Τεχνολογία λογισμικού στην πράξη
Σχεδιασμός λογισμικού

Διομήδης Σπινέλλης
Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας
Οικονομικό Πανεπιστήμιο Αθηνών

dds@aueb.gr
http://www.dmst.aueb.gr/dds
@CoolSWEng

2024-02-26

Παρουσιάσεις ομάδων

- Επιλέξτε ένα (σημαντικό-δημοφιλές) έργο ανοιχτού λογισμικού και εξετάστε στοιχεία του σχεδιασμού του:
  - αναζητήστε αρχές καλής σχεδίασης λογισμικού
  - αναγνωρίστε θέματα - παράγοντες σχεδίασης που αποκτούν ιδιαίτερη βαρύτητα στο συγκεκριμένο έργο
  - επιλέξτε μια στρατηγική σχεδίασης και ταιριαστή σημειογραφία που θεωρείτε κατάλληλες για να αναπαραστήσετε μια στατική όψη του σχεδίου λογισμικού. Μπορείτε να χρησιμοποιήσετε αφαίρεση, μία από τις αρχές καλής σχεδίασης λογισμικού, για να επικοινωνήσετε εκείνες τις οντότητες που θεωρείτε πιο σημαντικές.

Σχεδίαση λογισμικού

- Design: μια λέξη, δυο έννοιες
- Η διαδικασία: σχεδίαση
- Το αποτέλεσμα της διαδικασίας: σχέδιο
- Περιλαμβάνει
- Αρχιτεκτονικό σχέδιο του συνόλου
- Λεπτομερή σχέδια εφαρμογής των τμημάτων

Διαδικασία

- Προσδιορισμός των ενδιαφερομένων μερών
- Προσδιορισμός των αναγκών τους
- Επιλογή κατάλληλων οπτικών με βάση ένα συγκεκριμένο πλαίσιο αναφοράς (π.χ. 4+1, TOGAF, (GERA))
  - Κάθε οπτική μπορεί να χρησιμοποιεί δικά της μοντέλα και γλώσσα
- Συμπλήρωσή των οπτικών με τρόπο που να διασφαλίζει την μεταξύ τους συνέπεια

Figure 1: College of Architecture and Planning

Παραδοτέα

- Μοντέλα
- Περιγραφές υψηλού επιπέδου
- Τεκμηρίωση
- Αποφάσεις
- Λόγους που λήφθηκαν οι αποφάσεις

Βασικές αρχές

- Αφαίρεση (abstraction)
- διαδικασιών
- δεδομένων
- δομών ελέγχου
- Χαμηλή σύζευξη και συνεκτικότητα (coupling)
- Υψηλή συνεκτικότητα (cohesion)
- Αποσύνθεση και τμηματικότητα (decomposition and modularity)
- Ενθυλάκωση και απόκρυψη πληροφοριών (encapsulation and information hiding)
- Διαχωρισμός της διεπαφής από την υλοποίηση
- Πληρότητα, οικονομία, απλότητα
- Διαχωρισμός ευθυνών (separation of concerns)

Αυξανόμενα επίπεδα συνεκτικότητας

Η καλή σχεδίαση πρέπει να φέρνει κοντά τμήματα που εμφανίζουν υψηλή συνεκτικότητα. Διακρίνουμε τα παρακάτω αυξανόμενα επίπεδα συνεκτικότητας:

- Συμπτωματική συνεκτικότητα (coincidental cohesion)
- Λογική συνεκτικότητα (logical cohesion)
- Χρονική συνεκτικότητα (temporal cohesion)
- Διαδικαστική συνεκτικότητα (procedural cohesion)
- Επικοινωνιακή συνεκτικότητα (communicational cohesion)
- Ακολουθιακή συνεκτικότητα (sequential cohesion)
- Λειτουργική συνεκτικότητα (functional cohesion)

Σύζευξη, από το κακό στο χειρότερο

Η καλή σχεδίαση πρέπει κατά το δυνατόν να αποφεύγει τη σύζευξη μεταξύ τμημάτων. Διακρίνουμε τα παρακάτω αυξανόμενα επίπεδα σύζευξης:

- Σύζευξη δεδομένων (data coupling)
- Σύζευξη δομής δεδομένων (stamp coupling) (ή αντιγράφου ή μήτρας)

- Σύζευξη ελέγχου (control coupling)
- Σύζευξη κοινών δεδομένων (common coupling) (ή από κοινού σύνδεση)
- Σύζευξη εξωτερικών δεδομένων (external coupling)
- Σύζευξη περιεχομένων (content coupling)

Θέματα της σχεδίασης

- Συνδρομή (concurrency)
- Έλεγχος και χειρισμός γεγονότων
- Αποθήκευση των δεδομένων
- Κατανομή των αρθρωμάτων
- Χειρισμός λαθών και εξαιρέσεων
- Βλαβοανοχή
- Τεχνολογία διεπαφής με το χρήστη και εμφάνισης
- Ασφάλεια

Σχεδιαστική οπτική 4+1

Τα σχέδια είναι συχνά από διαφορετικές οπτικές:

- Λογική (βάσει λειτουργικών προδιαγραφών)
- Διεργασίες (βάσει ιδιοτήτων συνδρομής)
- Φυσική
- Σχέδιο ανάπτυξης
- Σενάρια χρήσης

Σχεδιαστικές οπτικές

Τα σχέδια είναι συχνά από διαφορετικές οπτικές:

- Λογική (βάσει λειτουργικών προδιαγραφών)
- Διεργασίες (βάσει ιδιοτήτων συνδρομής)
- Φυσική
- Σχέδιο ανάπτυξης

Αρχιτεκτονικές περιοχές TOGAF

Σύμφωνα με το «The Open Group Architecture Framework» επιχειρησιακές αρχιτεκτονικές καλύπτουν τις εξής περιοχές.

