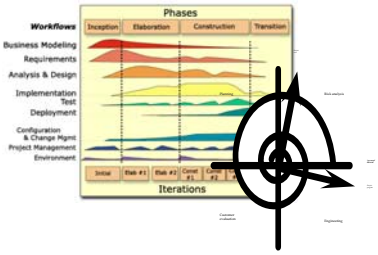
HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Γενικές Μεθοδολογίες Ανάπτυξης Λογισμικού



Γιάννης Τζίτζικας

Διάλεξη : 4
Ημερομηνία :

Διάρθρωση

- Στο τέλος του 2ου μαθήματος σχολιάσαμε για λίγο μεθοδολογίες ανάλυσης και σχεδίαση Πληροφοριακών Συστημάτων
  - (process-centered, data-centered, object-oriented)
- Σήμερα θα εστιάσουμε σε Μεθοδολογίες Ανάπτυξης Λογισμικού (γενικά)

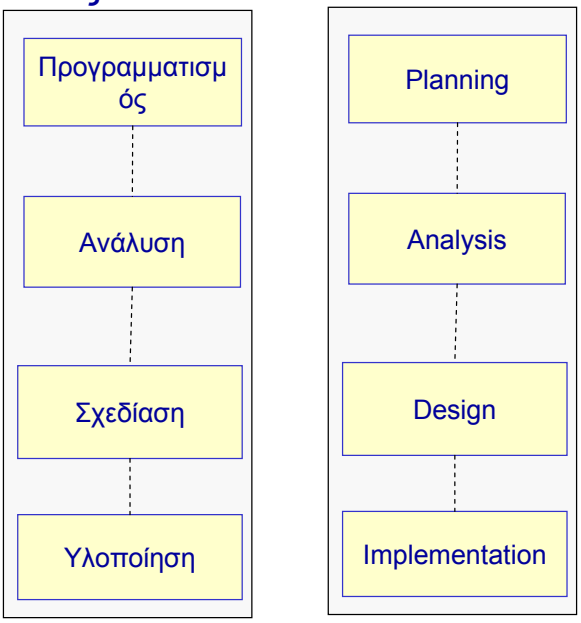
- Μεθοδολογίες Ανάπτυξης Λογισμικού
- Οι 4 θεμελιώδεις φάσεις
- Παραδείγματα Μεθοδολογιών
- Πως επιλέγουμε μια Μεθοδολογία
- Λίγα λόγια για την Μοντελοδηγούμενη Αρχιτεκτονική (MDA)
- Μοντέλα Βελτίωσης Διαδικασίας (Process Improvement Models)

Οι 4 θεμελιώδεις φάσεις

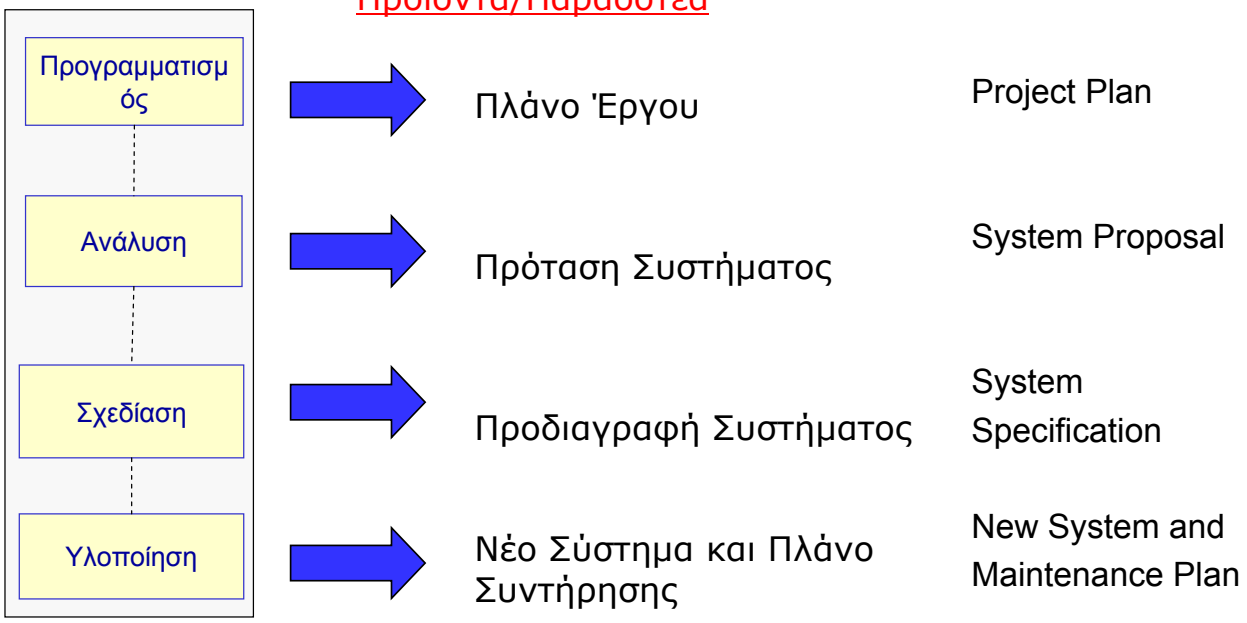
Οι 4 θεμελιώδεις φάσεις (προγραμματισμός, ανάλυση, σχεδίαση, υλοποίηση)

