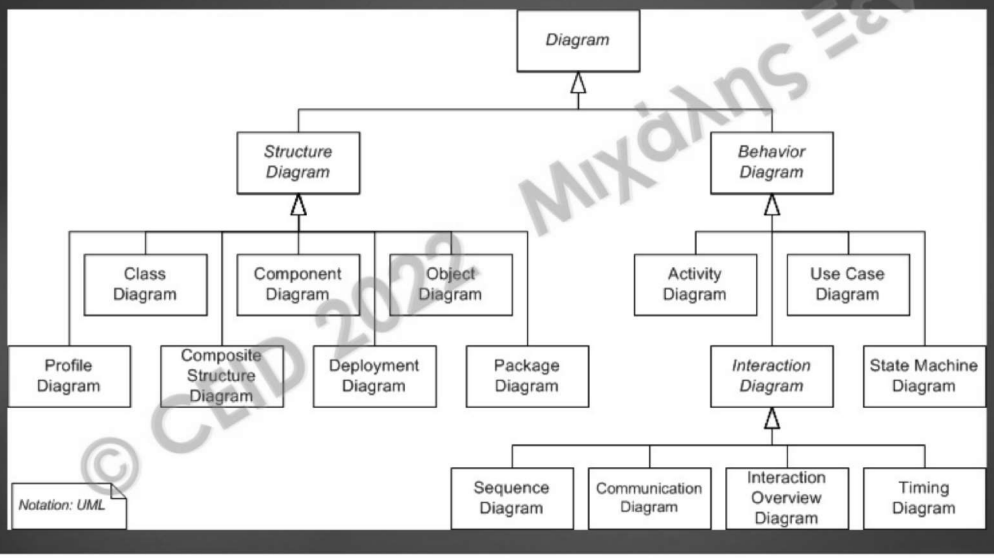
Επανάληψη: Class diagram



Class diagram

Γιατί μοντελοποιούμε;

Το λογισμικό είναι εξ ορισμού μια αφαίρεση: γιατί πρέπει και να το μοντελοποιούμε;

Το λογισμικό γίνεται όλο και μεγαλύτερο

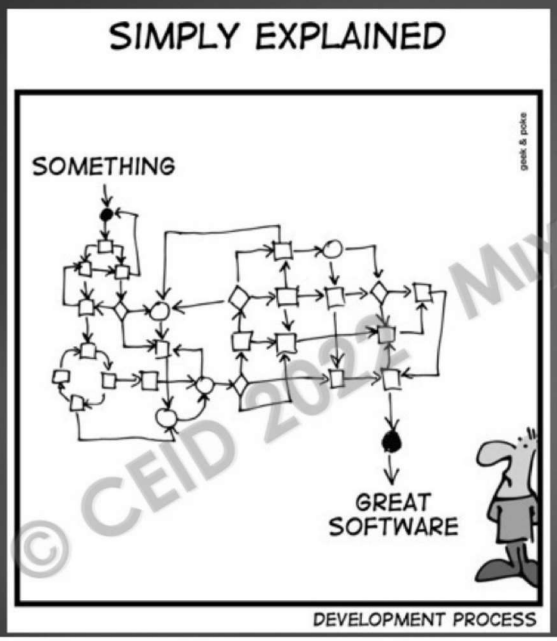
- Είδαμε παραδείγματα με Μ και Β LOC
- Ένας προγραμματιστής δε μπορεί να διαχειριστεί μόνος του μια τέτοια ποσότητα κώδικα

Ο κώδικας συχνά δεν είναι εύκολα κατανοητός από developers που δεν συμμετείχαν στη διαδικασίας ανάπτυξης

Χρειαζόμαστε απλούστερες αναπαραστάσεις (δηλ. μοντέλα) για τα πολύπλοκα συστήματα

ΠΡΟΣΟΧΗ:

- **Η μοντελοποίηση είναι ένας τρόπος να διαχειριζόμαστε την πολυπλοκότητα**
- **ΌΧΙ ΝΑ ΔΗΜΙΟΥΡΓΗΣΟΥΜΕ ΠΟΛΥΠΛΟΚΟΤΗΤΑ**



Σχεδίαση

Ανάπτυξη λογισμικού

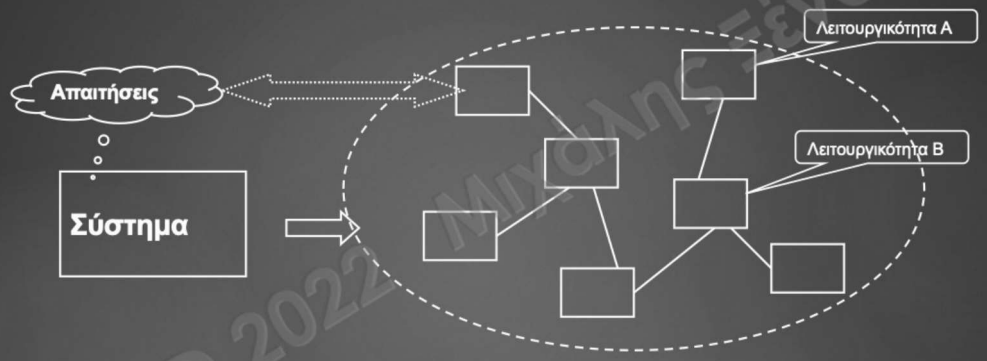
Οι απαιτήσεις (requirements) καθορίζουν
- Τους στόχους που πρέπει να ικανοποιεί το σύστημα

Οι προδιαγραφές (specifications) καθορίζουν
- Την συμπεριφορά που εκδηλώνει εξωτερικά το σύστημα

Η αρχιτεκτονική (architecture) καθορίζει
- Τα βασικά συστατικά σε επίπεδο συστήματος
- Τις μεθόδους που χρησιμοποιούν για να αλληλεπιδρούν
- Την τεχνολογία που χρησιμοποιείται

Η σχεδίαση (design) καθορίζει
- Πώς θα γίνει η δουλειά
- Τον κώδικα που χρειάζεται να γραφτεί
- (εστιάζουμε στην ΟΟ σχεδίαση)

Σχεδίαση (οποιουδήποτε τεχνικού έργου):



- Η αποσύνθεση ενός συστήματος σε τμήματα (μονάδες)
- Ο καθορισμός των σχέσεων μεταξύ των τμημάτων
- Η ανάθεση αρμοδιοτήτων σε κάθε σχήμα
- Η επικύρωση ότι όλα τα τμήματα μαζί επιτυγχάνουν τους σκοπούς του συστήματος

ΟΟ Σχεδίαση

Η διαδικασία προσδιορισμού του:

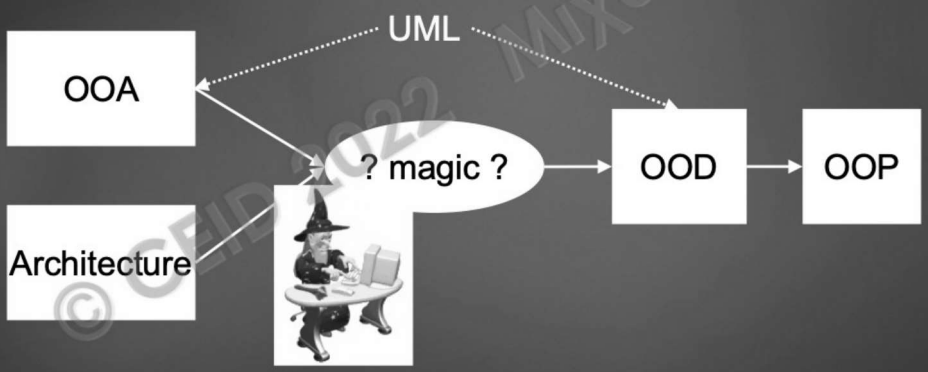
- Πώς θα λειτουργεί το λογισμικό ώστε να επιτύχει τους στόχους του
- Πώς θα συνεργαστούν στο λογισμικό οι κλάσεις που έχουν περιγραφεί στην ΟΟ ανάλυση
- Πώς θα πρέπει να υλοποιηθούν οι συνδέσεις και οι σχέσεις
- Πώς μπορούν να βοηθήσουν συστατικά που έχουν αγοραστεί/εντοπιστεί
- Η καλύτερη (βελτίωση δηλαδή) εκτίμηση προαπαιτούμενων σε όρους ανθρώπινου δυναμικού και υποδομών

