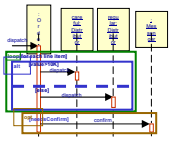
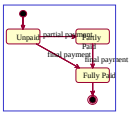
HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Μοντελοποίηση Συμπεριφοράς
(Behavioral Modeling)



Γιάννης Τζίτζικας

Διάλεξη : 11
Ημερομηνία : 29-11-2006

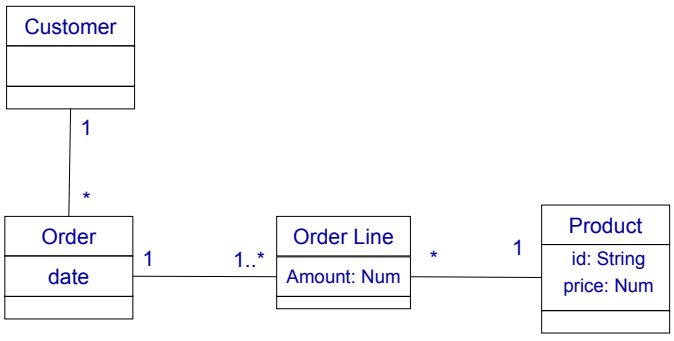
Διάρθρωση

- *Τι είναι η μοντελοποίηση Συμπεριφοράς*
- *Interaction Diagrams (διαγράμματα αλληλεπίδρασης)*
  - Sequence Diagrams (ακολουθιακά διαγράμματα)
  - Communication Diagrams (διαγράμματα επικοινωνίας)
- State diagrams (διαγράμματα καταστάσεων)

Τι εξυπηρετεί η Μοντελοποίηση Συμπεριφοράς;

- Για να δείξουμε τα μηνύματα που ανταλλάσσονται μεταξύ των αντικειμένων στα πλαίσια μιας Περίπτωσης Χρήσης

Πως τα αντικείμενα αυτού του μοντέλου επικοινωνούν; How the objects of this model interact ?



E.g. how the price of an order is calculated ?

Πως μοντελοποιούμε τη συμπεριφορά στην Αντικειμενοστραφή Ανάλυση και Σχεδίαση;

Συνήθως χρησιμοποιούμε 3 τύπους μοντέλων::

- ***Sequence diagrams***
- ***Communication diagrams*** } *Interaction diagrams*
  - **(in UML 1. they were called "Collaboration Diagrams")**
- ***Statechart diagrams***

Παρατήρηση:

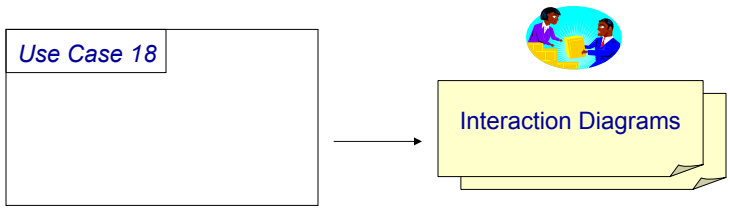
- Αν μοντελοποιήσουμε τη συμπεριφορά λεπτομερειακά, είναι σαν να υλοποιούμε το σύστημα!
- Άρα πρέπει να εστιάζουμε στα σημαντικά σημεία (*the key aspects)*
  - like storyboarding in film making (i.e. key frames)

Interaction Diagrams (Διαγράμματα Αλληλεπίδρασης)

Interaction Diagrams (Διαγράμματα Αλληλεπίδρασης)

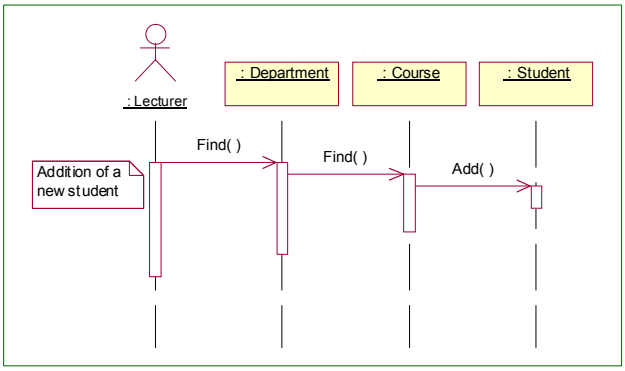
- **Sequence Diagrams** (Διαγράμματα Ακολουθίας)
- **Communication/Collaboration Diagrams** (Διαγράμματα Συνεργασίας)

- Περιγράφουν τον τρόπο με τον οποίο μια ομάδα αντικειμένων συνεργάζεται.
- Ένα διάγραμμα συνήθως περιγράφει τη συμπεριφορά σε **μια** Περίπτωση Χρήσης. Δείχνει παραδείγματα αντικειμένων που εμπλέκονται και τα μηνύματα που ανταλλάσσονται μεταξύ τους κατά τη διάρκεια της Περίπτωσης Χρήσης.



Sequence Diagrams
(Διαγράμματα Ακολουθίας)

[A] Sequence Diagrams
(διαγράμματα ακολουθίας/διαδοχής/αλληλουχίας)



- *Οριζόντια γραμμή:* δείχνει τα αντικείμενα ως κουτιά
- *Κατακόρυφη γραμμή:* χρόνος (αυξάνει προς τα κάτω)
- *Activation box*: δείχνει πότε το αντικείμενο είναι ενεργό (στη μνήμη)
- *μηνύματα*: μεταξύ των activation boxes δύο αντικειμένων

Message Types (Τύποι Μηνημάτων)

Τύποι μηνυμάτων

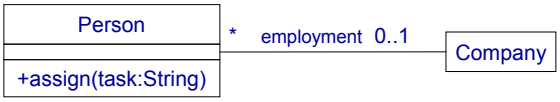
- ***Call***: κλήση μιας λειτουργίας (Invocation of an operation)
  - Ένα αντικείμενο μπορεί να στείλει ένα μήνυμα στον εαυτό του (local invocation of an operation)
- ***Return***: επιστρέφει μια τιμή στον καλούντα (returns a value to the caller)
- ***Send***: στέλνει ένα σήμα (signal) σε ένα αντικείμενο
- ***Create***: δημιουργεί ένα αντικείμενο
- ***Destroy***: καταστρέφει ένα αντικείμενο

- A signal is an object value communicated to a target object asynchronously.
- After sending a signal, the sending object continues its own execution.
- When the target object receives the signal message,it independently decides what to do about it.

Πως εικονίζουμε τα μηνύματα; How we depict messages?

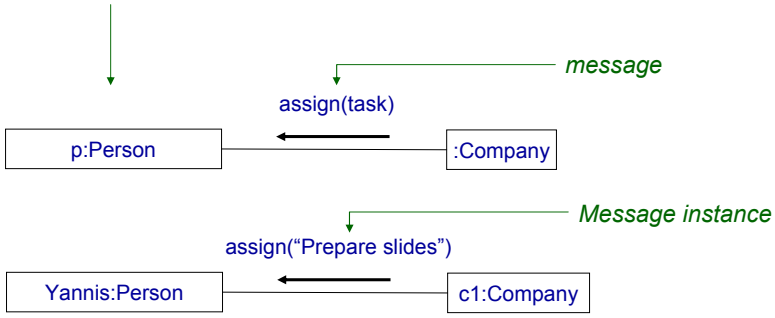
- Ένα βέλος μεταξύ των κατακόρυφων γραμμάτων δυο αντικειμένων
- Το βέλος συνοδεύεται από
  - όνομα μηνύματος (π.χ. όνομα της καλούμενης λειτουργίας)
  - Πιθανά ορίσματα (possible arguments)
  - Πληροφορίες ελέγχου (control info)
    - condition: δείχνει το πότε το μήνυμα στέλνεται, π.χ. [outOfStock]
    - iteration marker: δείχνει ότι το μήνυμα στέλνεται πολλές φορές σε πολλαπλά αντικείμενα, π.χ. *[for all order lines] // for UML 1.
- Τα μηνύματα επιστροφής (Return messages) συμβολίζονται με διάστικτα βέλη (<- -)
  - Μπορούμε να τα παραλείψουμε και να δείξουμε μόνο τα κρίσιμα.