Σε κάθε έργο και κάθε μεθοδολογία ανάπτυξης μπορούμε να εντοπίσουμε κάποιες θεμελιώδεις φάσεις

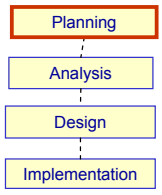
- Κάθε φάση αποτελείται από βήματα (steps), βασίζεται σε τεχνικές (techniques) και έχει παραδοτέα (deliverables)
- Διαφορετικές μεθοδολογίες ορίζουν διαφορετικά βήματα ή διαφορετική σειρά βημάτων



Οι 4 θεμελιώδεις φάσεις (προγραμματισμός, ανάλυση, σχεδίαση, υλοποίηση)



Προγραμματισμός (Planning)



Στόχος της είναι η απάντηση των ερωτημάτων:

- Γιατί και πως θα κατασκευαστεί το σύστημα;

Χωρίζεται σε δυο «υποφάσεις»:

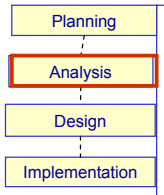
(A) **Έναρξη Έργου**

- ***(την οποία συζητήσαμε στο προηγούμενο μάθημα)***
- Παραδοτέο: Μελέτη Σκοπιμότητας

(B) Αν το έργο εγκριθεί, τότε μπαίνει στη φάση της **Διαχείρισης Έργου** (Project Management) η οποία διαρκεί όσο και το έργο.

- Ο **συντονιστής του έργου** (project management) δημιουργεί και ενημερώνει το πλάνο εργασίας (work plan), στελεχώνει το έργο, και παρακολουθεί και ελέγχει την πρόοδο του
- Παραδοτέο: **Πλάνο Έργου** (project plan)

Ανάλυση (Analysis)



Στόχος της είναι η απάντηση των ερωτημάτων:

- Ποιος θα χρησιμοποιήσει το σύστημα;
- Τι θα κάνει το σύστημα;
- Πού και πότε θα χρησιμοποιείται το σύστημα;

Βήματα

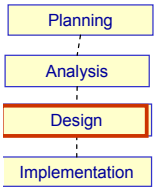
1. Συλλογή Απαιτήσεων: (μέσω συνεντεύξεων, ερωτηματολογίων, κλπ). Ανάλυση του υπάρχοντος συστήματος και των προβλημάτων του και περιγραφή του τρόπου λειτουργίας του νέου συστήματος

2. Μοντελοποίηση βασικών πλευρών του συστήματος

Αποτέλεσμα/Παραδοτέα:

Έγγραφο Περιγραφής Απαιτήσεων, Μοντελοποίηση διαφόρων απόψεων του συστήματος

Σχεδιασμός (Design)



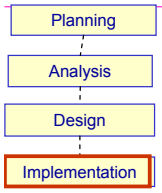
Βήματα:

1. Στρατηγική Σχεδιασμού (Design Strategy). Ανάπτυξη από τον **ίδιο τον οργανισμό**, ή **εξωτερική ανάθεση** (outsourcing) σε μια άλλη εταιρεία, ή **αγορά ενός υπάρχοντος πακέτου λογισμικού**;
2. Αρχιτεκτονικός Σχεδιασμός (Architecture Design): υλικό, λογισμικό, δικτυακή υποδομή, διεπαφή χρήστη, φόρμες, αναφορές, βάσεις δεδομένων, κλπ.
3. Περιγραφή Αρχείων και Βάσεων Δεδομένων : ποια ακριβώς δεδομένα θα αποθηκεύονται και πού;
4. Σχεδιασμός των Προγραμμάτων : ποια προγράμματα πρέπει να γραφτούν και τι θα κάνει το κάθε ένα;

Αποτέλεσμα/Παραδοτέα:

Προδιαγραφή Συστήματος (System Specification) η οποία δίδεται στην ομάδα των προγραμματιστών προς υλοποίηση

Υλοποίηση (Implementation)



Συνήθως είναι το πιο χρονοβόρο και ακριβό τμήμα της διαδικασίας

Βήματα

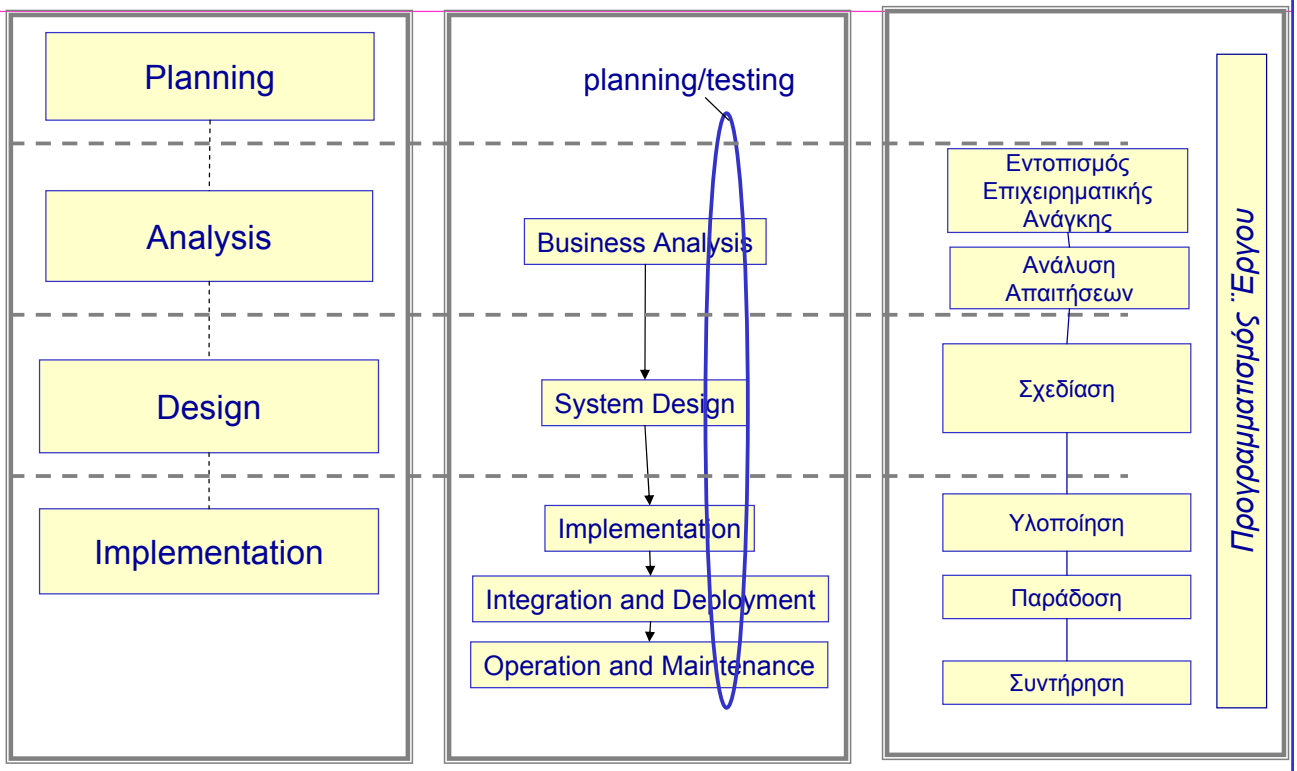
1. Κατασκευή (Construction)

2. Εγκατάσταση (Installation) και Μετάβαση (Transition)

   Το νέο σύστημα αντικαθιστά το υπάρχον (είτε εξ' ολοκλήρου, είτε μεσολαβεί διάστημα παράλληλης λειτουργίας των δύο συστημάτων, ή σε εγκατάσταση/λειτουργία σε φάσεις), πλάνο εκπαίδευσης.