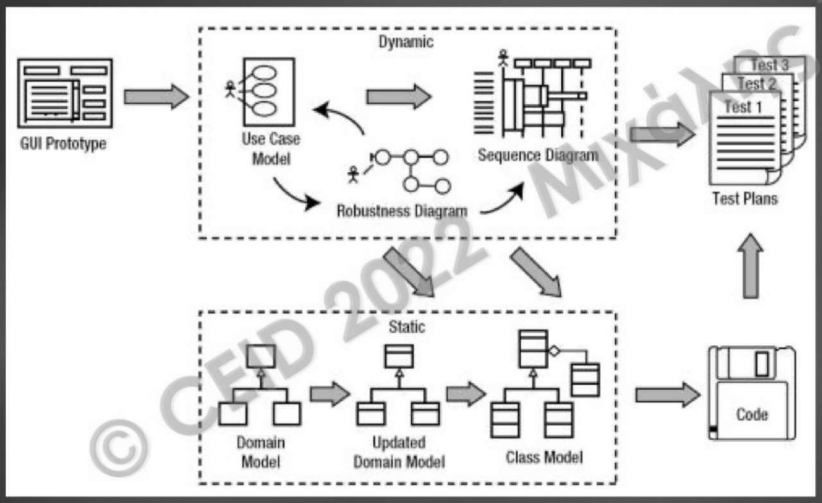
ΟΟ Σχεδίαση

- Η διαδικασία περαιτέρω περιγραφής των κλάσεων πάνω στις οποίες θα χτίσουμε το σύστημα με τη μορφή μεθόδων και ιδιοτήτων
- Η προσθήκη κλάσεων που δεν αποτελούν προφανώς μέρος του, όπως οι αφηρημένες κλάσεις και τα interfaces
- Η περιγραφή του τρόπου που οι κλάσεις φτιάχνουν components

Από την ανάλυση στη σχεδίαση

Δεν υπάρχει μέθοδος βήμα-προς-βήμα για τη μετάβαση από την ΟΟ ανάλυση σε μια ΟΟ σχεδίαση"