Παραδείγματα Μηνυμάτων (Examples of Messages)



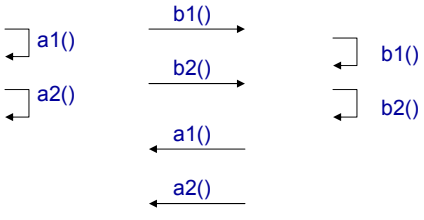
Διάγραμμα κλάσεων

Αντικείμενο τύπου Person

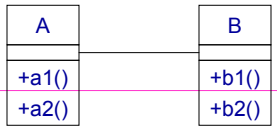


Παραδείγματα Μηνυμάτων

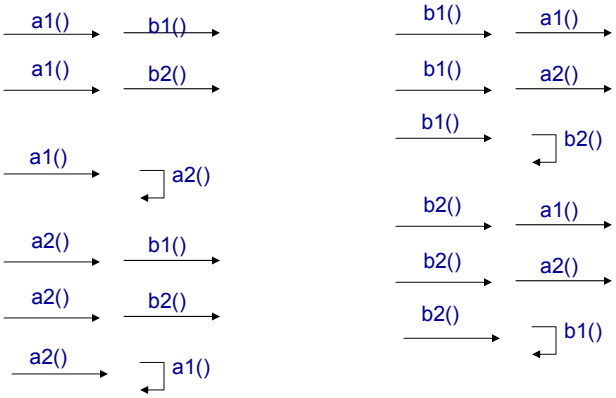
Ποια είναι όλα τα δυνατά μηνύματα θεωρώντας το παραπάνω διάγραμμα κλάσεων;



Παράδειγμα

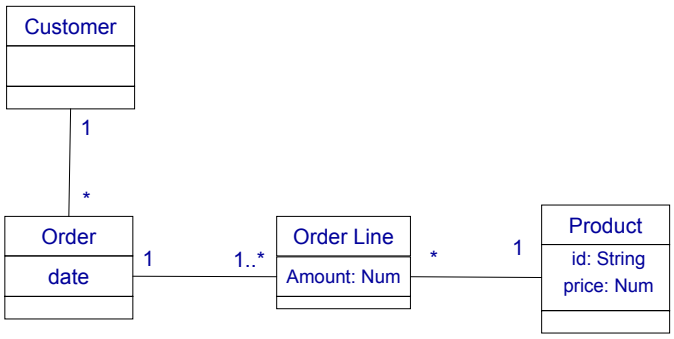


Μερικές ροές μηνυμάτων μήκους 2:



Ερώτηση: πόσες είναι όλες οι ροές μηνυμάτων μήκους 2;

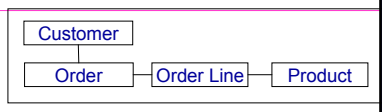
Πως τα αντικείμενα αυτού του μοντέλου επικοινωνούν; How the objects of this model interact ?

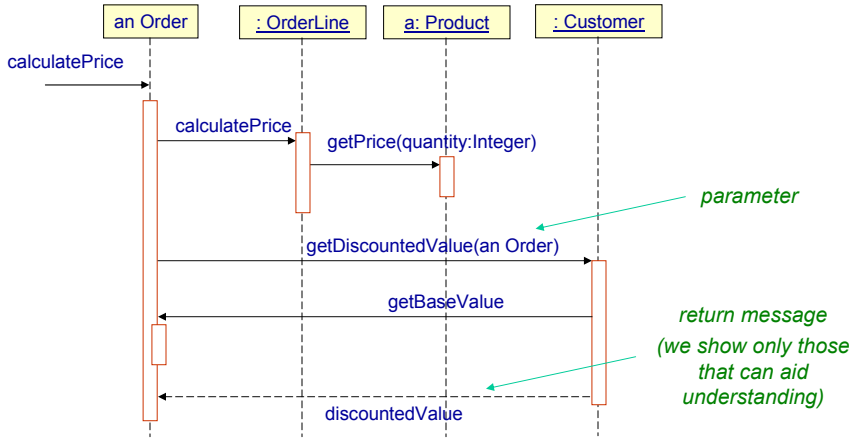


Πως υπολογίζεται η τιμή μιας παραγγελίας

Παράδειγμα διαγράμματος ακολουθίας
Example of a sequence diagram

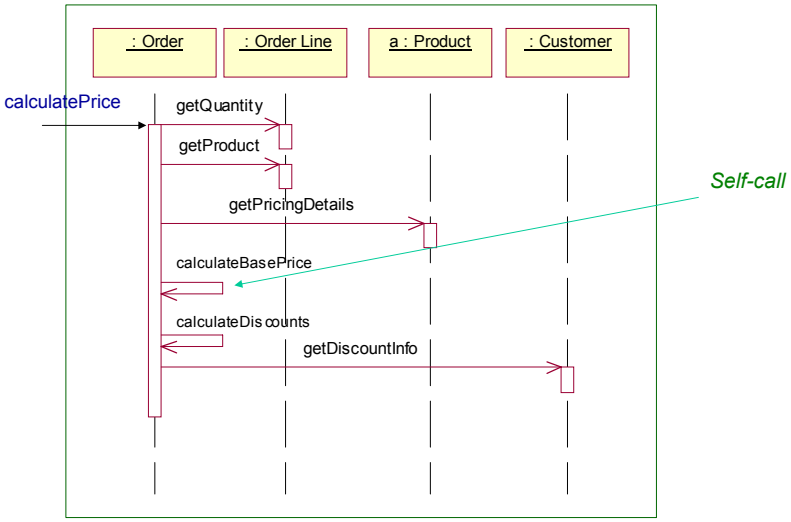
Υπολογισμός τιμής μιας γραμμής μιας παραγγελίας



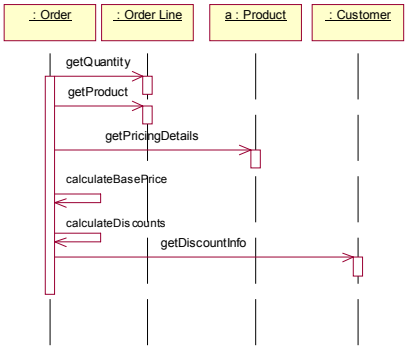


Παράδειγμα διαγράμματος ακολουθίας διαφορετικής υλοποίησης
Sequence diagram of a different implementation

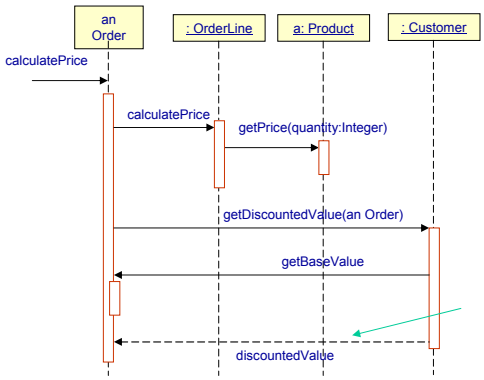
Υπολογισμός τιμής μιας γραμμής μιας παραγγελίας



Συγκρίνοντας τα δύο διαγράμματα
Comparing the two diagrams



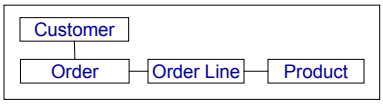
centralized control

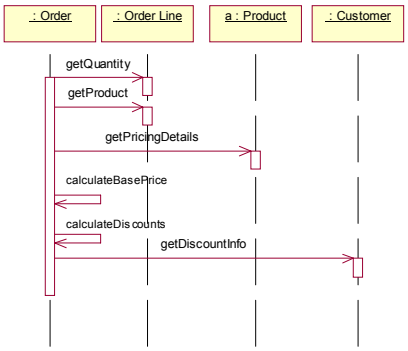


distributed control

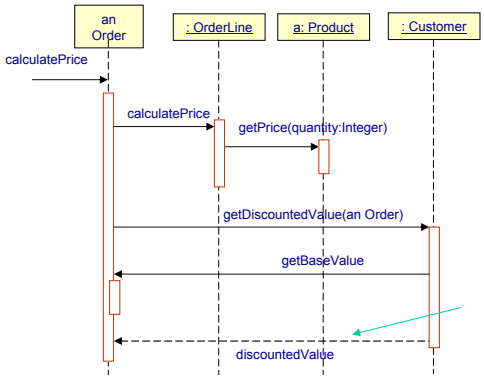
Sequence diagrams are not very good at showing details (algs with loops and conditions), but they make the calls between participants very clear and give a good picture about which participants are doing which processing.

