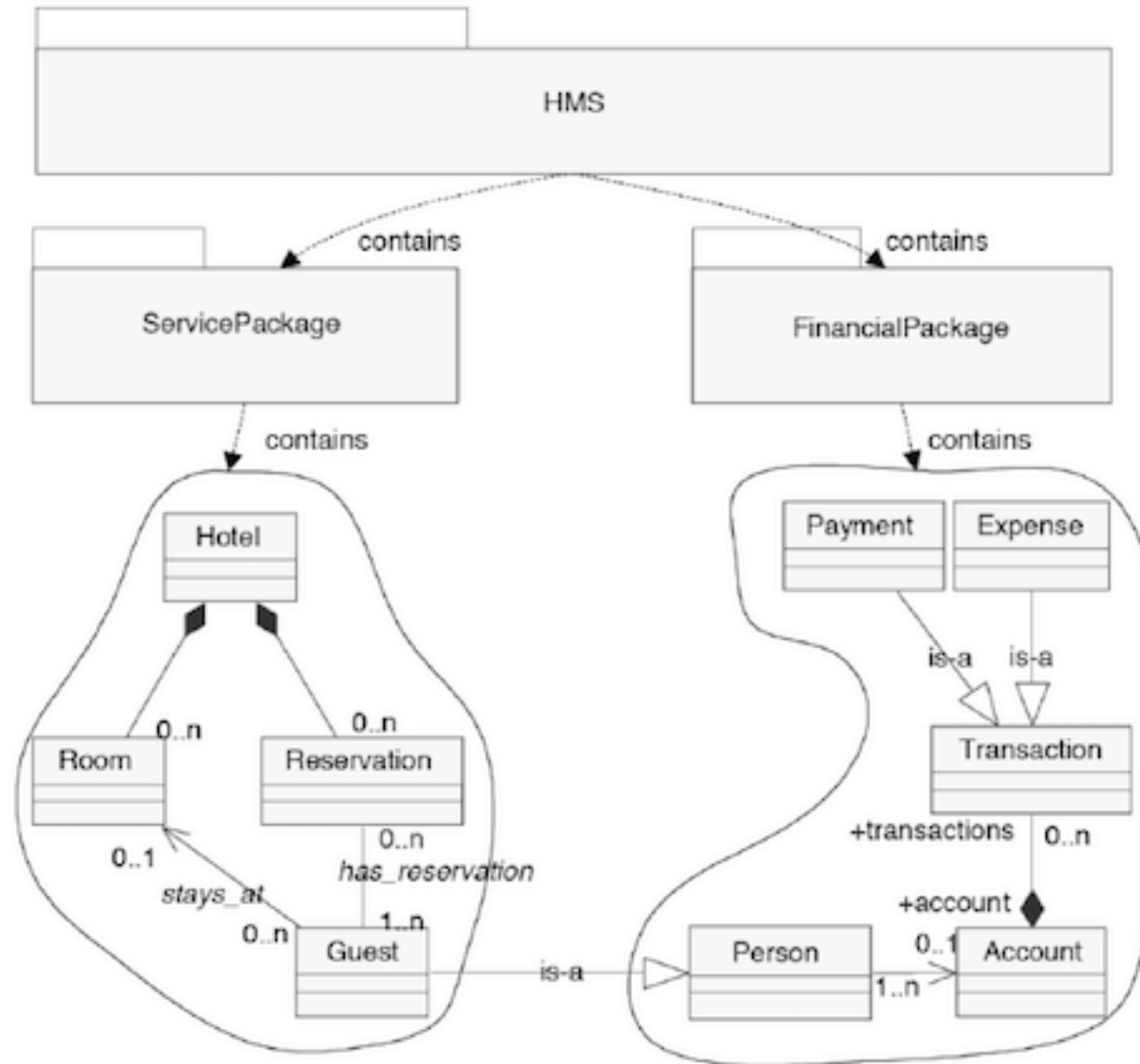
- as a blueprint:
  - emphasis on completeness
  - forward engineering: model as a detailed spec for a programmer
  - reverse engineering: model as a code browser
  - round-trip: tools provide both forward & reverse engineering to move back and forth between design and code



- as a programming language:
  - uml models are automatically compiled into working code
    - executable uml:  
[https://en.wikipedia.org/wiki/Executable\\_UML](https://en.wikipedia.org/wiki/Executable_UML)
  - requires sophisticated tools
  - the model ***is*** the code