Η μεθοδολογία ανάπτυξης ICONIX

Εισαγωγικά

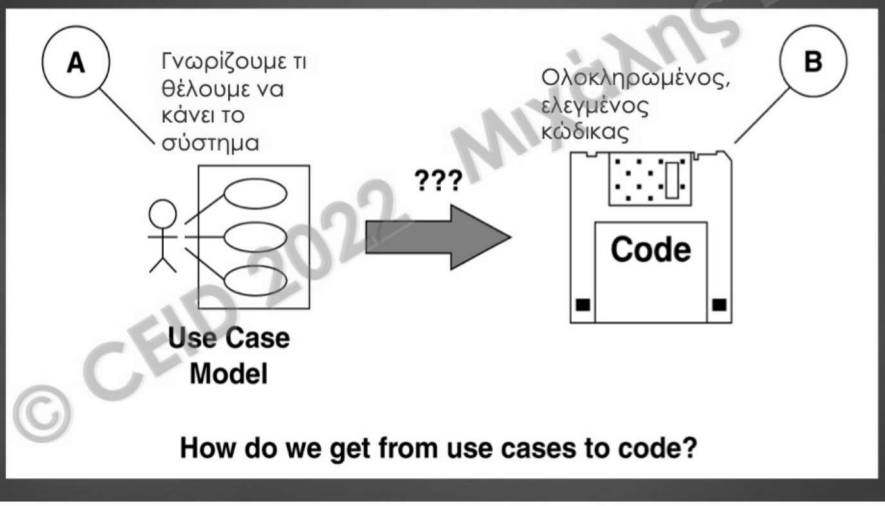
- Βρίσκεται κάπου ανάμεσα στην πολύ μεγάλη Rational Unified Process (RUP) και την πολύ μικρή eXtreme Programming προσέγγιση (XP).
- Αποφεύγει την πληθώρα μοντέλων χωρίς να παραλείπει τις διαδικασίες ανάλυσης και σχεδίασης
- Χρησιμοποιεί τη UML αλλά μόνο τα εντελώς απαραίτητα βήματα αποφεύγοντας το πρόβλημα του analysis paralysis.
- Σε κάθε βήμα έχουμε τη δυνατότητα να ανιχνεύουμε το βαθμό υλοποίησης των απαιτήσεων του χρήστη (δεν απομακρυνόμαστε ποτέ από τις ανάγκες του χρήστη).
- Είναι use case driven (καθοδηγούμενη από τις περιπτώσεις χρήσης):

  "…it results in concrete, specific, readily understandable use cases that the project team can actually use to drive the development effort"

Εισαγωγικά

- Είναι επαυξητική και επαναληπτική μέθοδος
  - Το στατικό μοντέλο εκλεπτύνεται καθώς αναλύεται το δυναμικό
- Χρησιμοποιεί μόνο 4 είδη διαγραμμάτων της UML
  - *Διαδράμματα περιπτώσεων Χρήσης / Use case diagrams* για να αναπαραστήσουμε τα σενάρια χρήσης και του χειριστές του συστήματος
  - *Διαγράμματα κλάσεων / Class diagrams* για να αναπαραστήσουμε το πεδίο του προβλήματος (domain) και τη λεπτομερή στατική δομή του συστήματος
  - *Διαγράμματα Ευρωστίας / Robustness diagrams* για να αναπαραστήσουμε τον τρόπο που η στατική δομή υλοποιεί τα σενάρια χρήσης του συστήματος
  - *Διαγράμματα ακολουθίας / Sequence diagrams* για να συσχετίσουμε λεπτομερώς τη δυναμική συμπεριφορά με τη στατική δομή του συστήματος

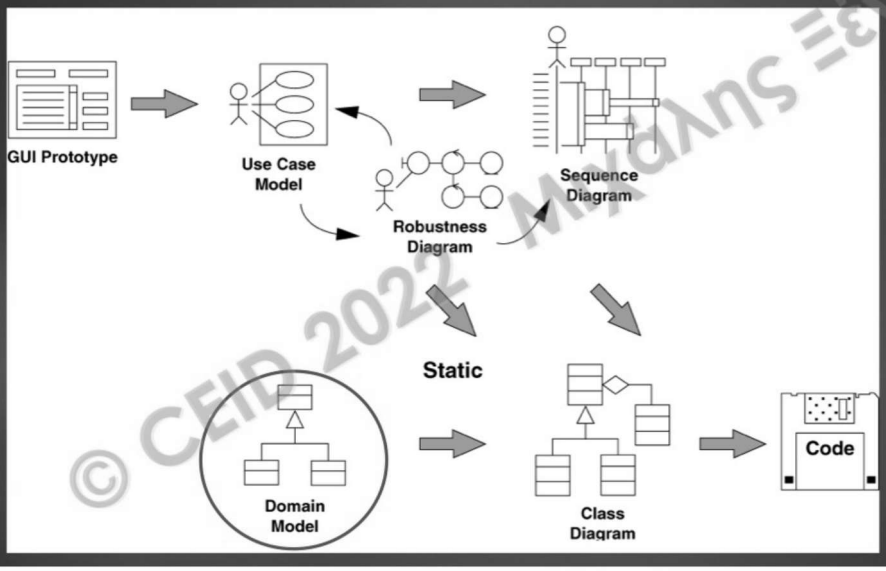
ICONIX Process:
Do OOAD but Keep It Simple



Αλλά δε μπορούμε να σχεδιάσουμε robustness diagrams πριν...