Συγκρίνοντας τα δύο διαγράμματα



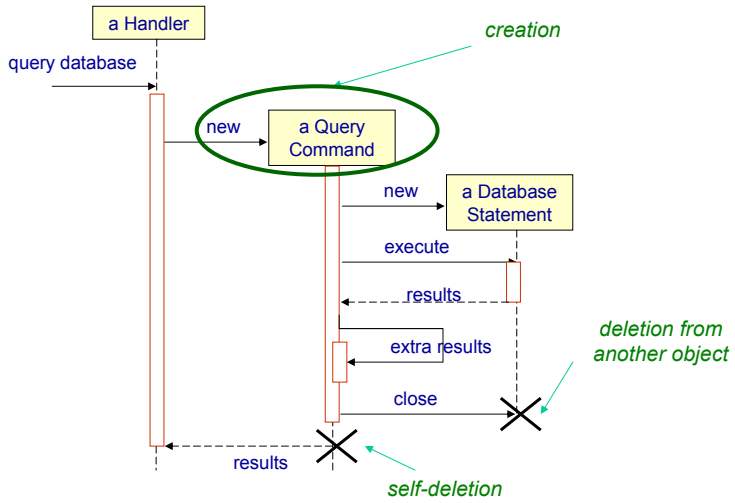


Εδώ μια Order επικοινωνεί με ένα Product (παρόλο που αυτά τα αντικείμενα δεν σχετίζονται στο διάγραμμα κλάσεων)



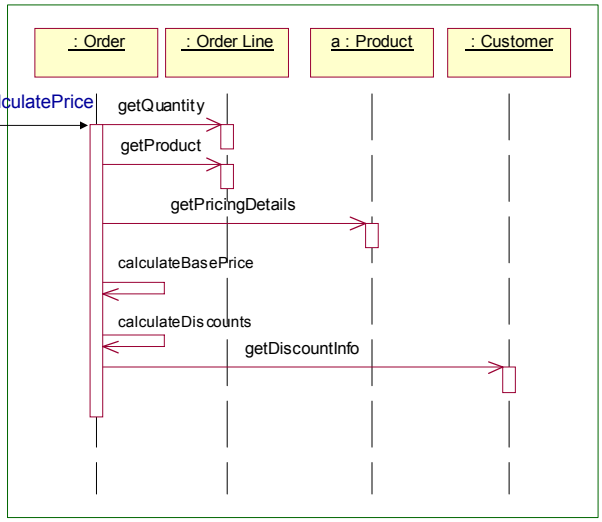
Εδώ μόνο τα αντικείμενα που είναι συσχετισμένα επικοινωνούν

Δημιουργία και Διαγραφή Συμμετεχόντων (Creating and Deleting Participants)



In a garbage-collected environment we don't delete objects directly, but it is still worth using **X** to know when an object is no longer available and can be deleted

Τι γίνεται με τους βρόγχους; (What about loops?)



Σε αυτό το διάγραμμα δεν φαίνεται πουθενά ότι οι εικονιζόμενες κλήσεις πρέπει να γίνουν για κάθε παραγγελιογραμμή (OrderLine) μιας Παραγγελίας (Order)

Μοντελοποιώντας την δομή ελέγχου (βρόγχοι, συνθήκες)
(Loops and Conditionals (modeling control logic))

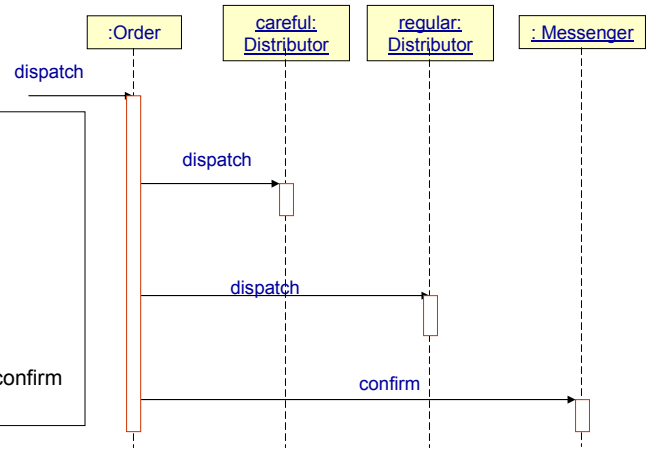
Δεν είναι αυτή η έμφαση των Διαγραμμάτων Ακολουθίας

Θα μπορούσαμε να χρησιμοποιήσουμε Activity Diagrams ή ψευδοκώδικα για αυτόν το σκοπό.

```
procedure dispatch
  foreach (lineitem)
       if (product.value > $10K)
            careful.dispatch
       else
            regular.dispatch
       endif
    endfor
  if (needsConfirmation) messenger.confirm
end
```

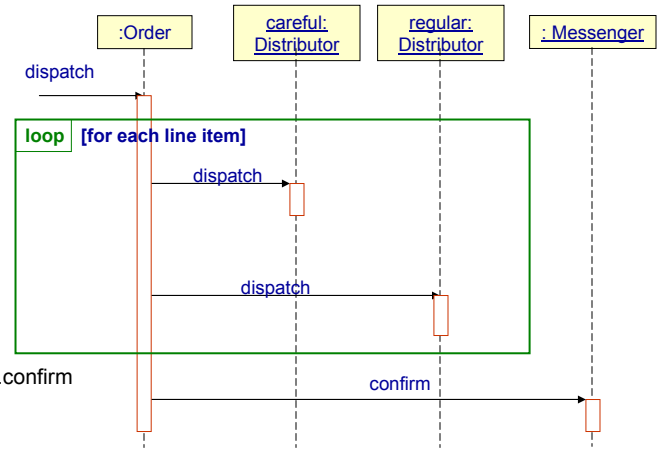

Μοντελοποιώντας την δομή ελέγχου (βρόγχοι, συνθήκες)

```
procedure dispatch
  foreach (lineitem)
       if (product.value > $10K)
            careful.dispatch
       else
            regular.dispatch
       endif
    endfor
  if (needsConfirmation) messenger.confirm
end
```



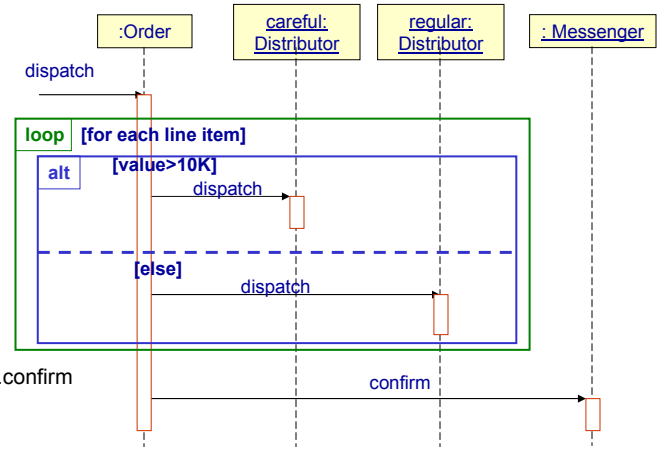
Μοντελοποιώντας την δομή ελέγχου (βρόγχοι, συνθήκες)

```
procedure dispatch
  foreach (lineitem)
      if (product.value > $10K)
          careful.dispatch
      else
          regular.dispatch
      endif
   endfor
  if (needsConfirmation) messenger.confirm
end
```



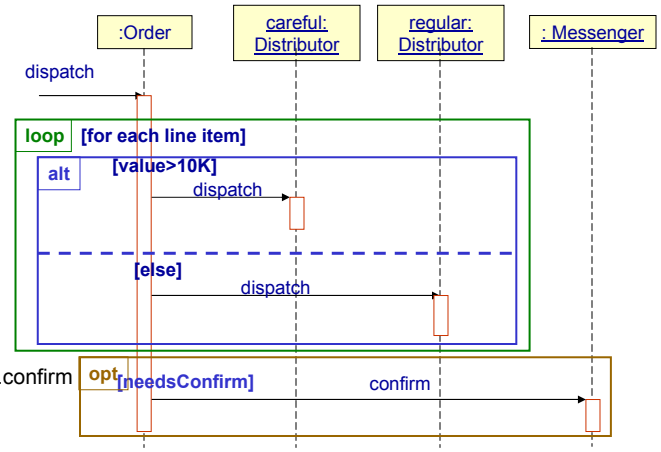
Μοντελοποιώντας την δομή ελέγχου (βρόγχοι, συνθήκες)

```
procedure dispatch
  foreach (lineitem(
      if (product.value > $10K)
          careful.dispatch
      else
          regular.dispatch
      endif
   endfor
  if (needsConfirmation) messenger.confirm
end
```



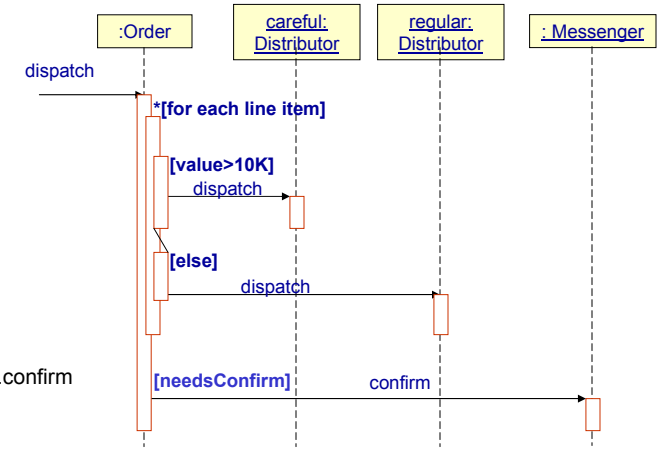
Μοντελοποιώντας την δομή ελέγχου (βρόγχοι, συνθήκες)

```
procedure dispatch
  foreach (lineitem(
      if (product.value > $10K)
          careful.dispatch
      else
          regular.dispatch
      endif
    endfor
  if (needsConfirmation) messenger.confirm
end
```