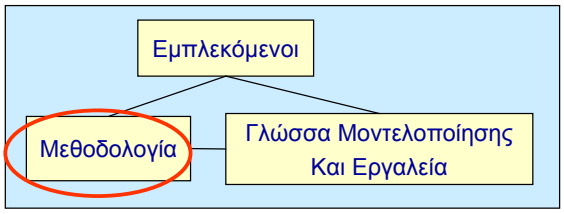
3. Πλάνο Υποστήριξης (Support Plan): Τυπική ή άτυπη αξιολόγηση του συστήματος μετά την εγκατάσταση, και συστηματικός τρόπος εντοπισμού και καταγραφής των αλλαγών (ή βελτιώσεων) που πρέπει να γίνουν.

Διαφορετικά ονόματα και σχηματικές απεικονίσεις των ιδίων εννοιών



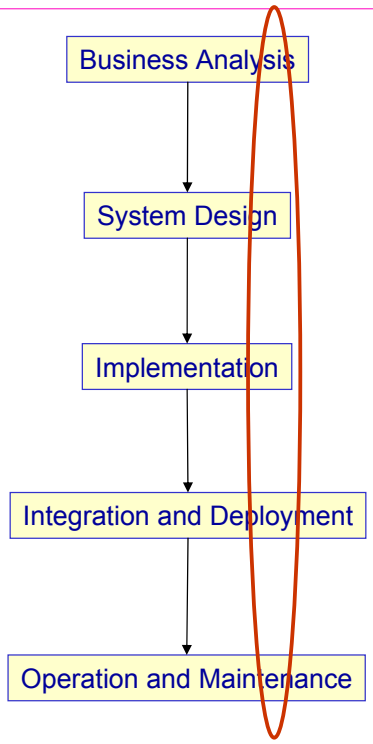
Πως οι προηγούμενες θεμελιώδεις φάσεις σχετίζονται με την έννοια της μεθοδολογίας;

Μεθοδολογία Ανάπτυξης Λογισμικού



- Ορίζει και οργανώνει τις δραστηριότητες παραγωγής και συντήρησης λογισμικού
- Μια μεθοδολογία (ή αλλιώς μοντέλο διαδικασίας (process model)):
  - Ορίζει τη σειρά των εργασιών και δραστηριοτήτων
  - Καθορίζει ποια τεχνουργήματα (artifacts) πρέπει να παραδοθούν και πότε
  - Αναθέτει εργασίες και τεχνουργήματα στους κατασκευαστές
  - Προσφέρει κριτήρια για την παρακολούθηση και μέτρηση της προόδου του έργου.
- Δεν επιδέχεται αυστηρής τυποποίησης/αυτοματοποίησης

Πως οι προηγούμενες φάσεις σχετίζονται με την έννοια της μεθοδολογίας;



Πως ακριβώς θα εργαστούμε;
- *Ποια βήματα;*
- *Με ποια σειρά;*
- *Ποια τα παραδοτέα και για πότε;*

Μια **Μεθοδολογία** μας προτείνει έναν τρόπο εργασίας (και απαντά στα παραπάνω ερωτήματα)

Μεθοδολογίες (ή Μοντέλα Ανάπτυξης, Μοντέλα Κύκλου Ζωής)
(or Development Models, or Lifecycle models)

Famous joke:
- What's the difference between a methodologist and a terrorist?

- You can negotiate with a terrorist.