- Επιχείρηση (διεργασίες, δομές, οδηγίες)
- Εφαρμογές (προς ανάπτυξη)
- Δεδομένα (λογικά και φυσικά μοντέλα)
- Τεχνολογία (υπολογιστές, δίκτυα, αποθήκευση)

Αρχιτεκτονικές τεχνικές

- Γενικές
- Επίπεδα
- Σωληνώσεις και φίλτρα
- Μαυροπίνακας
- Κατανεμημένες
- Πελάτη-εξυπηρετητή
- Τριών επιπέδων
- Ομότιμων πελατών
- Εξειδικευμένες
- Μοντέλο - Οπτική - Έλεγχος (MVC)
- Ανάκλαση (reflection)
- Γλώσσα εξειδικευμένου πεδίου (DSL)
- Διερμηνευτής
- Έλεγχος από δεδομένα

Παράδειγμα: επίπεδα

Παράδειγμα: σωληνώσεις και φίλτρα

```
tr -cs A-Za-z '\n' |
tr A-Z a-z |
sort |
uniq |
comm -23 - /usr/dict/words
```

Παράδειγμα: DSL

```
{{Chess diagram small|=
| tright
|
|=
 8 |rd|  |  |qd|  |rd|kd|  |=
 7 |pd|pd|  |  |pd|pd|bd|pd|=
 6 |  |nd|pd|  |  |nd|pd|  |=
 5 |  |  |ql|  |  |  |bl|  |=
 4 |  |  |  |pl|pl|  |bd|  |=
 3 |  |  |nl|  |  |nl|  |  |=
 2 |pl|pl|  |  |  |pl|pl|pl|=
 1 |  |  |  |rl|kl|bl|  |rl|=
    a  b  c  d  e  f  g  h
| The position after 11. Bg5.
}}
```

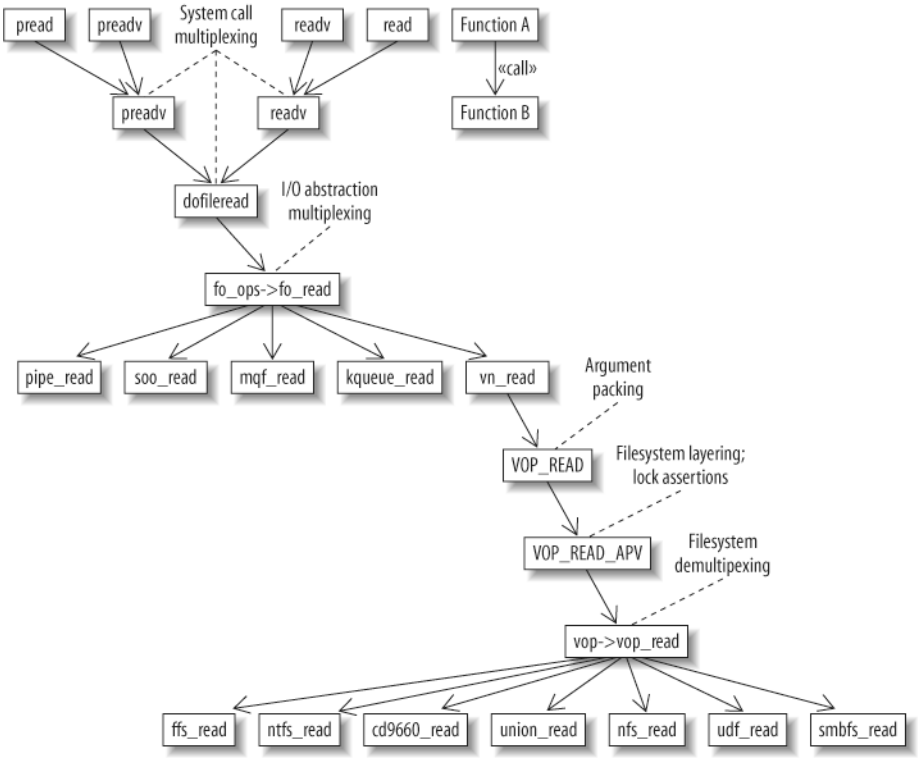


Figure 2: Layers of indirection in the FreeBSD implementation of the read system call