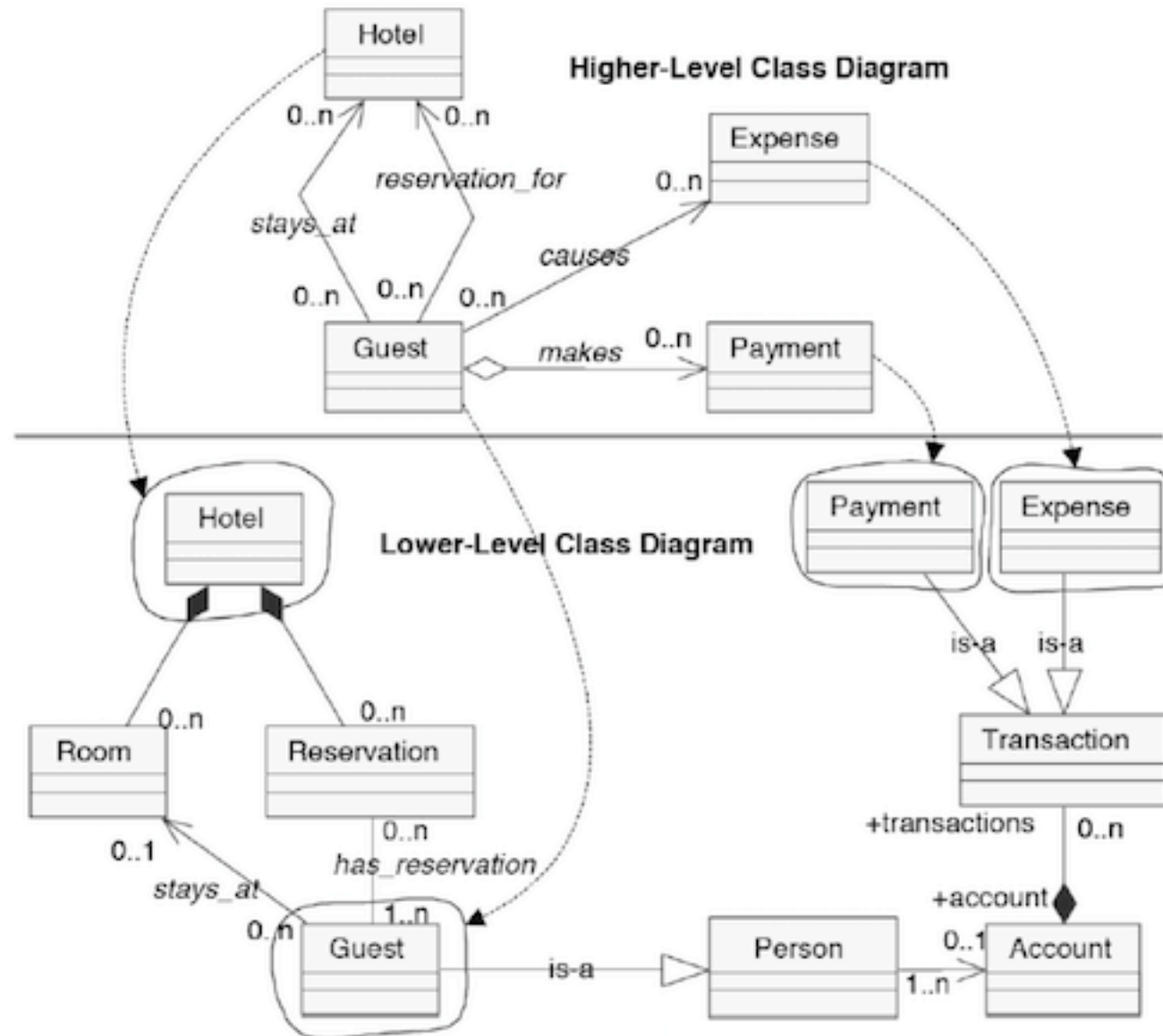
# package decomposition



Source: from Egyed "Automated Abstraction of Class Diagrams, TSE 2002

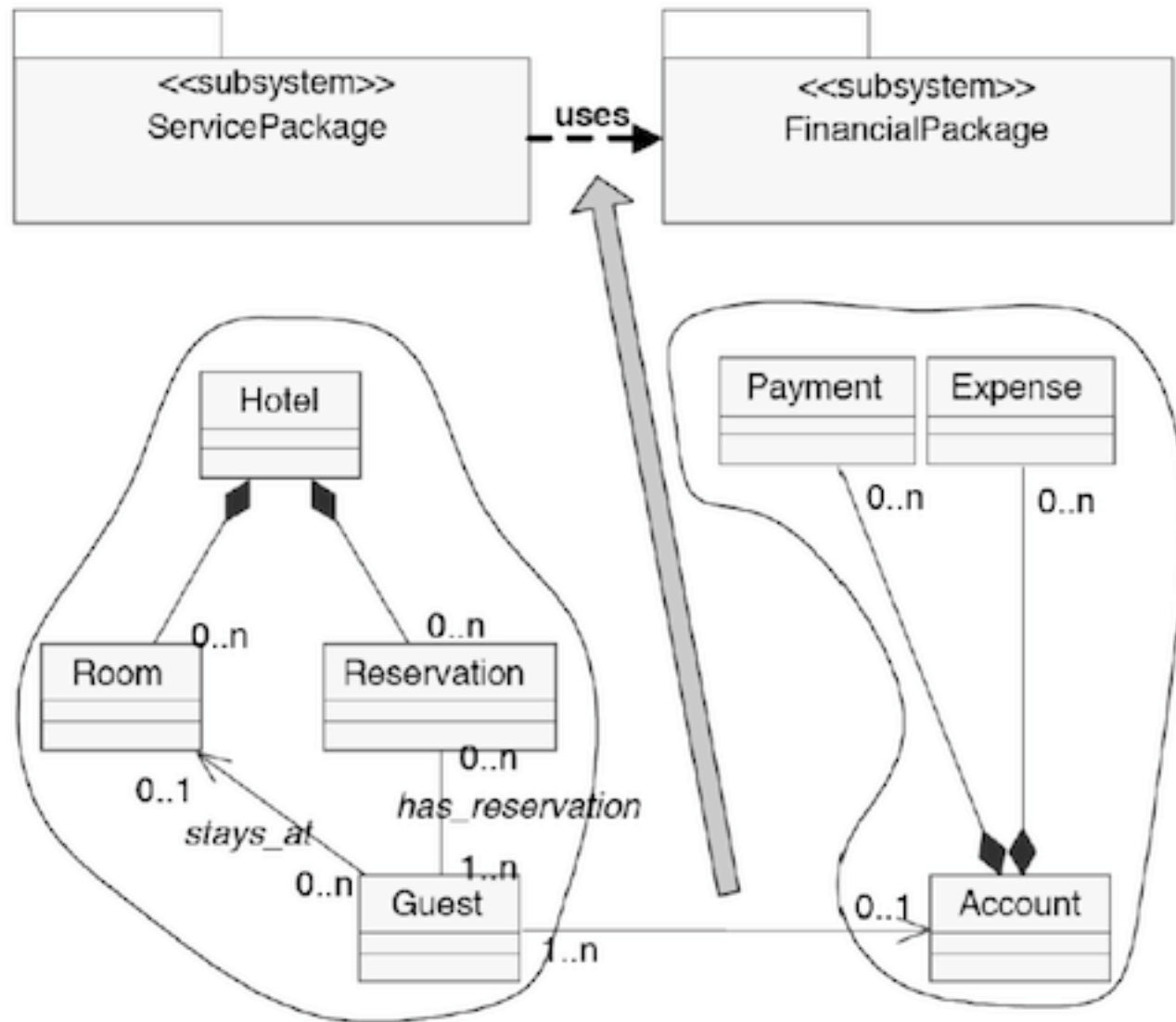


# class abstraction





# *finding dependencies*





## *things to model*

- structure of the code
  - code dependencies
  - components and couplings
- behaviour of the code
  - execution traces
  - state machine models of complex objects
- function of the code
  - what function(s) does it provide to the user?