Βρόγχοι και Συνθήκες (συμβολισμοί της UML 1)

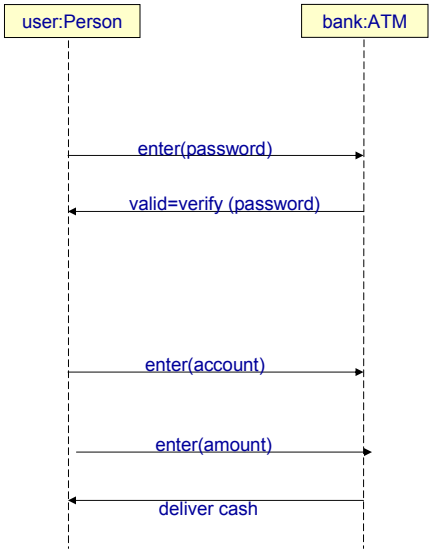
```
procedure dispatch
  foreach (lineitem(
      if (product.value > $10K)
          careful.dispatch
      else
          regular.dispatch
      endif
    endfor
  if (needsConfirmation) messenger.confirm
end
```



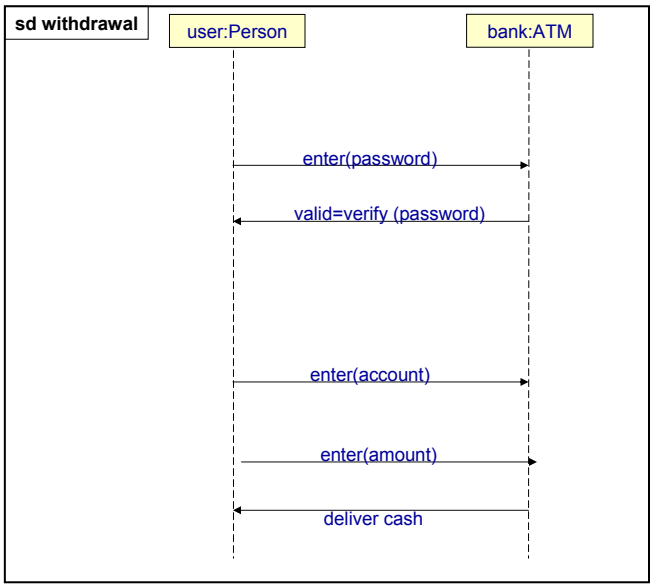
Τελεστές διαγραμμάτων ακολουθίας (Operators for sequence diagrams)

- **alt**: alternative multiple fragments; only the one whose condition is true will be executed
- **opt**: optional fragments; executed only if its condition is true (equiv to alt with one fragment)
- **par**: parallel execution of fragments
- **loop**: the fragments will be executed multiple times (based on the guard)
- **region**: critical region; the fragment can have only one thread executing it at once
- **neg**: the fragment shows an invalid interaction
- **ref**: reference: refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return a value.
- **sd**: sequence diagram; used to surround the entire diagram

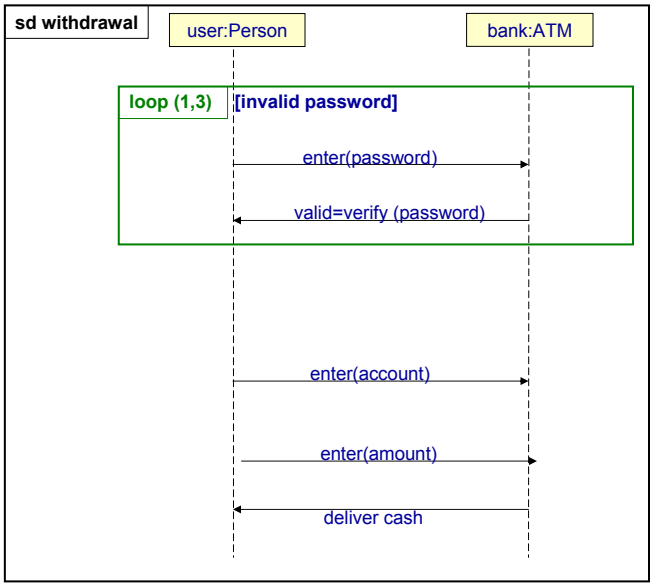
Παράδειγμα: Ανάληψη μετρητών από ATM (Withdraw cash from an ATM)



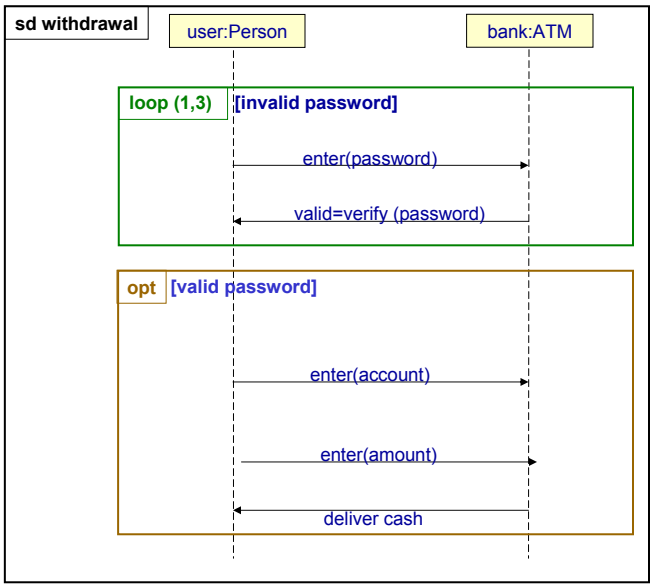
Ανάληψη μετρητών από ATM
Παράδειγμα χρήσης του τελεστή sd



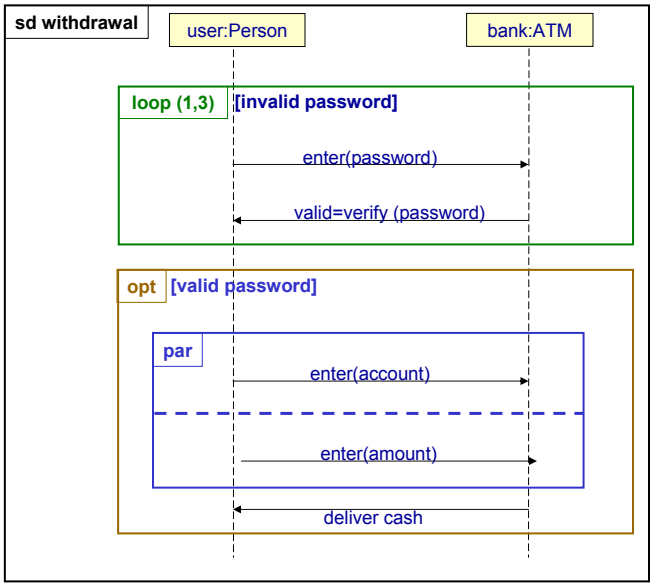
Ανάληψη μετρητών από ATM
Παράδειγμα χρήσης των τελεστών sd, loop



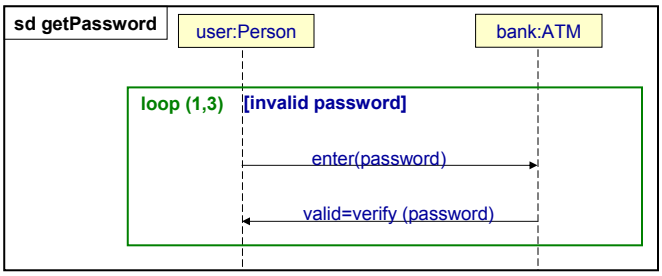
Ανάληψη μετρητών από ATM
Παράδειγμα χρήσης των τελεστών sd, loop, opt