Παράδειγμα: Αποτέλεσμα DSL

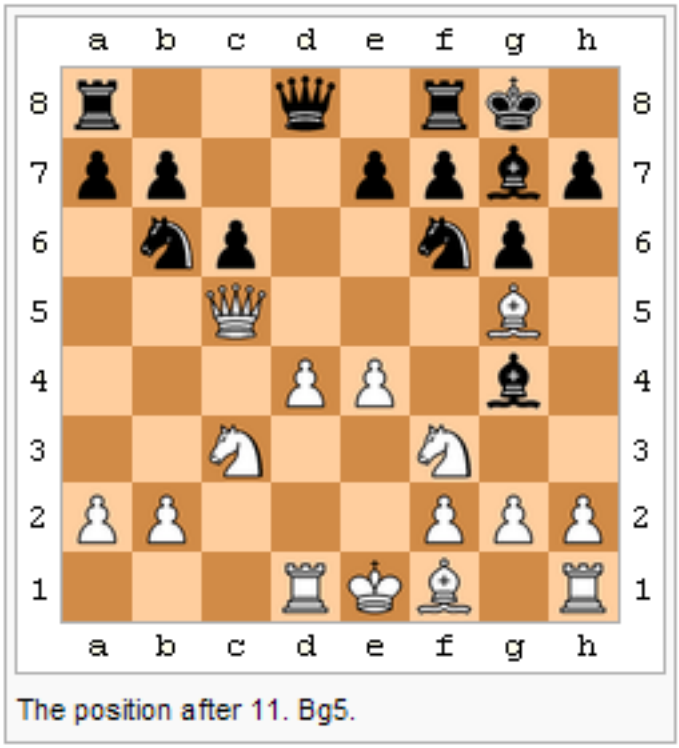


Figure 3: Chess board

Σχεδιαστικά πρότυπα

- Δημιουργίας: Builder, Factory, Prototype, Singleton
- Δομικά: Adapter, Bridge, Composite, Decorator, Facade, Flighweight Proxy
- Συμπεριφορικά: Command, Intepreter, Iterator, Mediator

Παράδειγμα: Factory

(Πηγή: Wikipedia)

Παράδειγμα: Observer

(Πηγή: Wikipedia)