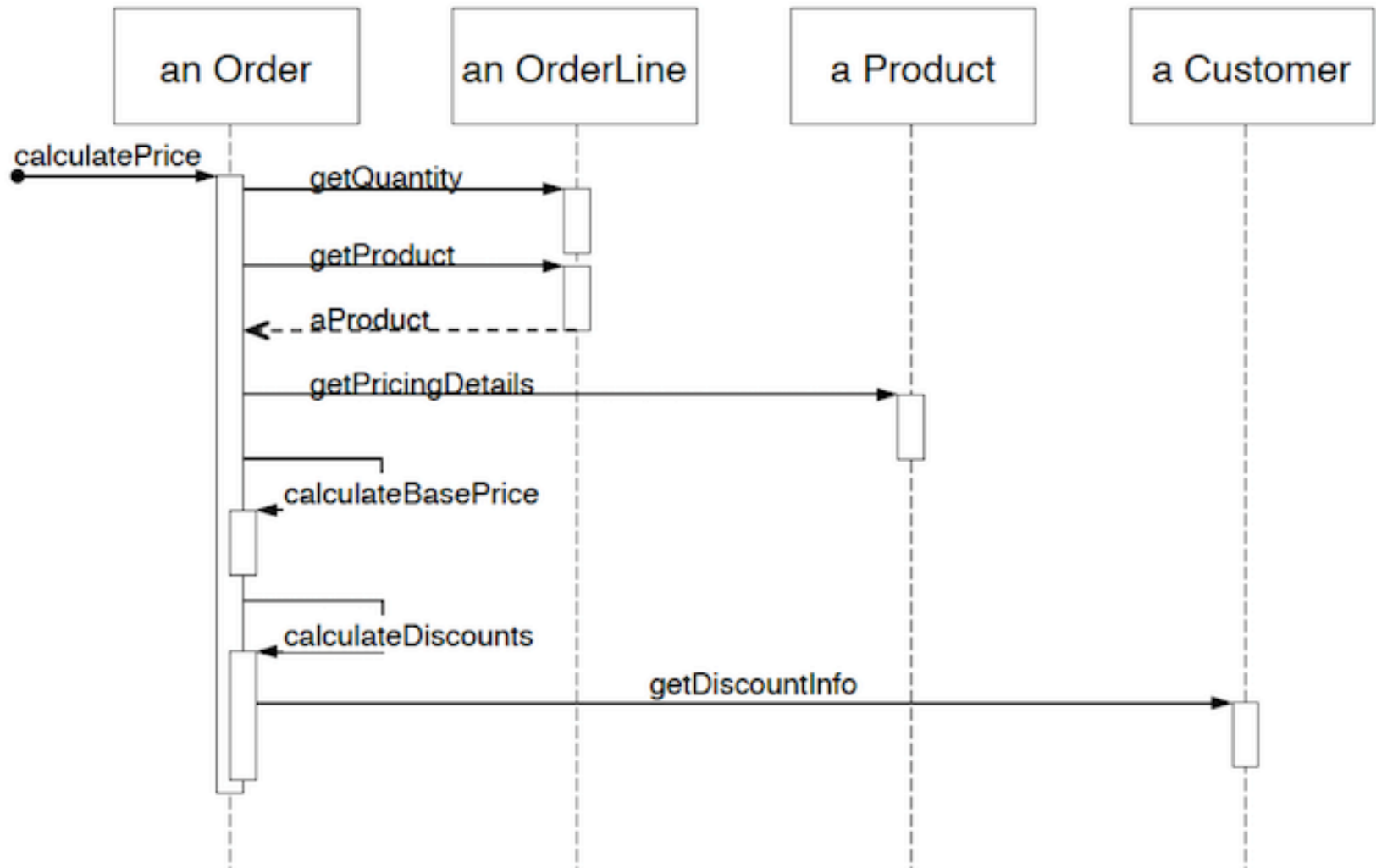


## *things to model*

- structure of the code
  - code dependencies
  - components and couplings
- behaviour of the code
  - execution traces
  - state machine models of complex objects
- function of the code
  - what function(s) does it provide to the user?