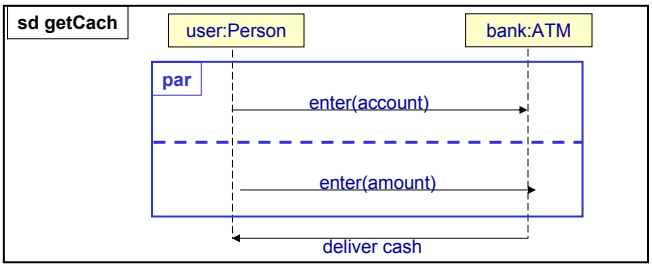


Ανάληψη μετρητών από ATM
Παράδειγμα χρήσης των τελεστών sd, loop, opt, par

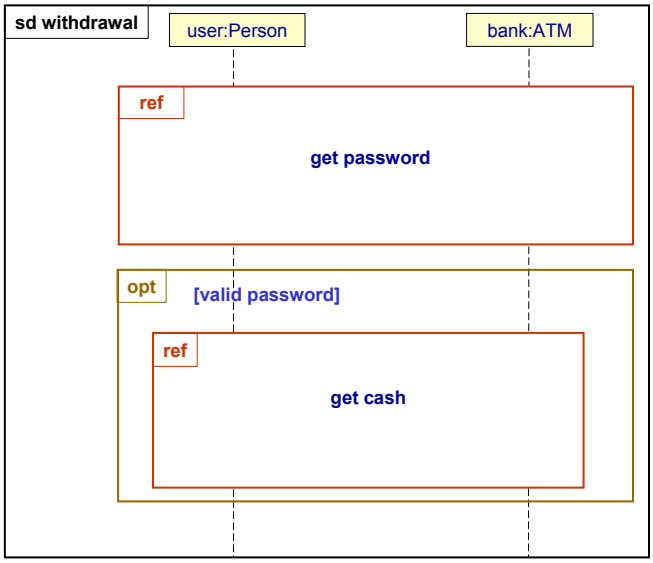


Έστω ότι έχουμε ήδη ορίσει τα παρακάτω 2 διαγράμματα

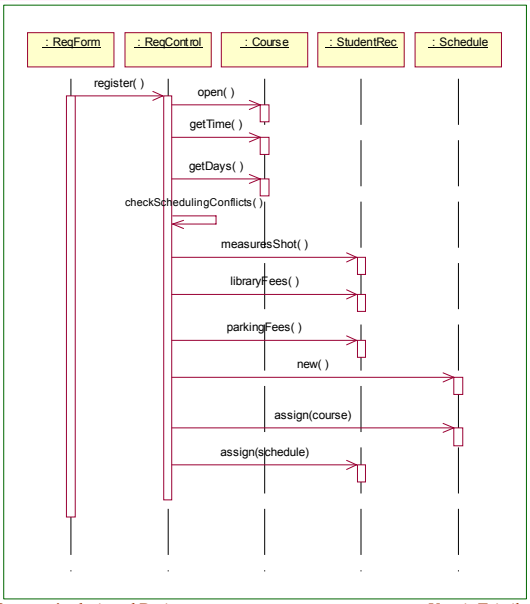




Μπορούμε να τα «εκμεταλλευτούμε» (επαναχρησιμοποιήσουμε) με τον τελεστή ref



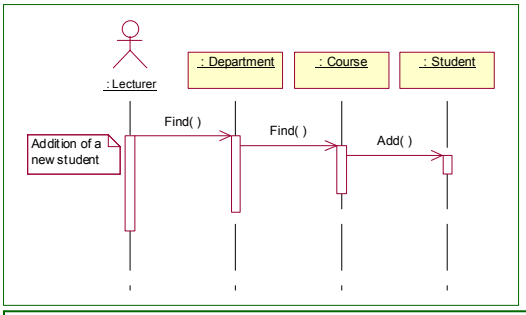
Άλλο ένα παράδειγμα



Communication Diagrams (UML 2.0)
~ Collaboration Diagrams (UML V 1.3)

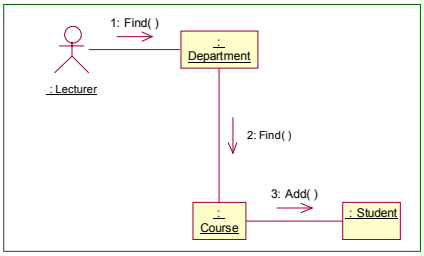
[B] Communication Diagrams (διαγράμματα επικοινωνίας) ≡ Collaboration Diagrams (v.1)

Sequence Diagram



≡

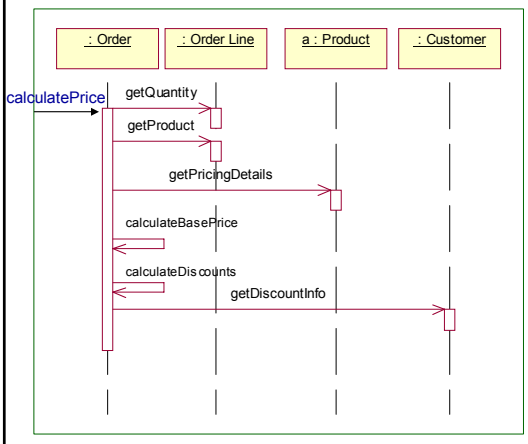
Communication Diagram

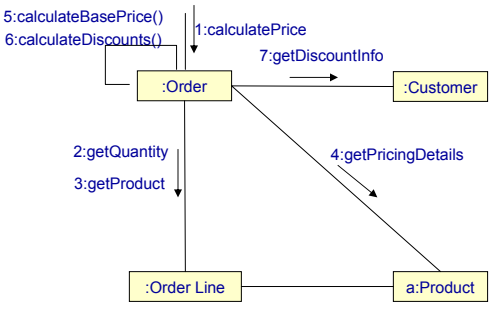


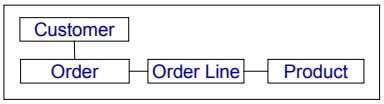
Here the sequence is indicated by numbering messages.
- Advantage: better exploits the drawing space (more compact)
- Weakness: makes it harder to see the sequence (comparing to sequence diagrams)

Sequence Diagram <=> Communication Diagram
- Automatic transformation is possible (e.g. F5 in Rational rose)

Sequence vs Collaboration Diagrams: Example

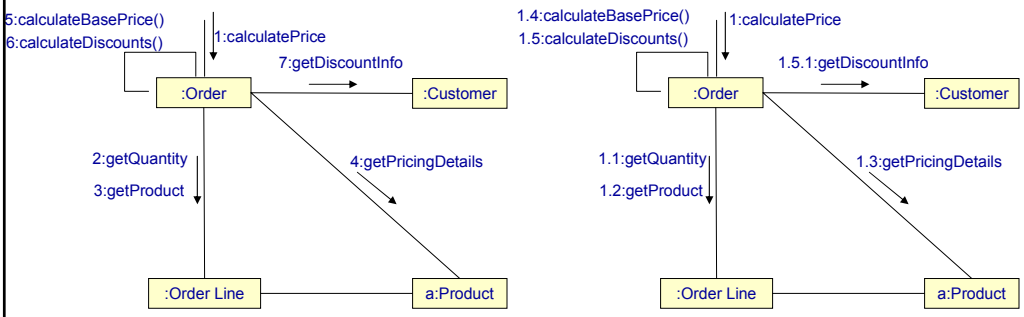






It is like an object diagram that shows message passing relationships instead of aggregation or generalization associations

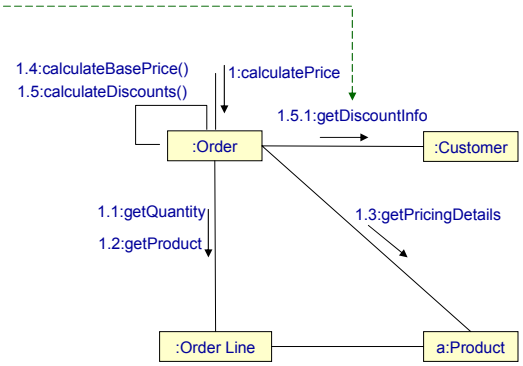
Τρόποι Αρίθμησης
Numbering Methods



- Numbering methods
  - 1, 2, 3, …
  - 1, 1.1, 1.1.1, 1.1.2, 2.1 (Decimal numbering (used by UML))
- communication diagrams have not a precise notation for control logic
  - we could however use iteration markers and guards

Numbering Methods (II)

Why 1.5.1 and not 1.6?

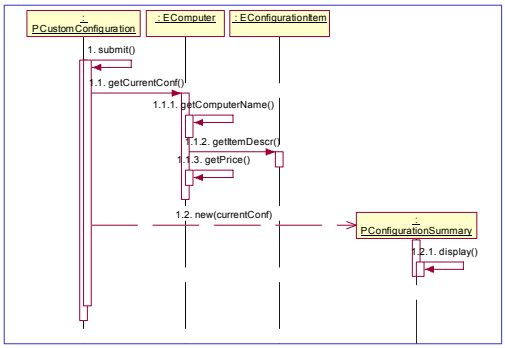


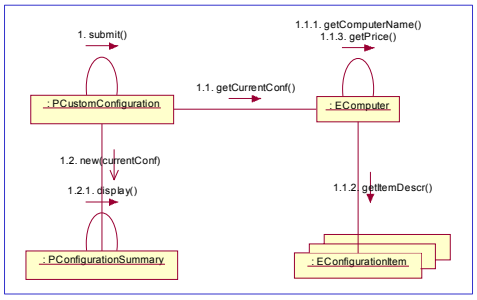
- Procedural (or nested) sequence
  - 1, 2, 2.1, 2.2
- Flat sequence
  - 1, 2, 3, 4

(2.1 and 2.2 are performed while the object of 2 is still active)

Sequence Diagrams vs Communication Diagrams

- Μερικοί (προγραμματιστές/αναλυτές) προτιμούν τα μεν άλλοι τα δε.