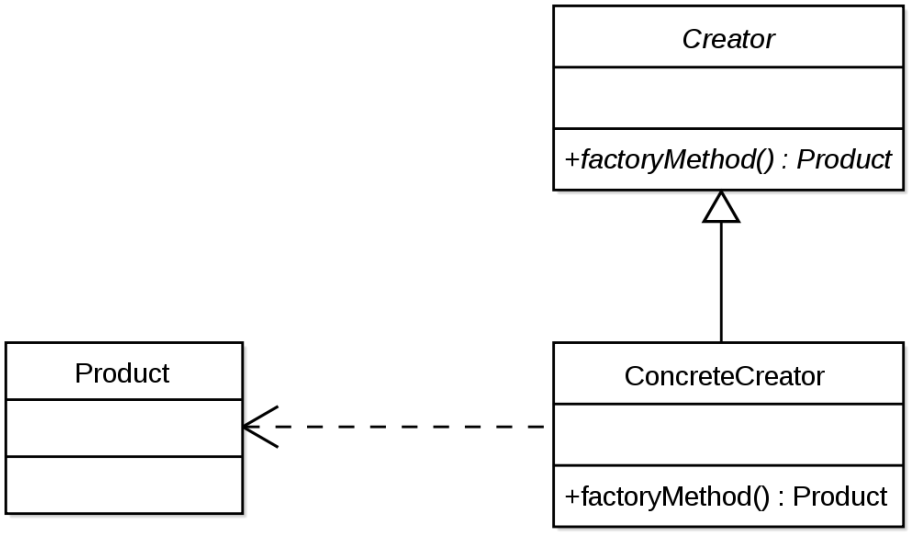


Figure 4: Factory

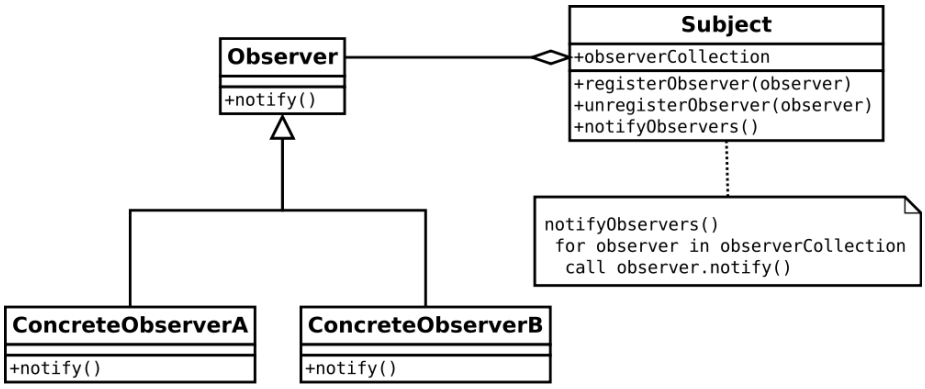


Figure 5: Observer

Παράδειγμα: Decorator

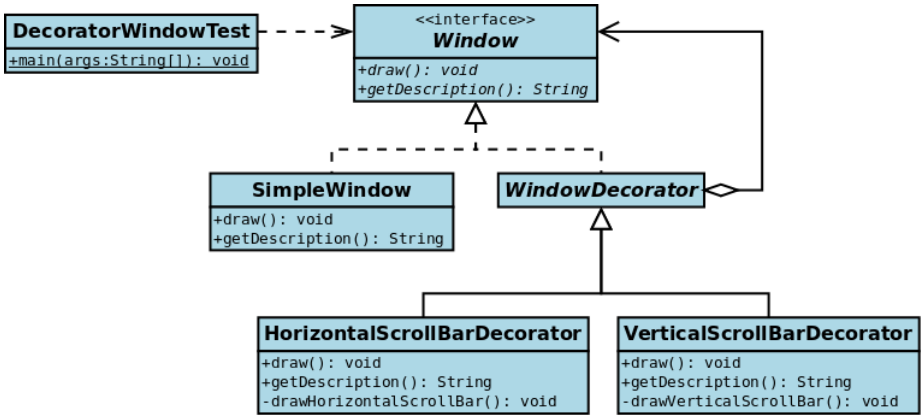


Figure 6: Decorator

(Πηγή: Wikipedia)

Ικανοποίηση διαφορετικών χρηστών

- Οικογένειες προγραμμάτων
- Λογισμικό-πλαίσιο (framework)

Αρχές διεπαφής χρήστη

- Εύκολη εκμάθηση
- Υιοθέτηση γνωστών τρόπων διεπαφής
- Συνέπεια
- Όχι εκπλήξεις (POLA)
- Ανακτησιμότητα
- Καθοδήγηση του χρήστη
- Μέριμνα για όλους τους χρήστες

Τρόποι διεπαφής

- Ερώτηση - απάντηση
- Άμεσος χειρισμός
- Επιλογή από μενου
- Συμπλήρωση φόρμας
- Γλώσσα εντολών
- Φυσική γλώσσα

Σημαντικά θέματα διεπαφής

- Σύντομος χρόνος απόκρισης
- Ενημέρωση του χρόνου απόκρισης
- Προσοχή στη χρήση των χρωμάτων
- Διεθνοποίηση και τοπικοποίηση
- Χρήση μεταφοράς

Παράσταση δομής

- Διάγραμμα κλάσεων (UML class)
- Διάγραμμα αντικειμένων (UML object)
- Διάγραμμα εξαρτημάτων (UML component)
- Διάγραμμα διάπτυξης (UML deployment)
- Διάγραμμα οντοτήτων συσχετίσεων (ERD)
- Κάρτες «class, responsibility, collaborator» (CRC)

Διαγράμματα UML

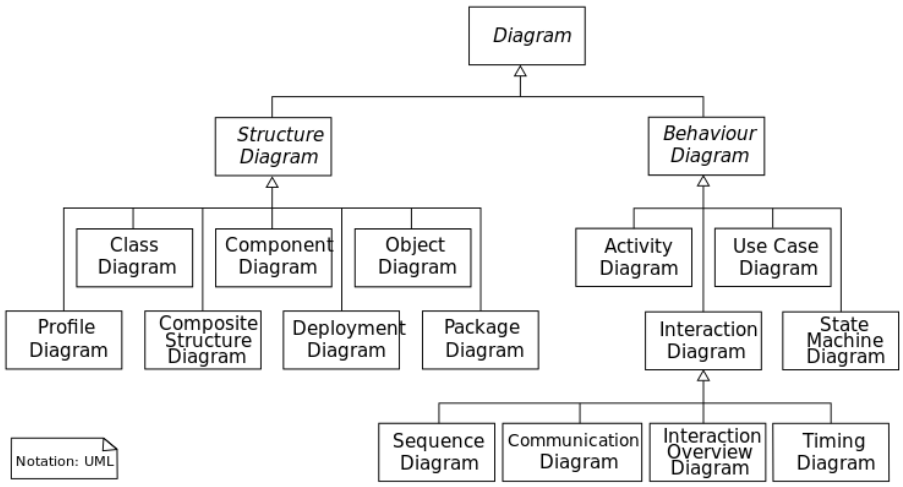


Figure 7: Διαγράμματα UML

(Πηγή: Wikipedia)

Σχέσεις UML

Στη UML ορίζονται τέσσερεις βασικές σχέσεις:

- εξάρτηση (dependency)
- γενίκευση (generalisation)

- σύνδεση (association)
- υλοποίηση (realisation)

Εξάρτηση

Η εξάρτηση δηλώνει πως μια αλλαγή σε μιαν οντότητα θα επηρεάσει μιαν άλλη αλλά όχι απαραίτητα και το αντίστροφο. Παριστάνεται με μια διακεκομμένη γραμμή με ανοιχτό βέλος που δείχνει προς την οντότητα που υπάρχει εξάρτηση:

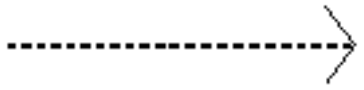


Figure 8: UML Dependency

Γενίκευση

Η γενίκευση δηλώνει μια σχέση ανάμεσα σε κάτι γενικό (τη βασική κλάση ή αλλιώς γονέα) και κάτι ειδικό (μιαν υποκλάση ή αλλιώς παιδί της). Παριστάνεται με μια συνεχή γραμμή με κλειστό βέλος που δείχνει προς τη βασική κλάση:

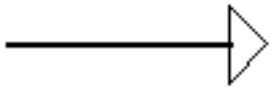


Figure 9: UML Dependency

Σύνδεση

- Η σύνδεση αναφέρεται σε αντικείμενα τα οποία συνδέονται με κάποιο τρόπο με άλλα. Όταν δύο κλάσεις είναι συνδεδεμένες μπορεί κανείς να μεταβεί από αντικείμενα της μιας σε αντικείμενα της άλλης. Η σύνδεση παριστάνεται με μια ευθεία γραμμή ανάμεσα στα δύο αντικείμενα.