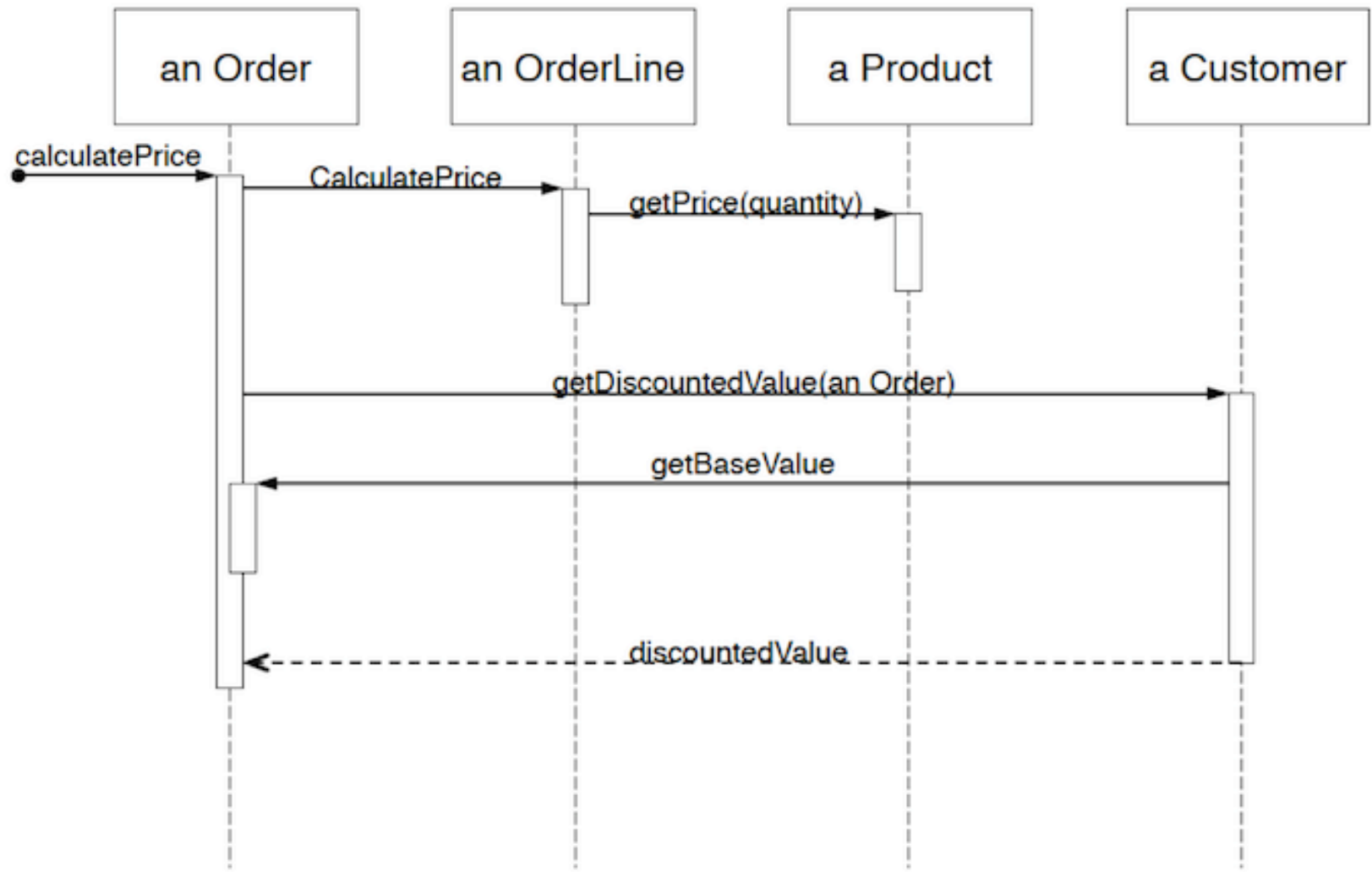


# *sequence diagrams*



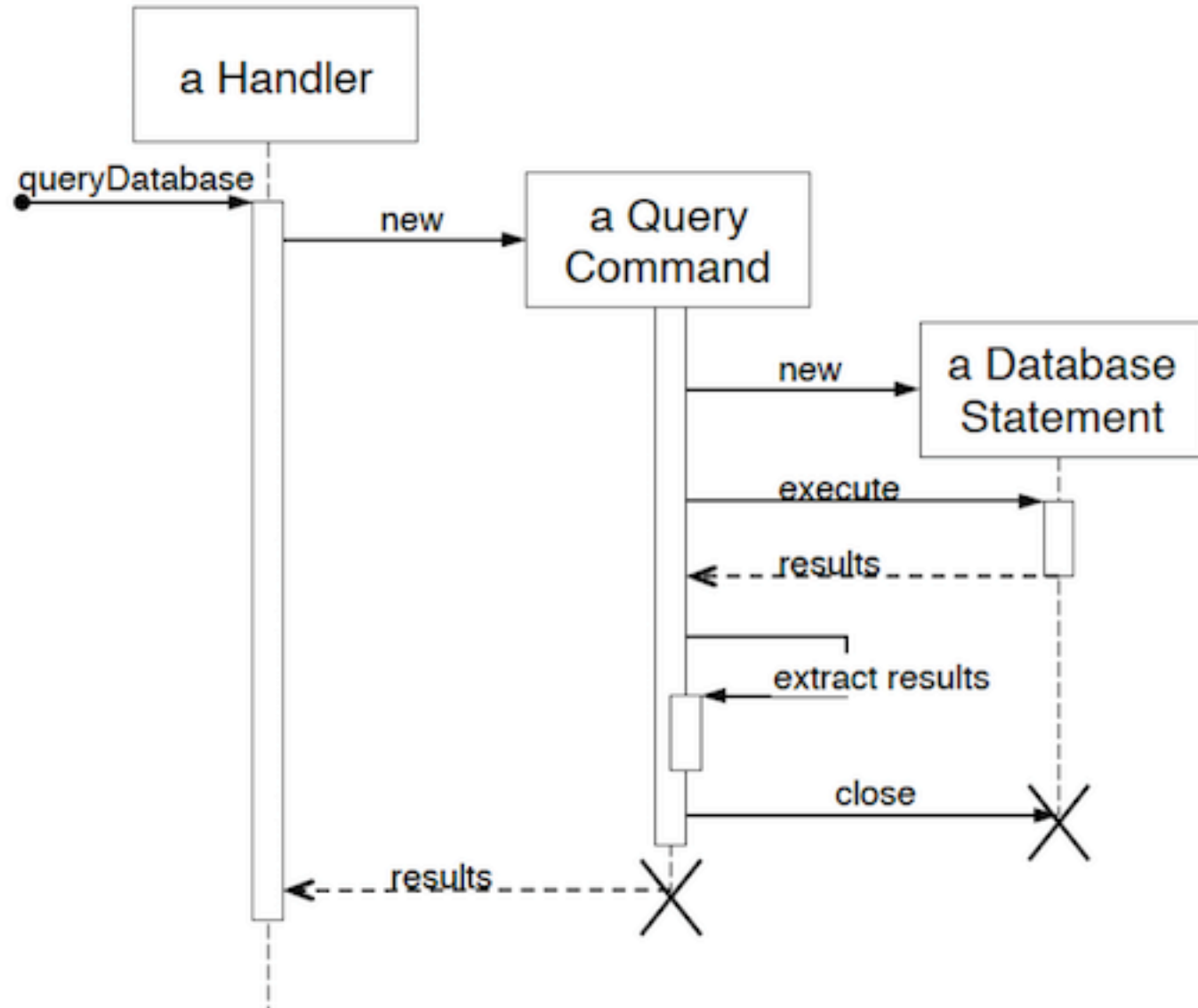


# *design choices*



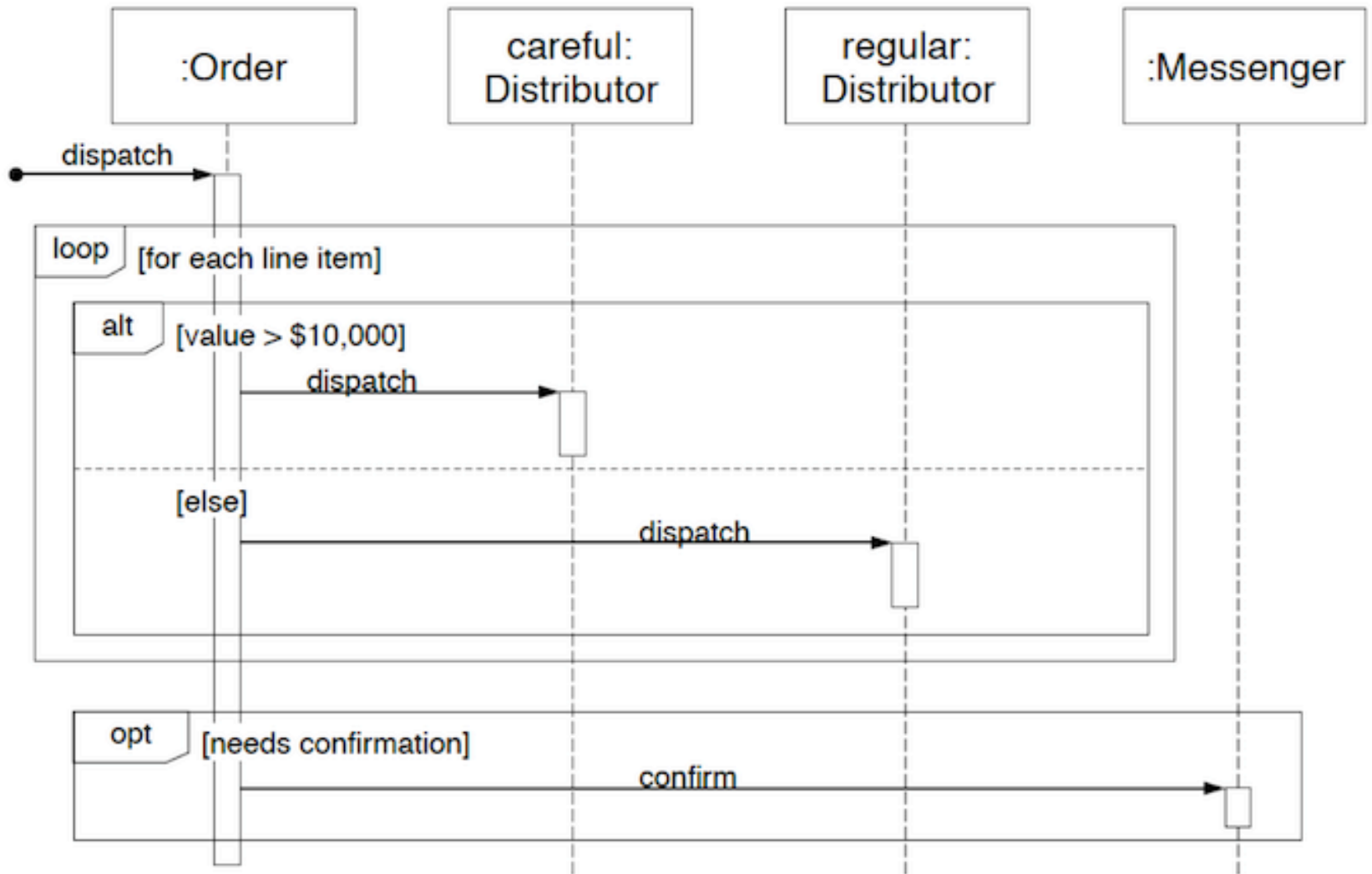


# *object creation and deletion*





# *interaction frames*





## *interaction frames (2)*

Operator	Meaning
alt	Alternative; only the frame whose guard is true will execute
opt	Optional; only executes if the guard is true
par	Parallel; frames execute in parallel
loop	Frame executes multiple times, guard indicates how many
region	Critical region; only one thread can execute this frame at a time
neg	Negative; frame shows an invalid interaction
ref	Reference; refers to a sequence shown on another diagram
sd	Sequence Diagram; used to surround the whole diagram (optional)