- Έχουμε περιγράψει τη χρήση του συστήματος σε όρους του πεδίου του προβλήματος.
- Δε γράφουμε δηλαδή αόριστες και γενικές περιπτώσεις χρήσης που δεν συνδέονται με το σχεδιασμό του συστήματος. Αντίθετα θέλουμε use cases που αναφέρονται σε ονόματα αντικειμένων από το πεδίο του προβλήματος.
- Άρα χρειάζεται να εντοπίσουμε τις βασικές αφαιρέσεις του πεδίου του προβλήματος, δηλ. ένα **domain model:**
  - Απλοποιημένο class diagram
  - Κατασκευάζεται πριν γράψουμε και σχεδιάσουμε τις περιπτώσεις χρήσης

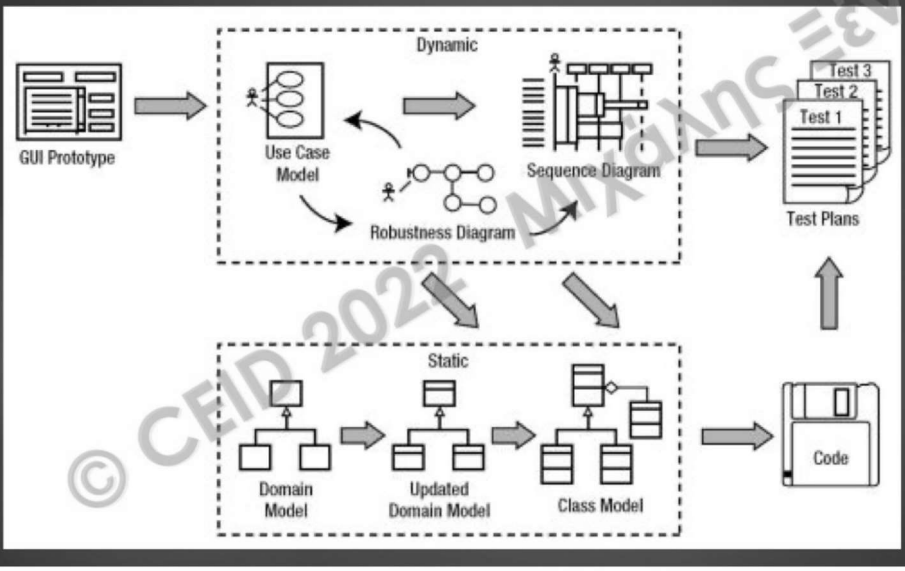
Domain Model



Εκλέπτυνση (refinement) των class diagrams

- Ενώ διερευνούμε τη δυναμική συμπεριφορά του συστήματος με όλο και περισσότερη λεπτομέρεια, θα επιστρέφουμε συχνά στο (στατικό) domain model (αλλιώς γνωστό και ως analysis level class diagrams) και θα το εμπλουτίζουμε με λεπτομέρειες.
- Ο στόχος μας είναι στο τέλος της διαδικασίας να καταλήξουμε σε ένα σύνολο από design-level class diagrams, από τα οποία μπορούμε να περάσουμε απευθείας στην παραγωγή κώδικα.