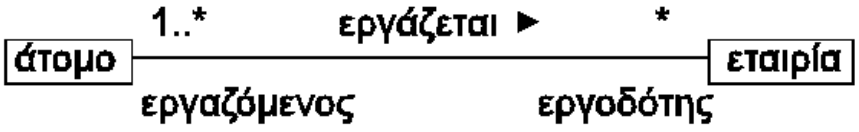


Figure 10: UML Association

Συγκρότημα και σύνδεση

- Αν σε μια σχέση τα αντικείμενα απαρτίζουν τμήματα ενός όλου, τότε αυτή απεικονίζεται ως συγκρότημα (aggregation) με την παράσταση ενός

διαμαντιού στην άκρη του “όλου”.

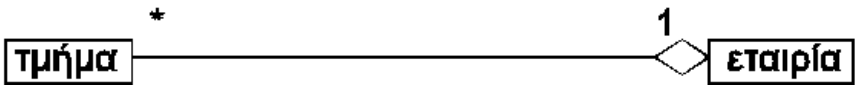
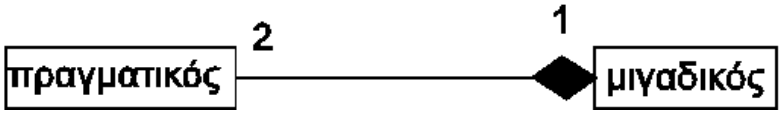


Figure 11: UML Aggregation

- Αν σχέση τα αντικείμενα που απαρτίζουν τμήματα ενός όλου έχουν την ίδια διάρκεια ζωής με το όλο, τότε αυτή απεικονίζεται ως σύνθεση (composition) με την παράσταση ενός γεμάτου διαμαντιού στην άκρη του “όλου”.



Πληροφορίες στη σύνδεση

- Αν η σύνδεση δεν είναι αμφίδρομη τότε η κατεύθυνσή της μπορεί να οριστεί με ένα ανοιχτό βέλος.
- Το όνομα της σύνδεσης μπορεί να γραφεί πάνω από τη γραμμή, ενώ η κατεύθυνση του ονόματος ορίζεται από ένα βέλος πλάι στο όνομα.
- Ο ρόλος των οντοτήτων που συνδέονται προσδιορίζεται από ένα όνομα στην κάθε άκρη της γραμμής.
- Ο αριθμός που δηλώνει πόσα αντικείμενα αντιστοιχούν σε κάθε αντικείμενο στην άλλη άκρη της σχέσης (πολλαπλότητα (multiplicity)) δηλώνεται από έναν αριθμό (π.χ. 3), ή μια περιοχή αριθμών (π.χ. 1..* για ένα έως πολλά) πάνω από την αντίστοιχη άκρη της γραμμής.
- Ένα X πάνω σε μια άκρη της γραμμής δηλώνει πως δεν προσφέρεται μετάβαση (navigation) προς τη συγκεκριμένη κατεύθυνση.

Υλοποίηση

Η υλοποίηση δηλώνει πως ο εξυπηρετούμενος (αυτός που βρίσκεται στην ουρά του βέλους) υποστηρίζει τη διεπαφή (τουλάχιστον όλες τις πράξεις) που ορίζονται από τον παροχέα (αυτόν που βρίσκεται στην κεφαλή του βέλους):