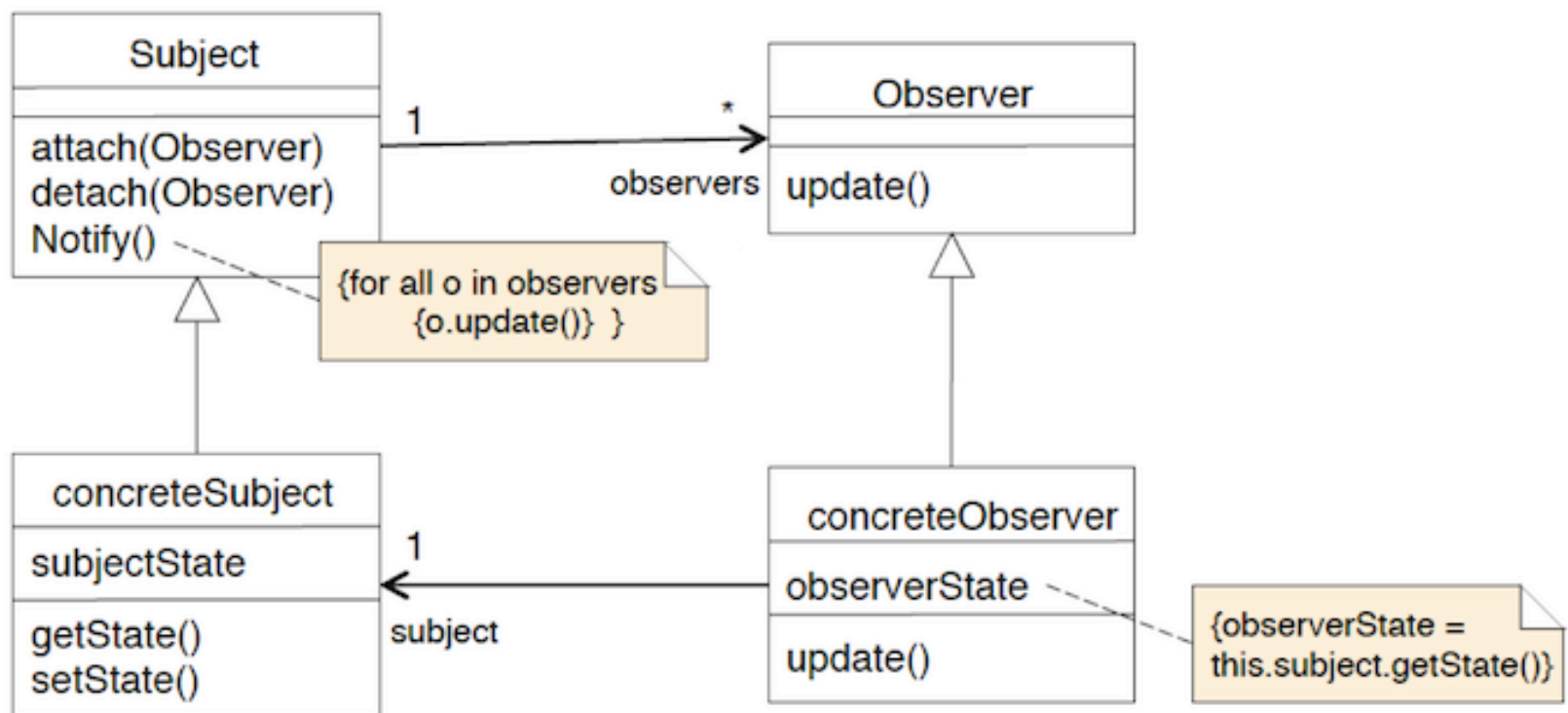
## *when to use sequence diagrams*

- **comparing design options**
  - shows how objects collaborate to carry out a task
  - graphical form shows alternative behaviours
- **assessing bottlenecks**
  - ex. an object through which many messages pass
- **explaining design patterns**
  - mostly an academic exercise (my opinion)
- **elaborating use cases**
  - shows how user expects to interact with system
  - shows how user interface operates



## *modeling a design pattern*

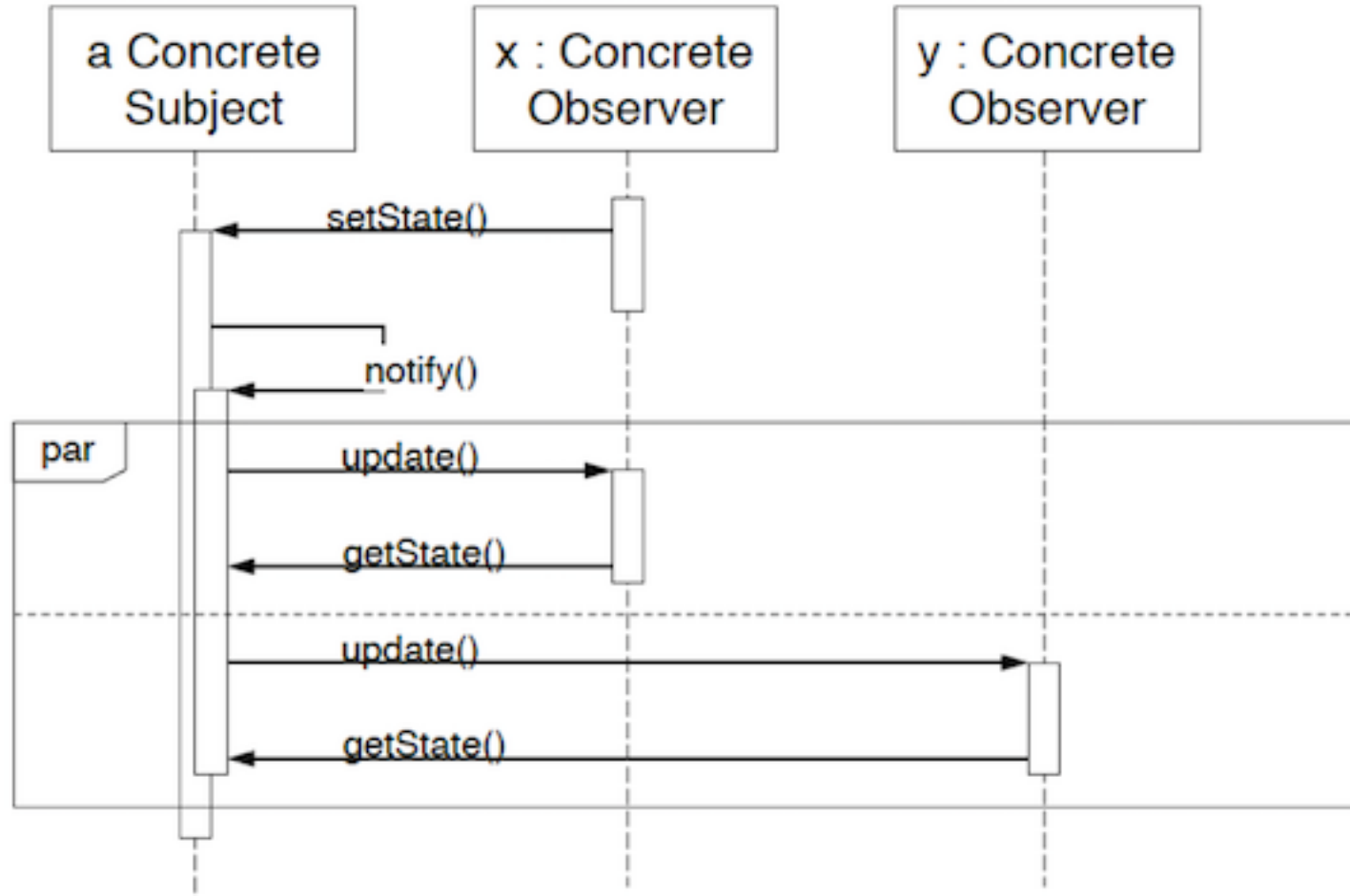
- ex. observer pattern
  - one-to-many dependency, maintaining consistency
  - subject pushes updates to observers







# *observer pattern sequence diagram*





## *sequence diag. – style points*

- spatial layout
  - left-to-right ordering of “messages”
  - proactive “actors” on left, reactive on right
- readability
  - keep as simple as possible
  - ok to omit **obvious** return values
  - ok to omit object destruction
- usage
  - focus on critical interactions (part of keep it simple)
- consistency
  - class names consistent with class diagrams
  - message routes consistent with class associations