The ICONIX Process



ICONIX Milestones

4 milestones για ένα project:

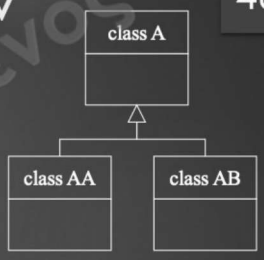
1. Requirements Review
2. Preliminary Design Review
3. Detailed / Critical Design Review
4. Testing and Delivery

- UML Διαγράμματα που χρησιμοποιούνται:
  - Use Case
  - Robustness
  - Sequence
  - **Class**
  - *(Collaboration, Deployment, Component, State, etc.)*

Βήμα 1ο: Ανάλυση απαιτήσεων

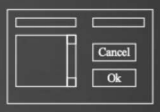
Μοντέλο πεδίου (Domain Modeling)

- Εντοπισμός των οντοτήτων του πραγματικού κόσμου (domain objects) και των σχέσεων περιεκτικότητας και κληρονομικότητας μεταξύ τους.



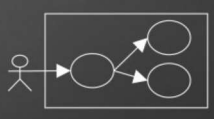
Σχεδίαση GUI / Προτυποποίηση

- Ταχεία προτυποποίηση του προτεινόμενου συστήματος (ο καλύτερος τρόπος για να εντοπίσεις περιπτώσεις χρήσης είναι η χρήση πιθανών οθονών μαζί με user manual, σαν να υπήρχε το πλήρως λειτουργικό σύστημα)



Μοντελοποίηση περιπτώσεων χρήσης

- Εντοπισμός των περιπτώσεων χρήσης (use cases)
- Οργάνωσή τους σε πακέτα (packages).
- Ανάθεση λειτουργικών απαιτήσεων στις περιπτώσεις χρήσης.



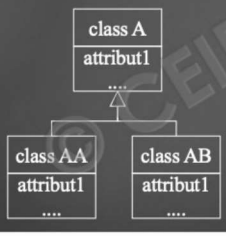
Βήμα 2ο: Ανάλυση, Αρχική Σχεδίαση

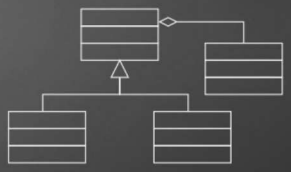
Συγγραφή του κειμένου των περιπτώσεων χρήσης

- Αναλυτική περιγραφή της βασικής ροής κάθε περίπτωσης χρήσης ("sunny day" scenario) καθώς και των τυχόν εναλλακτικών ροών

Robustness Analysis. Για κάθε περίπτωση χρήσης:

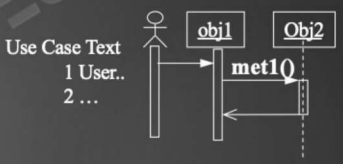
- Εντοπισμός ενός πρώτου συνόλου αντικειμένων των οποίων η συνεργασία επιτυγχάνει το επιλεγμένο σενάριο.
- Αναθεώρηση του μοντέλου του πεδίου προβλήματος με νέες κλάσεις και ιδιότητες.





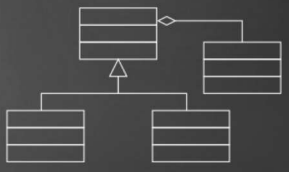
Βήμα 3ο: Αναλυτική Σχεδίαση

Μοντελοποίηση Αλληλεπίδρασης (Ανάθεση συμπεριφοράς)



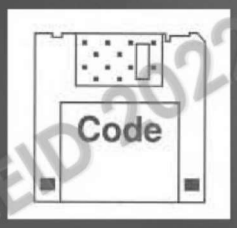
για κάθε Περίπτωση χρήσης:

- Εντοπισμός των μηνυμάτων που πρέπει να ανταλλαχθούν μεταξύ αντικειμένων, των αντικειμένων και των συσχετιζόμενων μεθόδων.
- Σχεδίαση ενός διαγράμματος ακολουθίας (με παράλληλη εξέταση του κειμένου της περίπτωσης χρήσης).
- Συνεχής αναθεώρηση του διαγράμματος κλάσεων με ιδιότητες και λειτουργίες καθώς εντοπίζονται.