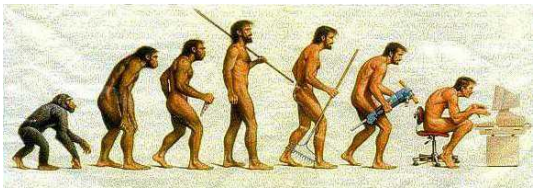




Πότε να χρησιμοποιούμε;

- When to use Interaction Diagrams
  - To show the behaviour of several objects within a single Use Case
  - Tip: Focus on simplicity
    - If the control is complex split it to several interaction diagrams
- When use not Interaction Diagrams
  - If you want to look at the behaviour of a single object across multiple use cases, then use a **state diagram**
  - If you want to look at the behaviour across many use cases and many threads consider an **activity diagram**

Διαγράμματα Καταστάσεων (State Diagrams)



State Diagrams
Διαγράμματα Καταστάσεων

Παρουσιάζει όλες τις πιθανές **καταστάσεις** που μπορεί να έχει ένα συγκεκριμένο αντικείμενο (σε όλη τη διάρκεια ύπαρξης του) και πως αυτές **αλλάζουν** ανάλογα με τα **γεγονότα** (events) που φθάνουν στο αντικείμενο

- άρα δεν περιοριζόμαστε σε μια Use Case

- Ένα διάγραμμα καταστάσεων συνήθως αφορά μία κλάση

State Diagrams
Διαγράμματα Καταστάσεων

Μπορούμε να τα χρησιμοποιήσουμε από διάφορες προοπτικές

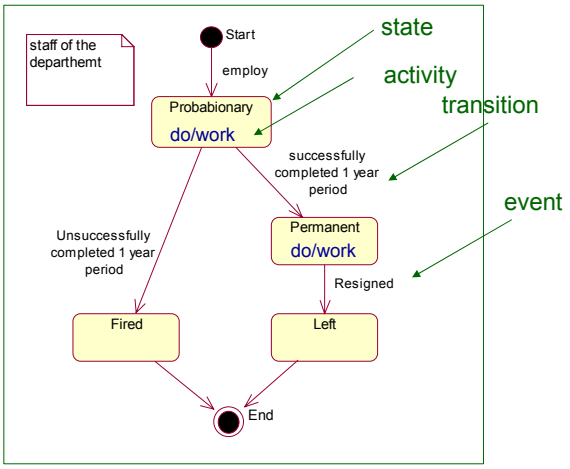
Προοπτικές

- **Εννοιολογική**:
  - ***Π.χ. ποιες είναι οι καταστάσεις μιας παραγγελίας στην επιχείρηση; Οι ακυρώσεις επιτρέπονται;***
- **Προδιαγραφική**
  - ***Οι καταστάσεις που πρέπει διαχειριστούν οι διεπαφές των κλάσεων***
- **Υλοποιητική**
  - ***Οι πραγματικές καταστάσεις των αντικειμένων υλοποίησης***

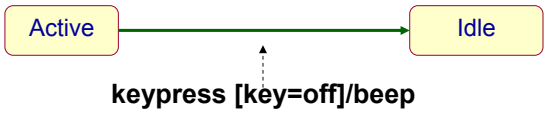
Βασικές έννοιες

- States
- Transitions
- Events
- Activities

- Καταστάσεις
- Μεταβάσεις
- Γεγονότα
- Δραστηριότητες



Μεταβάσεις
Transitions



Οι μεταβάσεις μπορεί να έχουν ετικέτες της μορφής: **Event[Condition]/Action**
- και τα τρία συστατικά τους είναι προαιρετικά

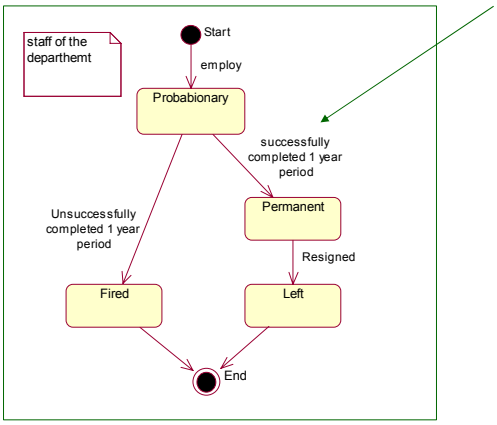
- Event (γεγονός)
  - Εάν είναι κενό (nil) τότε η μετάβαση πραγματοποιείται μόλις η εργασία έχει ολοκληρωθεί
- Condition (συνθήκη)
  - Λογική συνθήκη (πρέπει να αληθεύει για να πραγματοποιηθεί η μετάβαση)
  - Οι συνθήκες των μεταβάσεων που εκκινούν από μια κατάσταση πρέπει να είναι αμοιβαίως αποκλειόμενες (mutually exclusive) ώστε να διασφαλίζεται ότι έχουμε μια μοναδική επόμενη κατάσταση
- Action (δράση)
  - Εργασία που γίνεται «στιγμιαία» και δεν διακόπτεται

Παράδειγμα μετάβασης με ετικέτα της μορφής **Event[Condition]/Action**

Διάγραμμα καταστάσεων ενός υπαλλήλου

Event[Condition]/Action

After 1 year [successful so far]/inform the director of personnel



Τύποι Γεγονότων
Kinds of Events

- Entry
  - any action related to entry event is executed whenever the given state is entered via a transition
- Exit
  - when we exit the transition
- After 20 minutes
  - example of event generated after a period of time
- When (temperature > 40)
  - example of event generated when a condition becomes true
- ...

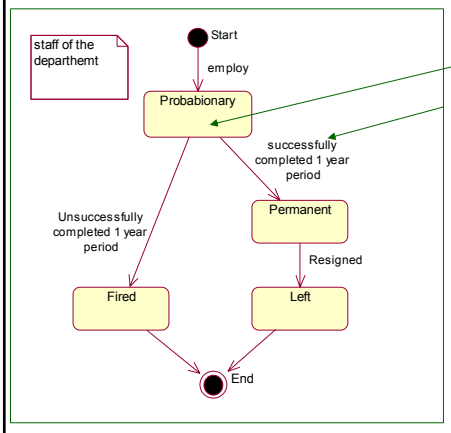
Δράσεις vs Δραστηριότητες (UML V2) ή
Εσωτερικές vs Εξωτερικές Δραστηριότητες (UML V2)

UML V 1. : Actions vs Activities
UML V 2.0: Internal vs External Activities

Η διαφορά είναι η εξής:

- Actions (ή internal activities)
  - σχετίζονται συνήθως με μεταβάσεις (και διαρκούν λίγο)
  - **δεν διακόπτονται** (not interruptible)
- Activities (ή external activities)
  - σχετίζονται με καταστάσεις (μπορεί να διαρκούν πολύ)
  - **Μπορούν να διακοπούν** από γεγονότα (events)
- Κάθε κατάσταση μπορεί να έχει μια activity που σχετίζεται με αυτήν που συνήθως συμβολίζεται με το εξής συντακτικό: **do/activity**

Actions vs Activities



activity

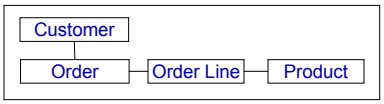
do/working

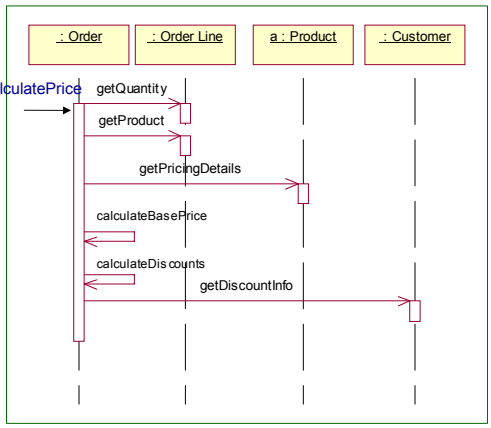
action

After 1 year [successful so far]/inform the director of personnel

Παράδειγμα

Θυμηθείτε το διάγραμμα κλάσεων με τις Παραγγελίες και το διάγραμμα ακολουθίας που περιγράφει τον υπολογισμό της αξίας μιας παραγγελίας