# ***use case diagrams***

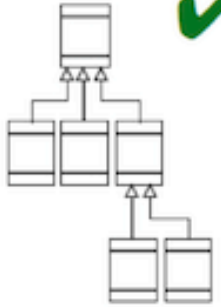
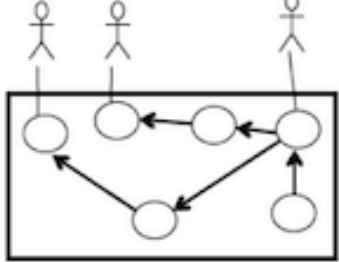
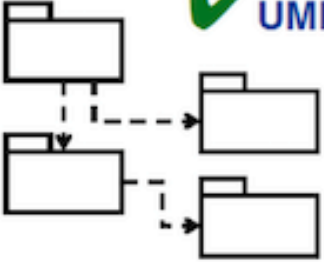
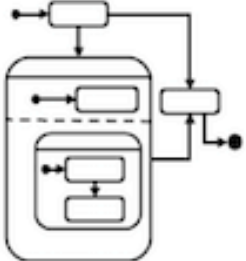
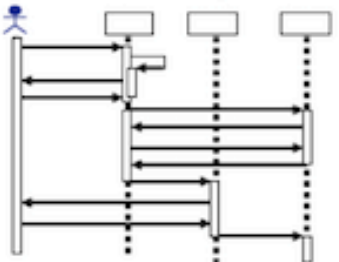
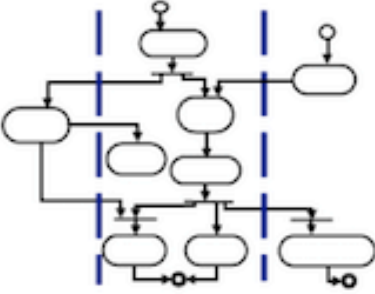


## *use case driven design*

- user stories in agile development
- introducing uml into the software process
- domain models
- use cases



# refresher – uml notations

 <p><b>UML Class Diagrams</b> information structure relationships between data items modular structure for the system</p>	 <p><b>Use Cases</b> user's view Lists functions visual overview of the main requirements</p>
 <p><b>UML Package Diagrams</b> Overall architecture Dependencies between components</p>	 <p><b>(UML) Statecharts</b> responses to events dynamic behavior event ordering, reachability, deadlock, etc</p>
 <p><b>UML Sequence Diagrams</b> individual scenario interactions between users and system Sequence of messages</p>	 <p><b>Activity diagrams</b> business processes; concurrency and synchronization; dependencies between tasks;</p>



# what do users want?

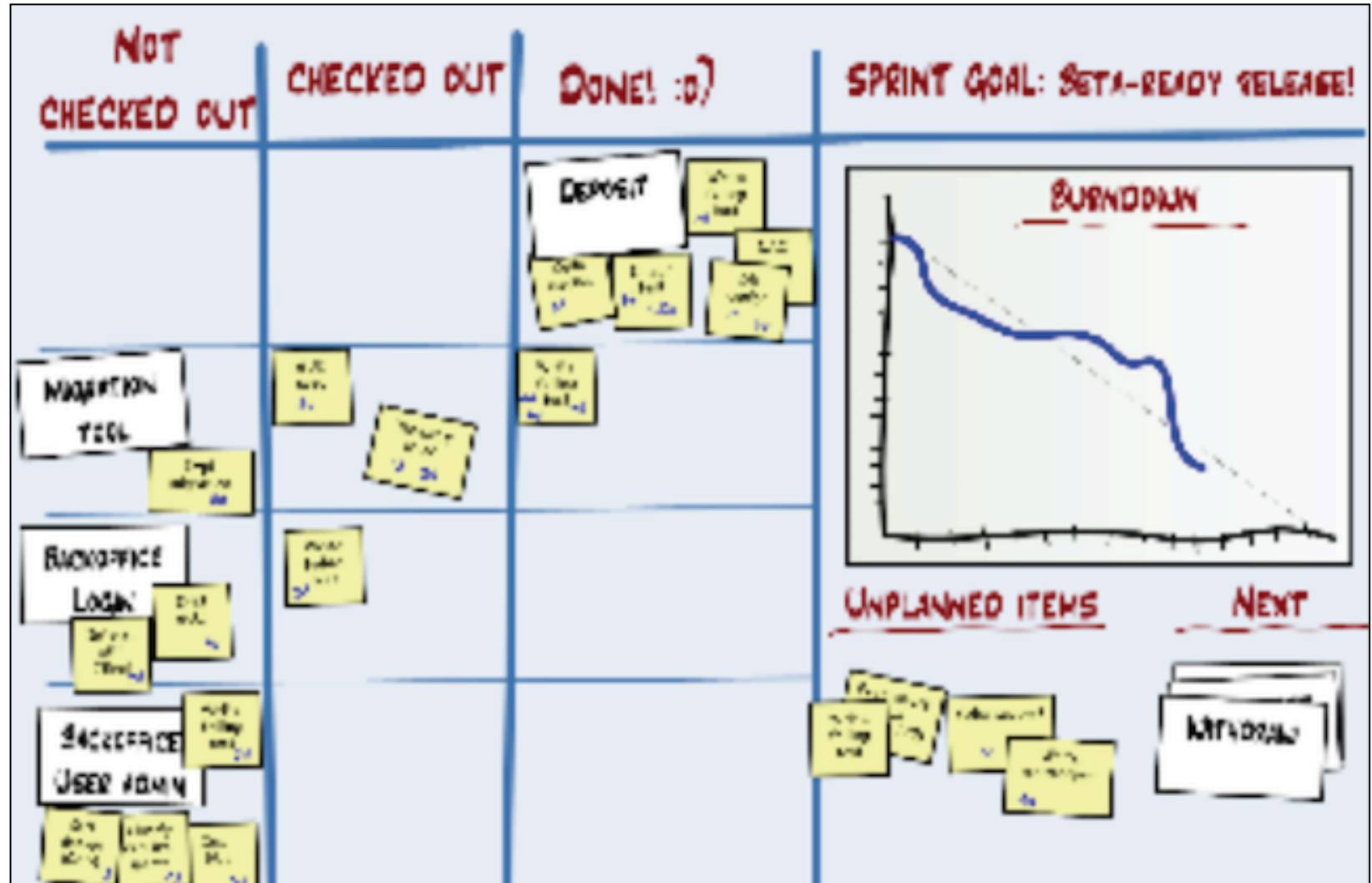
- user stories
  - used in XP, scrum, etc.
  - identify the user (role) who wants it
- (UI) storyboards
  - sketch of how a user will do a task
  - shows interactions at each step
  - used in UI design
- use cases
  - sets of user features
  - uml diagram shows inter-relationships

As a librarian, I want to be able to search for books by publication year.