Γενικές Μεθοδολογίες Ανάπτυξης Λογισμικού

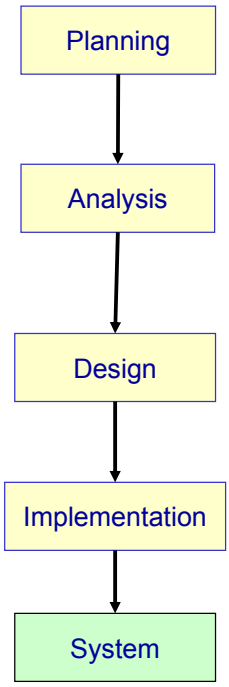
Μια κατηγοριοποίηση Μεθοδολογιών

- *Απουσία μεθοδολογίας* (code-and-fix)
- *Δομημένη Σχεδίαση (structured design)*
  - Καταρράκτη (Waterfall), Παράλληλη (Parallel)
- *Εξελικτικές / Ταχείας Ανάπτυξης Εφαρμογής (evolutionary / rapid application development (RAD))*
  - Πολυφασική (Phased), Πρωτοτυποποίηση (Prototyping), Throwaway prototyping
    - RUP (Rational Unified Process)
- *Εύκαμπτη Ανάπτυξη (Agile Development)*
  - **XP (eXtreme Programming)**
- *Υλοποίηση μέσω επαναχρησιμοποίησης (By reuse)*
- Άλλες :
  - Τυπικοί Μετασχηματισμού (Transformational)
  - Μοντελο-οδηγούμενη (MDA: Model Driven Architecture)

Μεθοδολογίες Δομημένης Σχεδίασης

- Καταρράκτη
- Παράλληλη

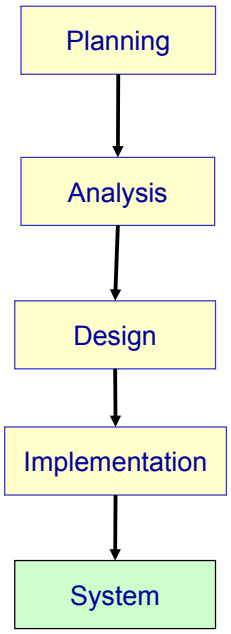
Μεθοδολογίες Δομημένης Σχεδίασης (Structured Design Methodologies)
Το μοντέλο του Καταρράκτη (Waterfall)



Χαρακτηριστικά
- Γραμμική ακολουθία φάσεων
- Μακροσκελείς αναφορές

Βασικές αρχές μοντέλου Καταρράκτη:
- Ακολουθία σαφώς καθορισμένων βημάτων
- Κάθε βήμα καταλήγει στην δημιουργία προϊόντος (έγγραφο ή κώδικας)
- Κάθε προϊόν αποτελεί τη βάση για το επόμενο βήμα
- Η ορθότητα κάθε προϊόντος μπορεί να ελεγχθεί.

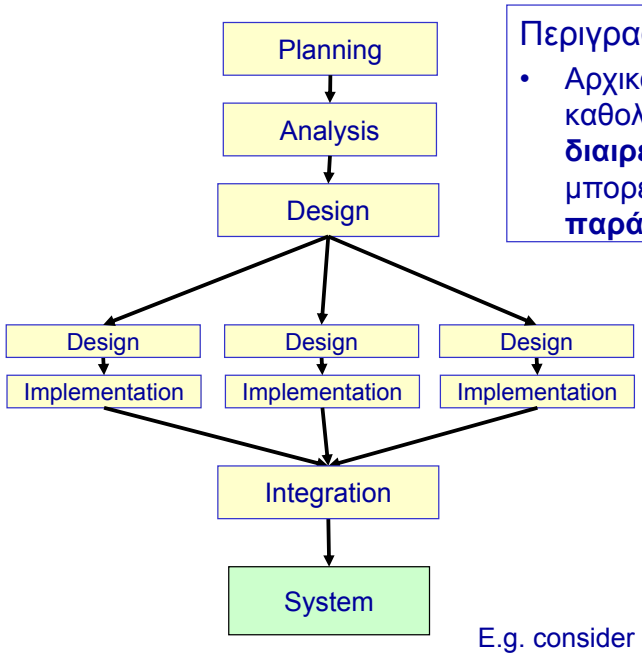
Μεθοδολογίες Δομημένης Σχεδίασης (Structured Design Methodologies)
Το μοντέλο του Καταρράκτη (Waterfall)



Πλεονεκτήματα
- Διαχωρισμός του έργου σε απλούστερες φάσεις
- Κάθε φάση παράγει ένα σαφώς καθορισμένο παραδοτέο