Βήμα 4ο: Υλοποίηση

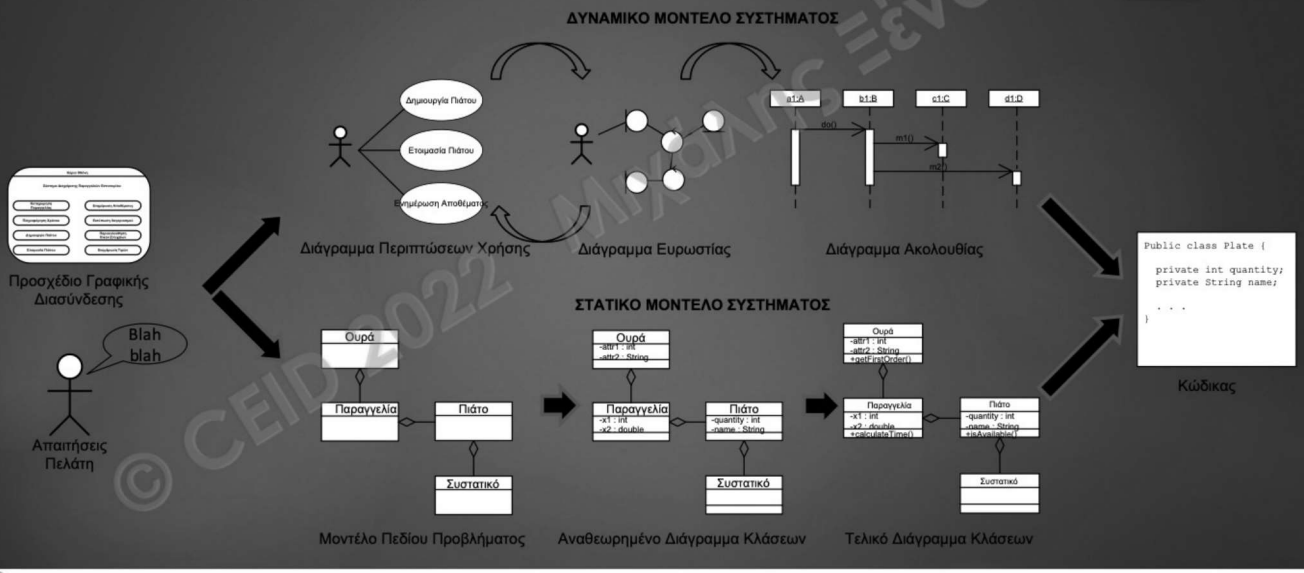
- Αν απαιτείται, παραγωγή διαγραμμάτων ανάπτυξης και συστατικών
- Συγγραφή/παραγωγή κώδικα



Βήμα 5ο: Έλεγχος

- Έλεγχος μονάδων και ολοκλήρωσης (unit and integration testing)
- Έλεγχος συστήματος και αποδοχής (system and user acceptance testing), χρησιμοποιώντας τις περιπτώσεις χρήσης ως περιπτώσεις ελέγχου μαύρου κουτιού (black-box test cases)

Συνοψίζοντας…



Σχεδίαση - Basics!

- **Less is more!**
  - Αρχίζω αδρά και εκλεπτύνω συνεχώς
  - Οι λεπτομέρειες απλά με βασανίζουν
  - Timebox s/w engineering
- **Start when confident**
  - Όχι use cases πριν είναι ξεκάθαρο τι θέλω να κάνει ο χρήστης
  - Όχι DM πριν έχω καταλάβει (διευκρινίσει) την αρχική περιγραφή
- **Η μεθοδολογία δεν είναι δόγμα!**
  - Χρησιμοποιώ ότι με βοηθά μόνο
  - Χρησιμοποιώ ότι επιπλέον χρειάζομαι
- **Oops I forgot the scenario!**