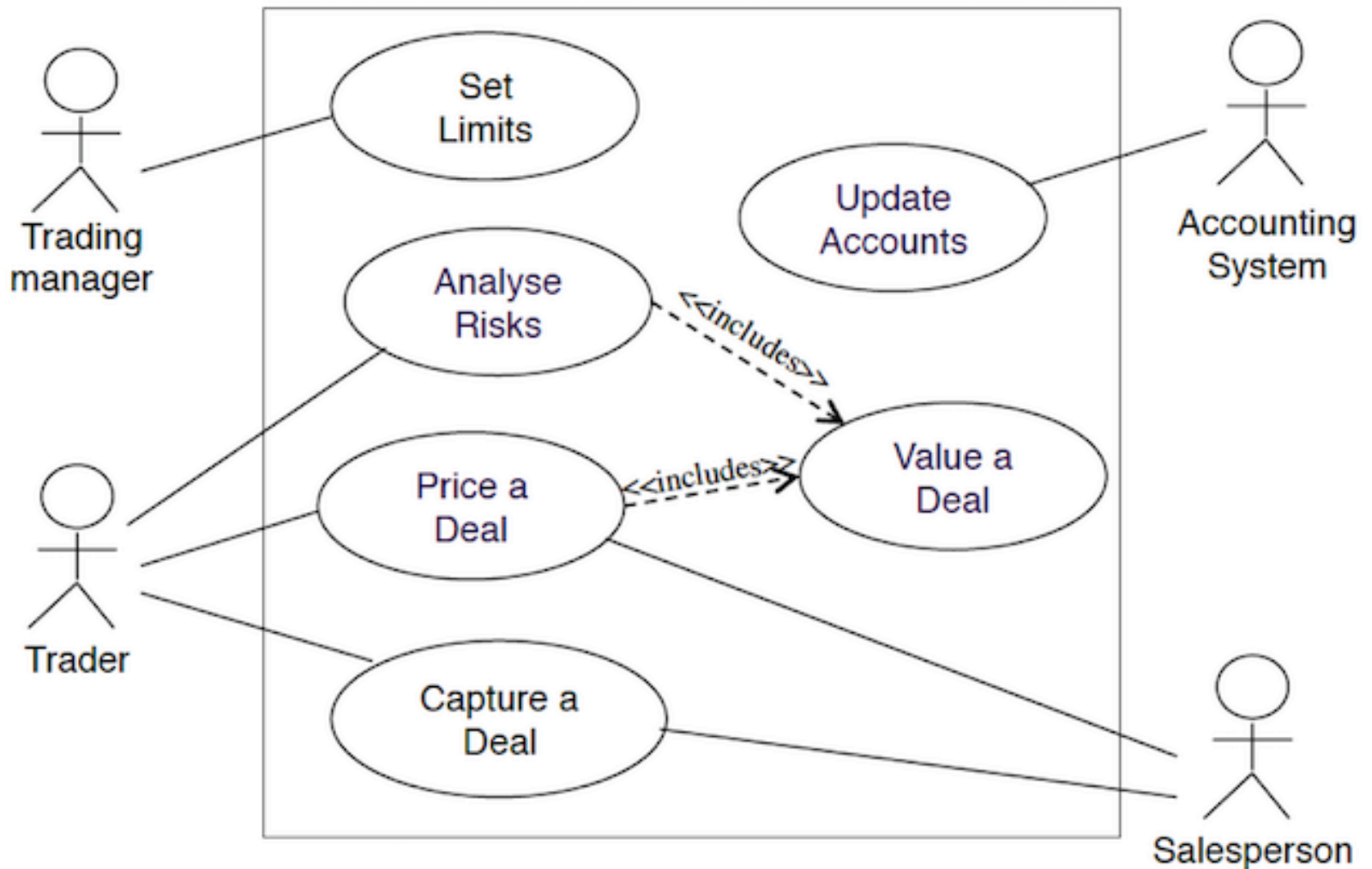
# tracking the stories







## *example use case diagram*







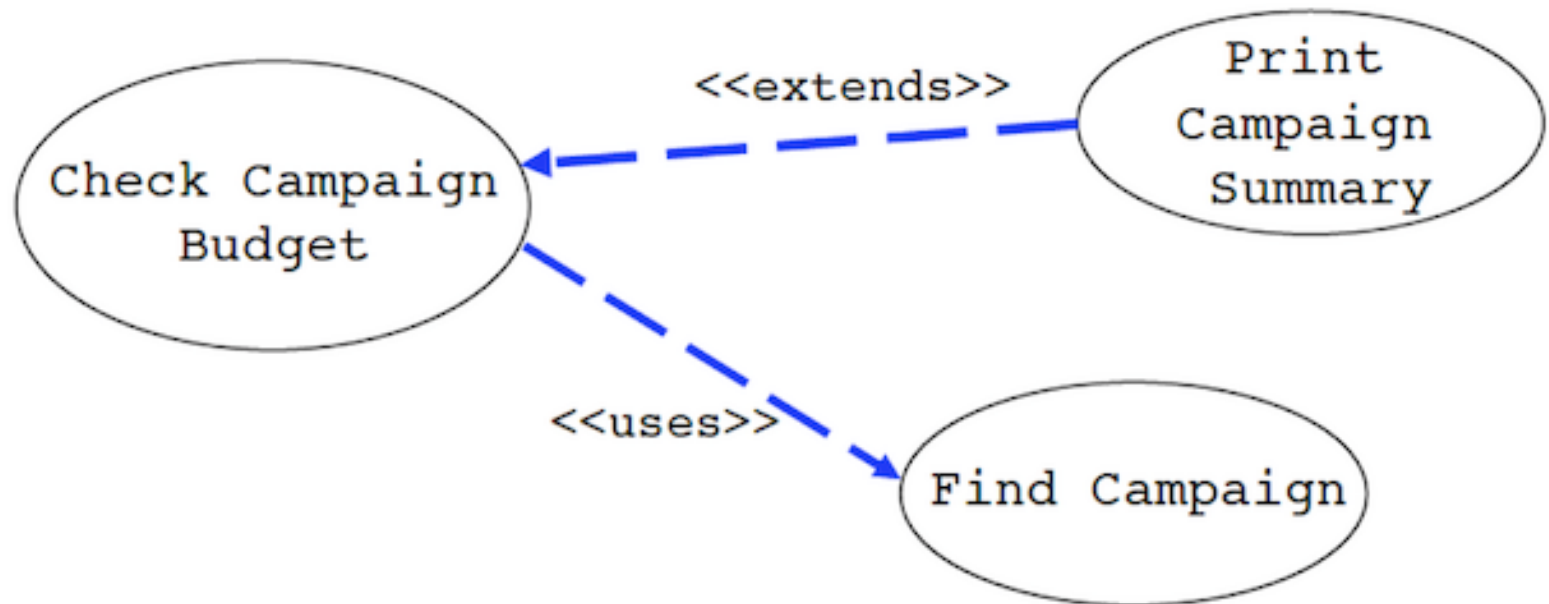
## *relationships between use cases*

<<extends>> one use case adds behaviour to a base case

- used to model a part of a use case that the user may see as optional system behaviour
- also models a separate sub-class which is executed conditionally