Μειονεκτήματα
- Στην πράξη οι φάσεις αλληλεπικαλύπτονται
- Στην πράξη το μοντέλο δεν είναι γραμμικό: συχνά επιστρέφουμε στην προηγούμενη φάση
- Συχνά, αλλαγές σε κάποιο στάδιο επιβάλλουν την οπισθοχώρηση και πραγματοποίηση αλλαγών σε πολλά από τα προηγούμενα στάδια
- Η σχεδίαση πρέπει να ολοκληρωθεί πριν την έναρξη της υλοποίησης
- Σαν να θεωρεί ότι όλα θα γίνουν εκ του μηδενός – δεν εκμεταλλεύεται την έννοια της αναχρησιμοποίησης (reuse)
- *Ο πελάτης βλέπει τι τελικά αγοράζει πολύ αργά*
- Μακροσκελείς αναφορές που δεν βοηθούν την επικοινωνία

Μεθοδολογίες Δομημένης Σχεδίασης (Structured Design Methodologies)
Το Παράλληλο μοντέλο (Parallel)



Περιγραφή
- Αρχικά γίνεται ένας **γενικός σχεδιασμός** του καθολικού συστήματος και κατόπιν το έργο **διαιρείται σε υποέργα** όπου κάθε ένα από αυτά μπορεί να σχεδιαστεί και να υλοποιηθεί **παράλληλα**

Πλεονεκτήματα
- **Μειωμένος χρόνος** ανάπτυξης έργου

Μειονεκτήματα
- Πολλές φορές τα υποέργα **δεν είναι εντελώς ανεξάρτητα**

E.g. consider a partition to two subprojects:
* one with a long design phase, but short implementation phase
* one with a short design phase, but long implementation phase

Εξελικτικές Μεθοδολογίες

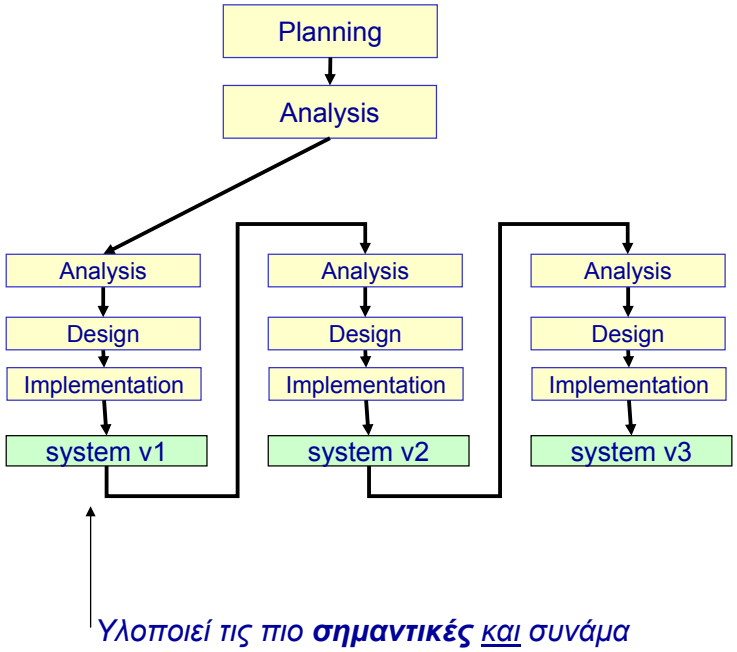
- Πολυφασική
- Πρωτοτυποποίηση

Εξελικτικές (/Ταχείας Ανάπτυξης) Μεθοδολογίες
Evolutionary / Rapid Application Development (RAD) Methodologies

Κύρια σημεία

- Η περάτωση κάποιων τμημάτων του συστήματος πολύ γρήγορα
- Προτείνεται η διενέργεια της ανάλυσης και της σχεδίασης με ειδικές τεχνικές και εργαλεία ώστε να επιταχυνθεί η όλη διαδικασία
  - Εργαλεία CASE
  - Joint application design (JAD)
  - Γλώσσες Οπτικού Προγραμματισμού (visual programming languages, e.g. VBasic)
  - Γεννήτριες κώδικα (code generators) που να παράγουν κώδικα από το σχέδιο (produce code from design)

Πολυφασική Ανάπτυξη (Phased Development)



*Υλοποιεί τις πιο **σημαντικές** και συνάμα **ξεκάθαρες** λειτουργίες*

Δημιουργία Πρωτοτύπων

Στόχος είναι η συνεργασία με τον πελάτη και η άμεση δημιουργία μιας πρώτης έκδοσης του συστήματος ξεκινώντας από μία αρχική περιγραφή.