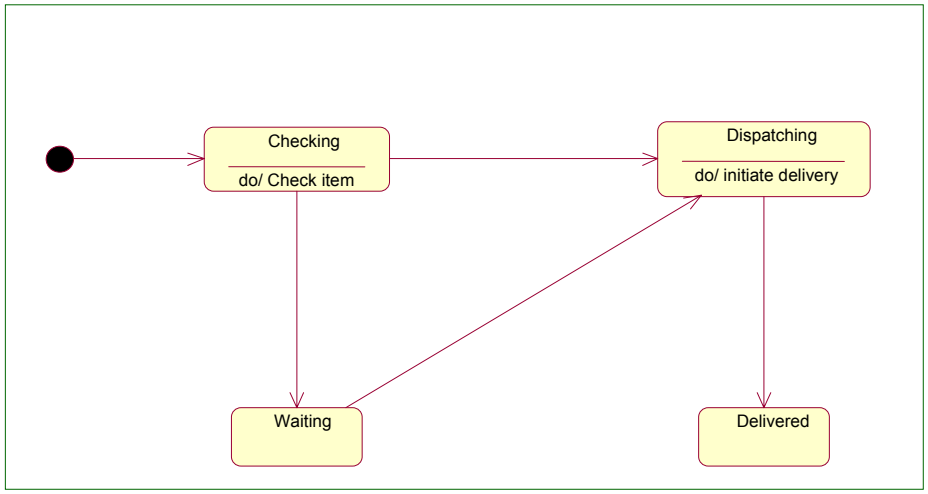




Ποιες είναι οι (σημαντικές) καταστάσεις ενός αντικειμένου τύπου Order?

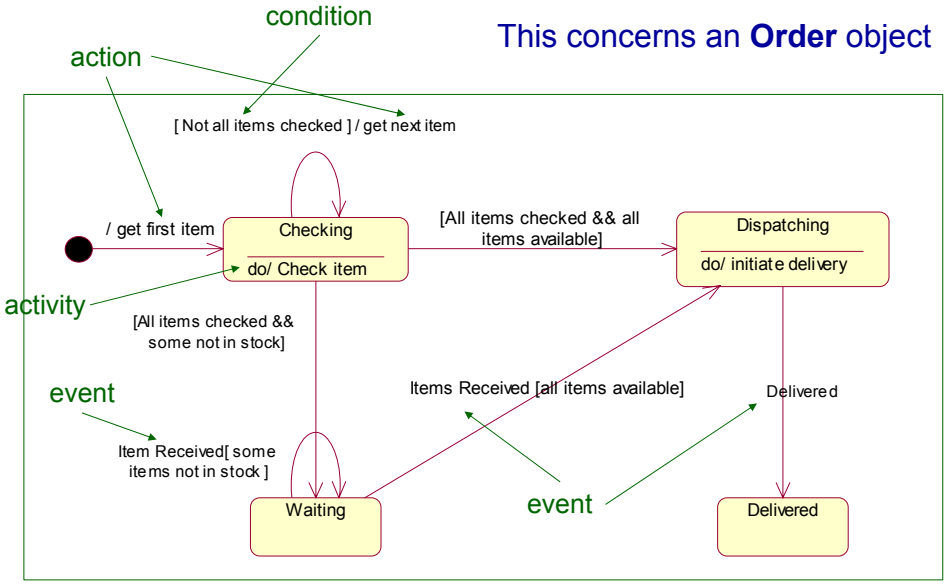
Οι καταστάσεις ενός αντικειμένου τύπου Order

This concerns an **Order** object



Οι καταστάσεις ενός αντικειμένου τύπου Order (II)

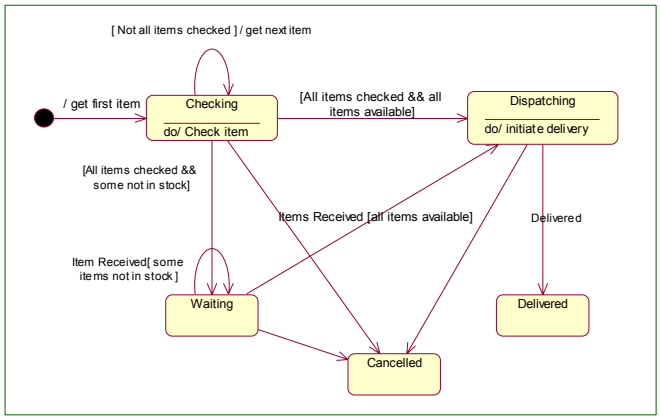
This concerns an **Order** object



Παράδειγμα (συνέχεια)

Έστω ότι θέλουμε να μπορούμε να ακυρώσουμε οποιαδήποτε στιγμή

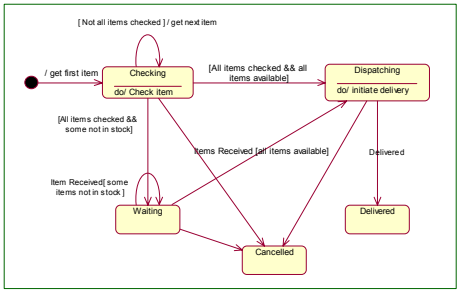
Λύση 1: προσθήκη μιας μετάβασης cancel από κάθε κατάσταση

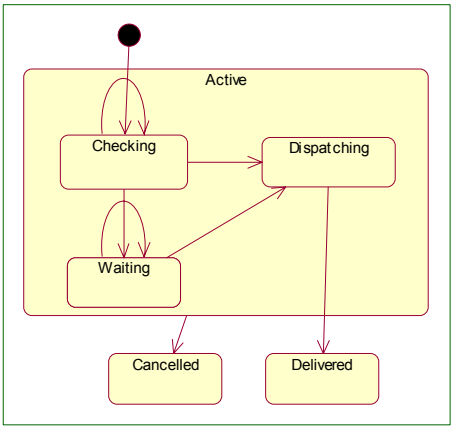


Παράδειγμα (συνέχεια) : **Superstates**

Έστω ότι θέλουμε να μπορούμε να ακυρώσουμε οποιαδήποτε στιγμή

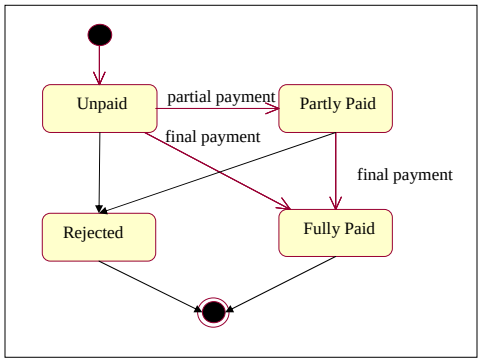
*Λύση 2 : Ορισμός μιας **superstate** και προσθήκη μετάβασης cancel μόνο σε αυτήν (οι «υποκαταστάσεις» (substates) την κληρονομούν)*





Άλλο παράδειγμα

Εδώ βλέπουμε τις καταστάσεις ενός αντικειμένου τύπου Order όσον αφορά την πληρωμή



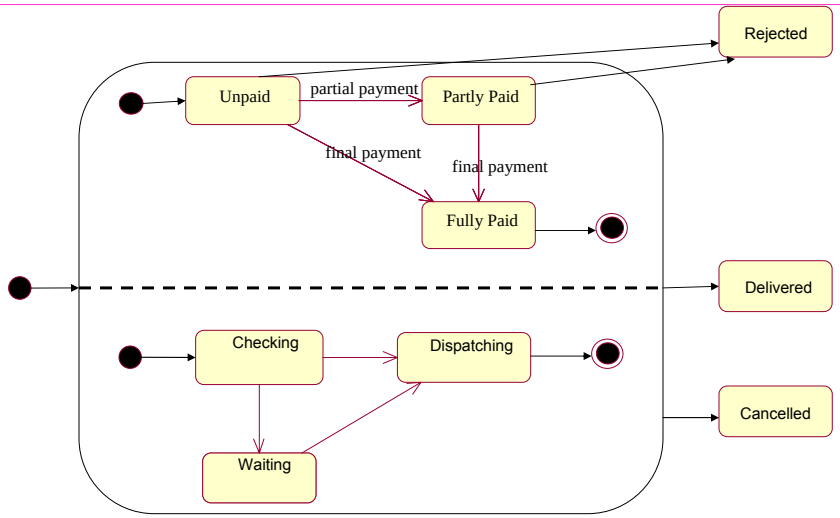
Ερώτημα: Πώς να συνδυάσουμε αυτές τις καταστάσεις με τις προηγούμενες (checking, waiting, dispatching, delivered) ;