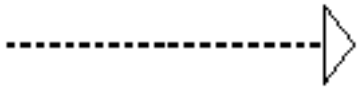


Figure 12: UML Realization

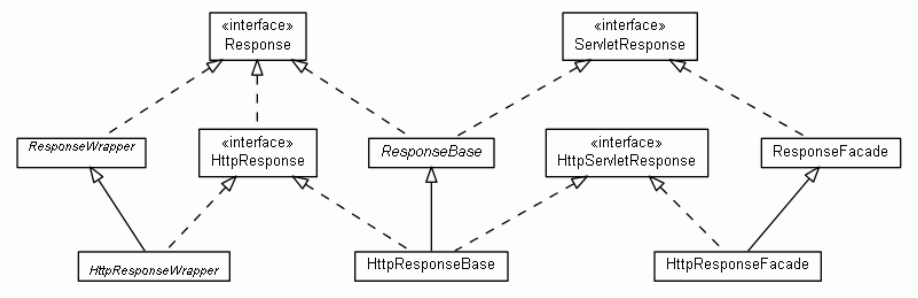


Figure 13: UML class diagram

UML: Διάγραμμα κλάσεων

UML: Διάγραμμα διάπτυξης

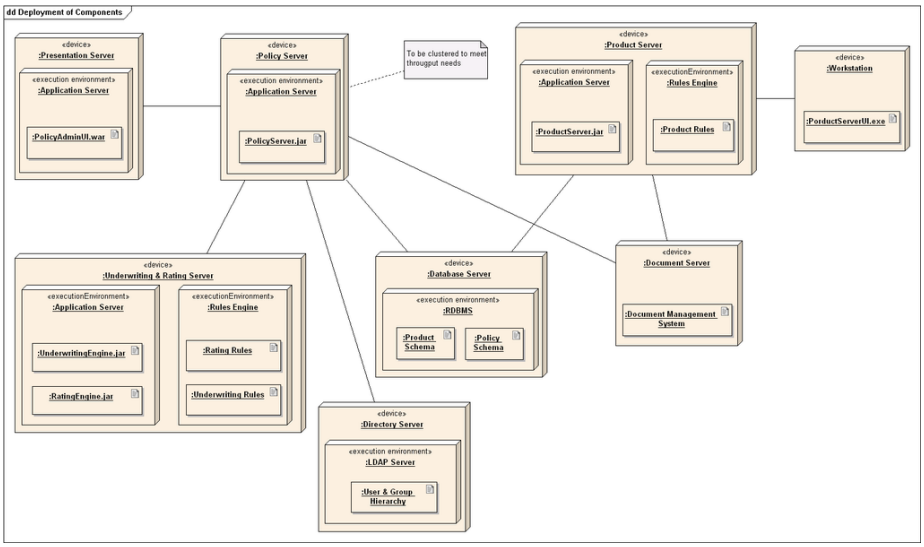


Figure 14: UML

(Πηγή: Wikipedia)

Διάγραμμα οντοτήτων συσχετίσεων

(Πηγή: Χρήστος Παπαδουλής)

UML: Διάγραμμα αρθρωμάτων

(Πηγή: Wikipedia)