<<uses>> one use case invokes another

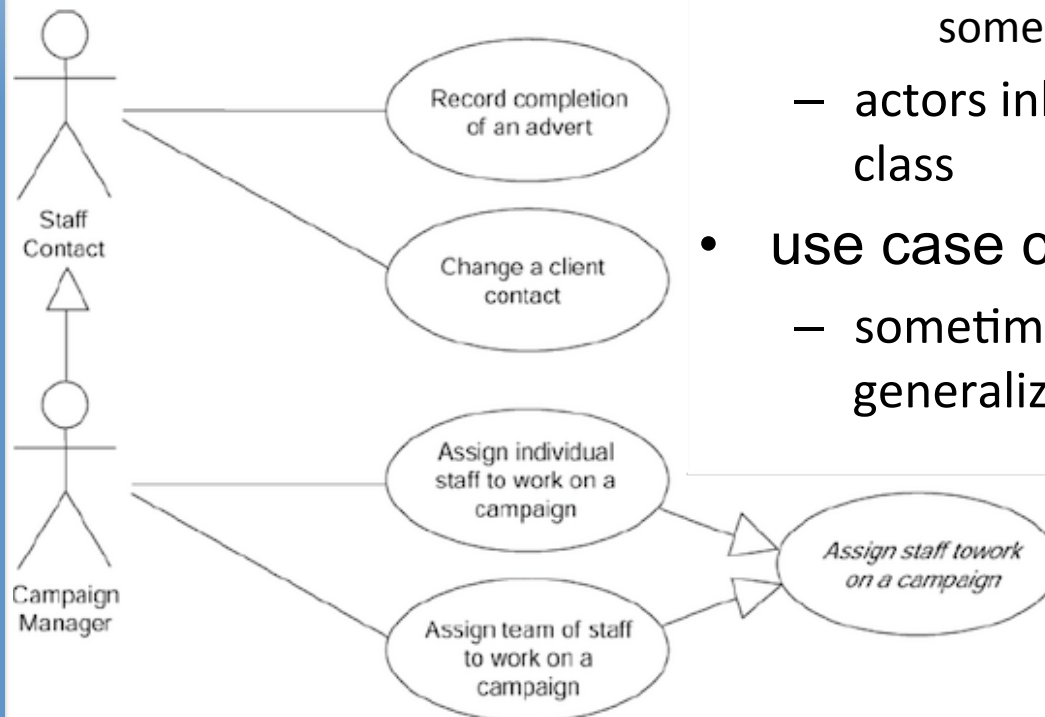
- used to avoid describing the same flow of events several times
- puts common behaviour in a use case of its own





## *using generalizations*

- actor classes
  - identify classes of actor
    - where several actors belong to a single class
    - some use cases are needed by all members in the class
    - other use cases are only needed by some members in the class
  - actors inherit use cases from the class
- use case classes
  - sometimes useful to identify a generalization of several use cases





## *describing use cases*

- for each use case:
  - a “flow of events” document, written from actor’s p.o.v.
  - describes what system must provide to actor when use case is executed
- typical contents
  - how the use case starts & ends
  - normal flow of events
  - alternate & exceptional (separate) flow of events



## *describing use cases (2)*

- documentation style
  - choices on how to elaborate the use case:
    - english language description
    - activity diagrams – good for business process
    - collaboration diagrams – good for high-level design
    - sequence diagrams – good for detailed design



## *detailed use case – description*

### **Buy a Product**

#### Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

#### Extensions:

##### 3a: Customer is Regular Customer

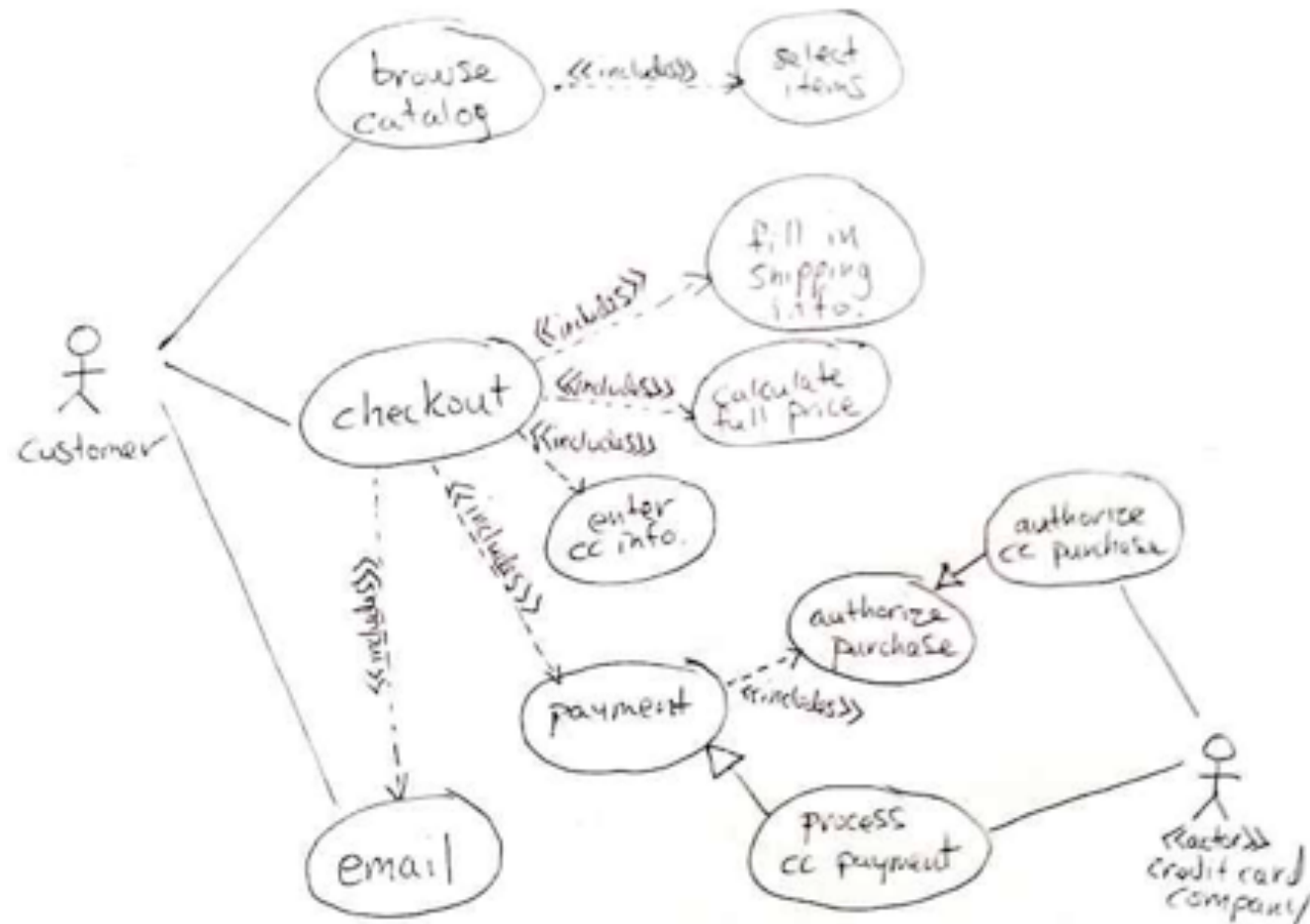
- .1 System displays current shipping, pricing and billing information
- .2 Customer may accept or override these defaults, returns to MSS at step 6

##### 6a: System fails to authorize credit card

- .1 Customer may reenter credit card information or may cancel



# ***detailed use case – diagram***





## *finding use cases*

- browse through existing documents
  - *noun phrases* may be domain classes
  - *verb phrases* may be operations and associations
  - *possessive phrases* may indicate attributes
- for each actor, ask the following questions:
  - what functions does the actor require (from the system)?
  - what does the actor do?
  - does the actor need to read, create, destroy, modify or store info?
  - does the actor need to be notified about events?
  - does the actor need to notify the system about something?
  - what do the events require in terms of system functionality?
  - could the actor's work be simplified or made more efficient if new functions were added?