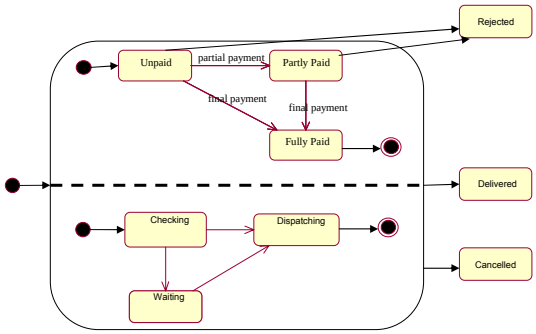
Concurrent State Diagrams (Διαγράμματα με Ταυτόχρονες Καταστάσεις)

Επιτρέπουν «παράλληλη» εκτέλεση

- Πολλές καταστάσεις μπορεί να είναι ενεργές (active) ταυτόχρονα
- Όταν ένα αντικείμενο βγει από τις παράλληλες καταστάσεις μεταβαίνει σε μία κατάσταση

Concurrent State Diagrams





- Άρα τα concurrent state diagrams μας επιτρέπουν να έχουμε ανεξάρτητα σύνολα καταστάσεων.
- Σύνδεση με προηγούμενη ύλη:
  - Θυμηθείτε ότι στα διαγράμματα δραστηριοτήτων (activity diagrams) παριστάναμε την παραλληλία δραστηριοτήτων με fork και join
  - Εδώ μιλάμε για παράλληλες καταστάσεις

Internal Activities (or self-transitions)
Εσωτερικές Δραστηριότητες (ή αυτομεταβάσεις)

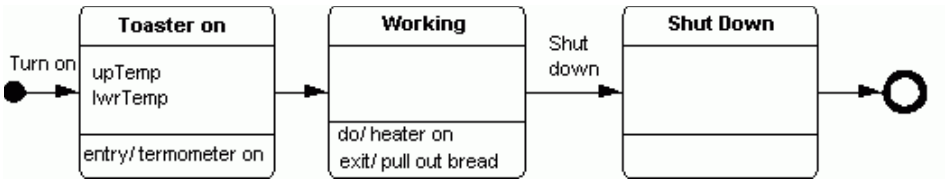
- Ακόμα και οι καταστάσεις μπορούν να αντιδρούν σε γεγονότα χωρίς να έχουμε μετάβαση (προς άλλη κατάσταση). Συγκεκριμένα μπορούμε μέσα στο κατάσταση να βάλουμε **event[guard]/activity**
- Έστω ότι έχουμε ένα αντικείμενο τύπου TextField το οποίο έχει μια κατάσταση Typing. Παρακάτω βλέπουμε τα εσωτερικά γεγονότα και τις δραστηριότητες.

Typing
entry/highlight all
exit/update field
character/handle character
help[verbose]/open help message
help[quiet]/update status bar

Παράδειγμα: Τοστιέρα

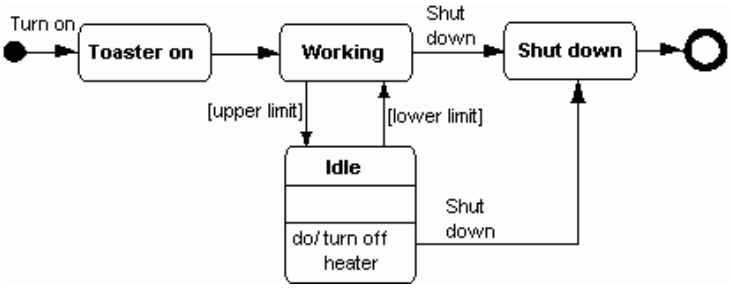
From http://odl-skopje.etf.ukim.edu.mk/uml-help/

- Υποθέστε ότι θέλετε να σχεδιάσετε μια τοστιέρα. Εδώ θα εστιάσουμε στα διαγράμματα κατάστασης.
- *Ποια είναι τα βήματα για την ετοιμασία ενός τοστ;*
- Αρχικά ανάβουμε το μηχάνημα, τοποθετούμε το ψωμί και αναμένουμε για μερικά λεπτά. Το αρχικό διάγραμμα καταστάσεων φαίνεται παρακάτω:



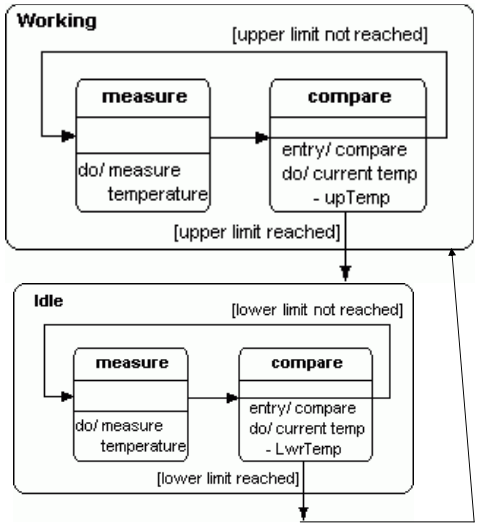
Τοστιέρα (2)

- Το προηγούμενο διάγραμμα είναι ελλιπές. Για αποφυγή καψίματος προσθέτουμε μια κατάσταση Idle και μεταβάσεις lowerLimit kai upperLimit

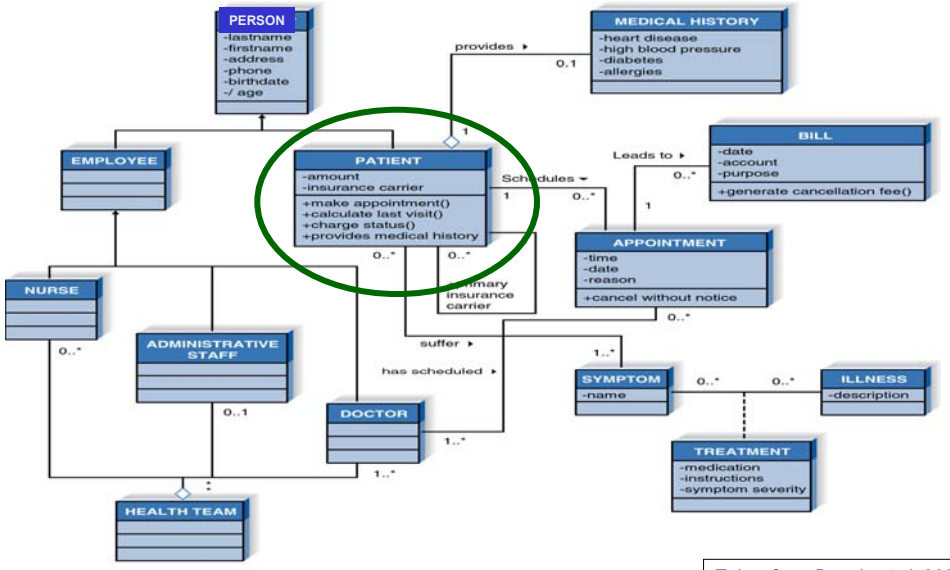


Τοστιέρα (3)

Η μετάβαση μεταξύ Working και Idle δεν παρουσιάστηκε λεπτομερώς. Μπορούμε με substates να περιγράψουμε πιο συγκεκριμένα.

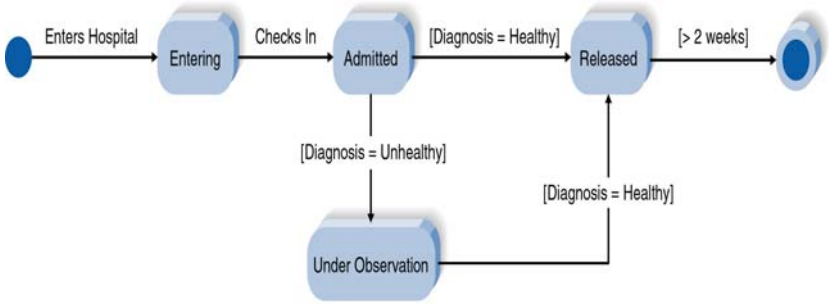


Παράδειγμα:
Class Diagram for the Appointment System



Taken from Dennis et al. 2005

Application: Hospital
State diagram of a Patient



Dennis: SAD
Fig: 8-11 W-37 100% of size
Fine Line Illustrations (516) 501-0400

Διαγράμματα Καταστάσεων
Πότε τα χρησιμοποιούμε ;

- Για να περιγράψουμε τη συμπεριφορά **ενός αντικειμένου** επί ενός συνόλου Περιπτώσεων Χρήσης
- Τα διαγράμματα αυτά δεν είναι «βολικά» αν υπάρχουν πολλά συνεργαζόμενα αντικείμενα
  - (τότε χρησιμοποιούμε interaction diagrams ή activity diagrams)

Classical cases for using state machine diagrams:

- Example applications
  - Cruise controls
  - vendor machines
- Formal methods
  - verification of network protocols

Summary

Sequence diagrams (and *Communication diagrams*)

- illustrate the classes that participate in a use case and the messages that pass between them.

State diagrams

- show the different states that a single class passes through in response to events.