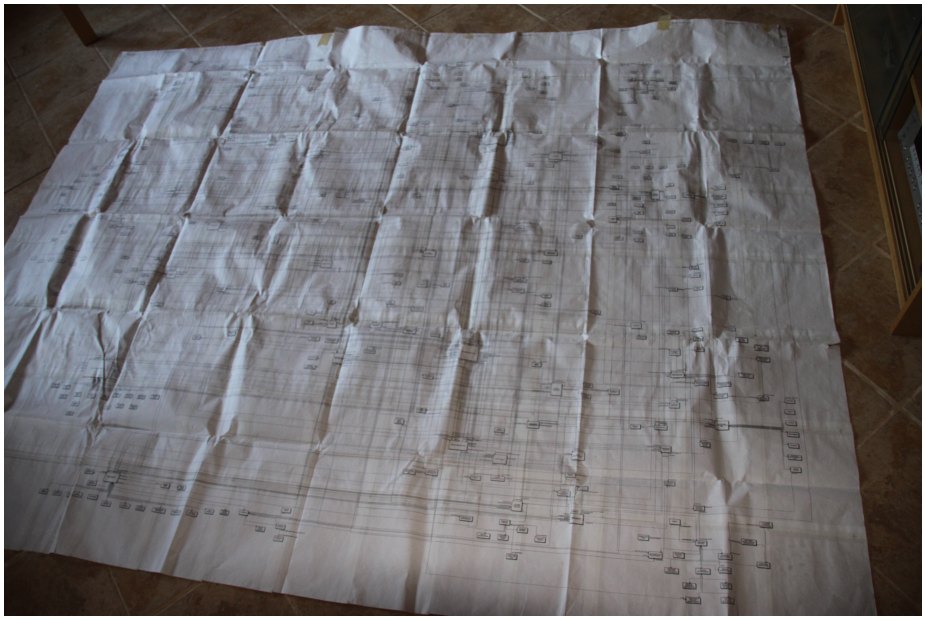


Figure 15: ERD

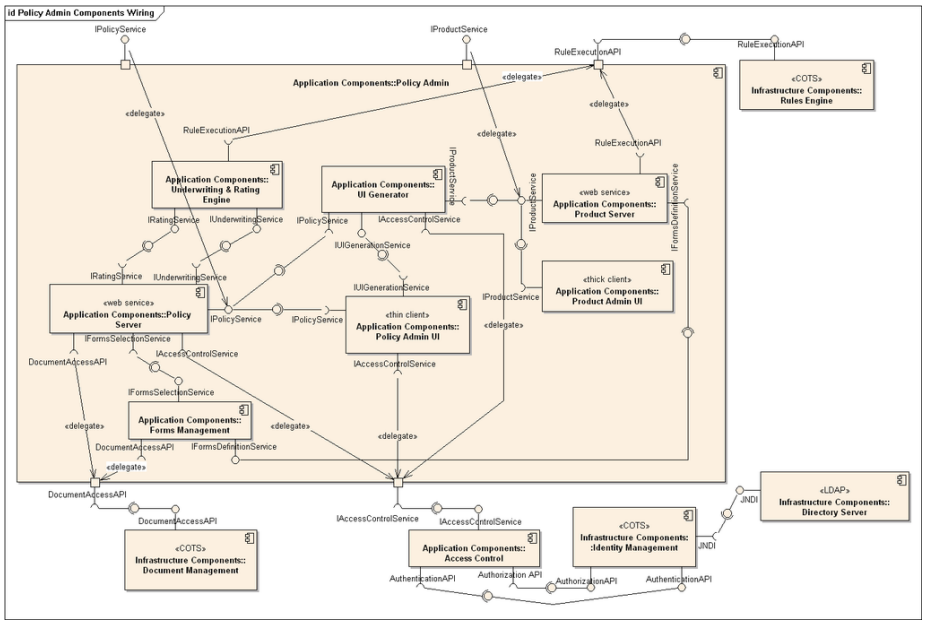


Figure 16: UML

UML: Διάγραμμα πακέτων

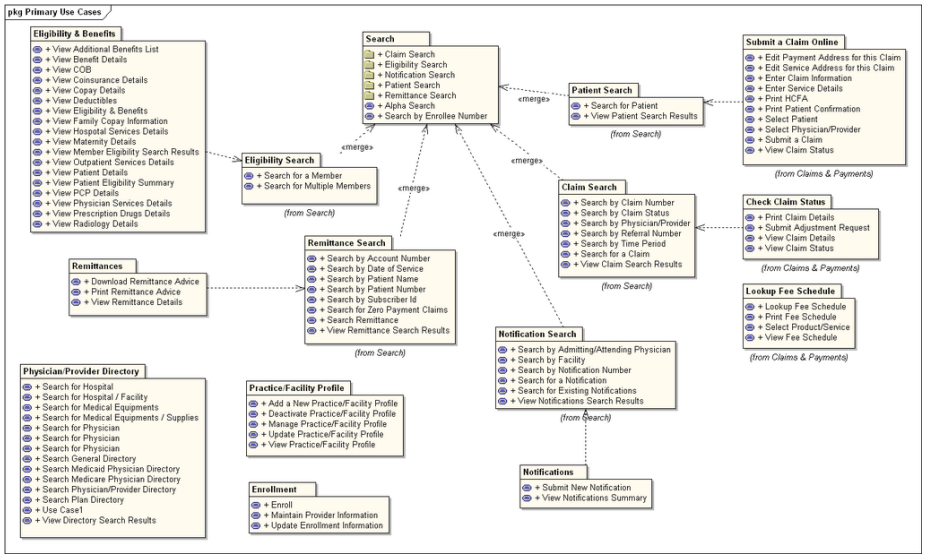


Figure 17: UML

(Πηγή: Wikipedia)

Παράσταση συμπεριφοράς

- Διάγραμμα δραστηριοτήτων (UML activity)
- Διάγραμμα καταστάσεων (UML statechart)
- Διάγραμμα ροής δεδομένων (data flow)
- Πίνακας αποφάσεων (decision table)
- Διάγραμμα ροής (flow chart)
- Διάγραμμα ακολουθίας (UML sequence)
- Διάγραμμα καταστάσεων (UML state)
- Διάγραμμα συνεργασίας (UML collaboration)

UML: Διάγραμμα καταστάσεων

UML: Διάγραμμα ακολουθίας

Εργαλεία και πηγές

- How to draw beautiful software architecture diagrams
- Community list of comparisons between Text to Diagram tools

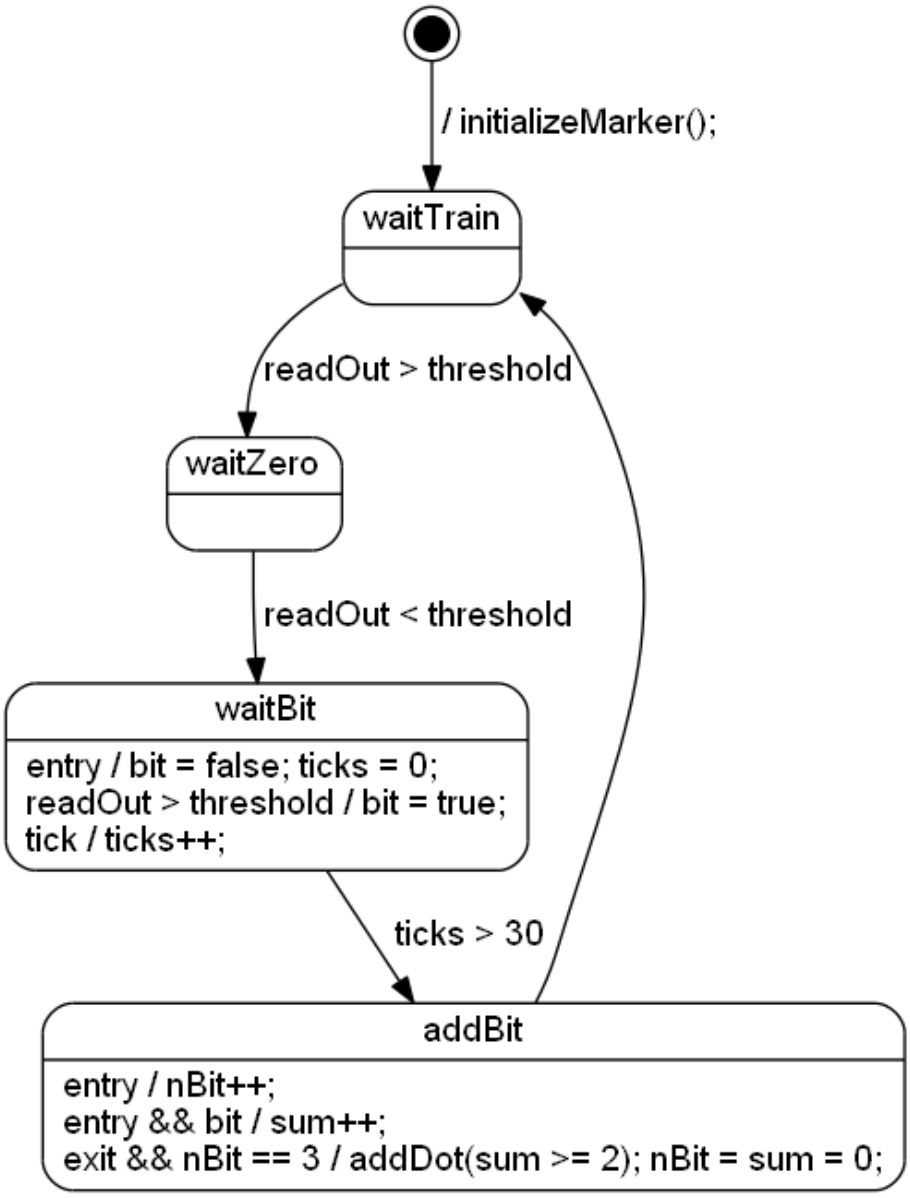


Figure 18: UML

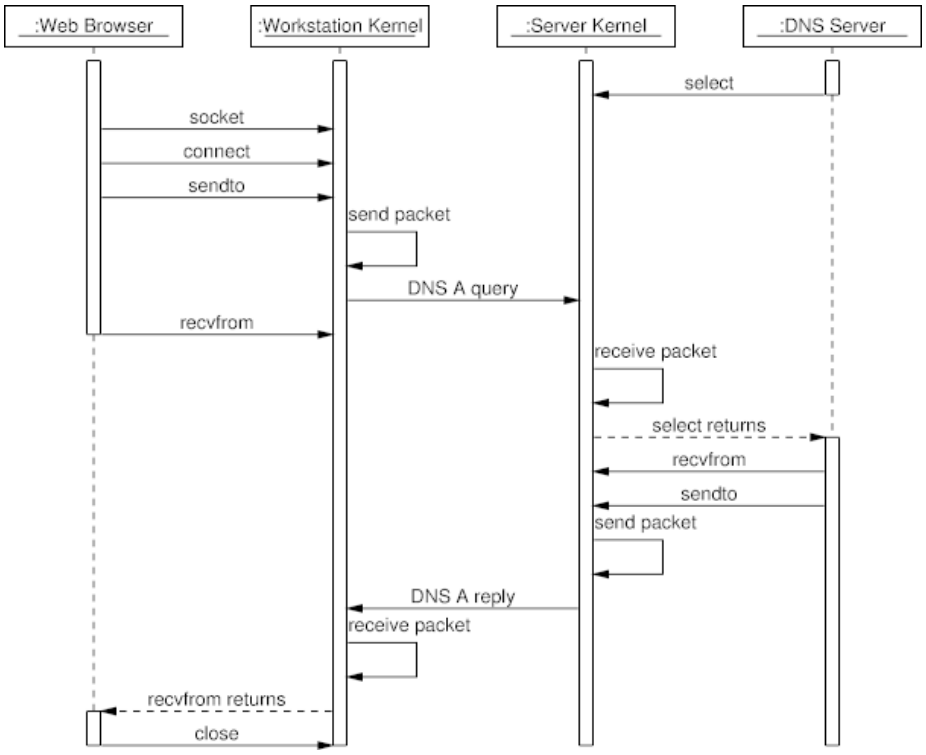


Figure 19: UML

Προετοιμασία για το επόμενο μάθημα (1)

- Διαβάστε το κεφάλαιο 3 του SWEBOK v 3.0
- Άσκηση (Κατασκευή λογισμικού)
- Αξιολογήστε στοιχεία-ιδιότητες ενός σημαντικού-δημοφιλούς έργου ανοιχτού λογισμικού που αφορούν την κατασκευή του:
- Αναζητήστε πρότυπα κατασκευής. Πως αυτά διευκολύνουν την επιβεβαίωση της ορθότητας του λογισμικού;
- Αναζητήστε επαναχρησιμοποιήσιμα και επαναχρησιμοποιούμενα συστατικά λογισμικού.
- Αναζητήστε πρακτικές ελέγχου ποιότητας κώδικα.
- Επιχειρήστε να βελτιώσετε την ποιότητα κατασκευής του έργου σε μία περίπτωση συνεισφέροντας σε αυτό την αλλαγή που προτείνετε.

Προετοιμασία για το επόμενο μάθημα (2)

- Βίντεο (Σχεδιασμός Προγραμματιστικής Διεπαφής Εφαρμογών (API)) https://www.youtube.com/watch?v=aAb7hSCtvGw

##Συμβουλές για την άσκηση: * Αναγνωρίστε και εστιάστε σε συστατικά λογισμικού που υλοποιούν τον πυρήνα της λειτουργικότητας του συστήματος. * Αναζητήστε πρότυπα και καλές πρακτικές κατασκευής λογισμικού. * Κατανοήστε τον όρο drive-by commit. * Χρησιμοποιήστε την υπηρεσία Github. Δείτε εδώ πώς.