Θα πρέπει να ξεκινά με **πολύ καλά κατανοητές απαιτήσεις**.

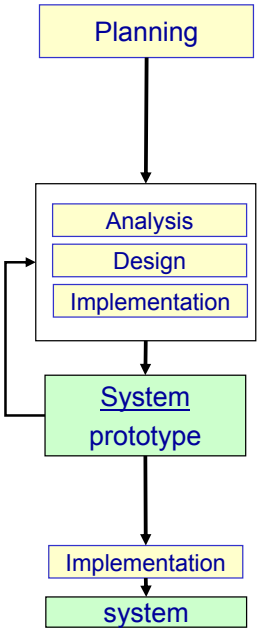
Το σύστημα αναπτύσσεται προσθέτοντας σταδιακά νέα χαρακτηριστικά. Το αρχικό πρωτότυπο θα εξελιχθεί στο τελικό σύστημα που θα εγκατασταθεί

Ανάπτυξη Throwaway πρωτοτύπων

Στόχος είναι η κατανόηση των απαιτήσεων του συστήματος. Ξεκινά με τις **λιγότερο κατανοητές απαιτήσεις**.

Το προϊόν πιθανότατα θα δημιουργηθεί με διαφορετικό τρόπο (throwaway prototype: διαφημιστικά/πρόχειρο πρωτότυπο)

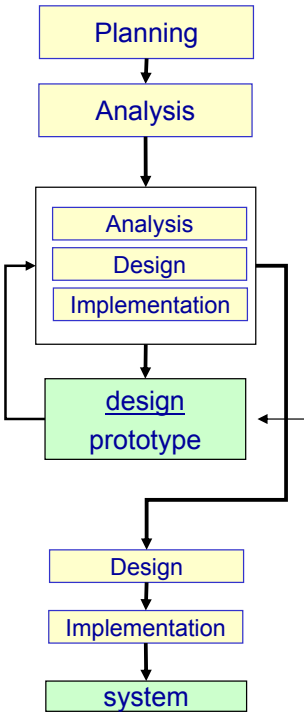
Rapid Application Development (RAD) Methodologies
Πρωτοτυποποίηση (prototyping)



Χαρακτηριστικά
- Κυκλική διαδικασία
- Το πρωτότυπο είναι κεντρικό σημείο

Concurrent phases
quick & dirty first prototype

Rapid Application Development (RAD) Methodologies
Throwaway prototyping



Χαρακτηριστικά
- Το πρωτότυπο απλά χρησιμοποιείται για να ξεκαθαρίσει τα πράγματα (δεν θα είναι τμήμα του πραγματικού συστήματος που θα κατασκευαστεί)

E.g. by Vbasic, Web UI,...

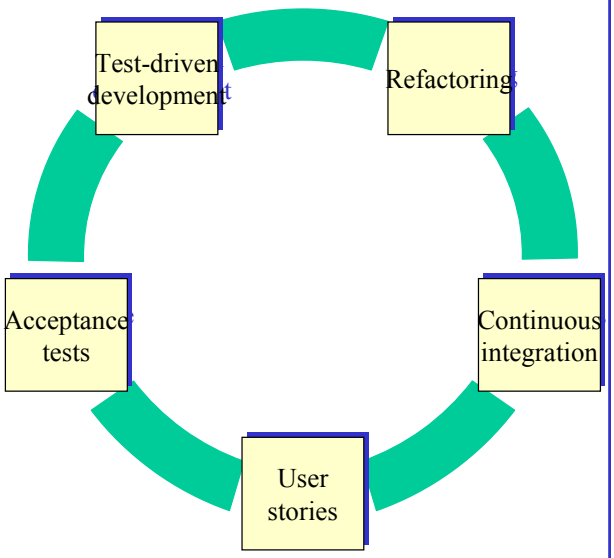
Εύκαμπτες Μεθοδολογίες Ανάπτυξης
Agile Development Methodologies

Εύκαμπτες Μεθοδολογίες Ανάπτυξης
(**Agile** Development Methodologies)

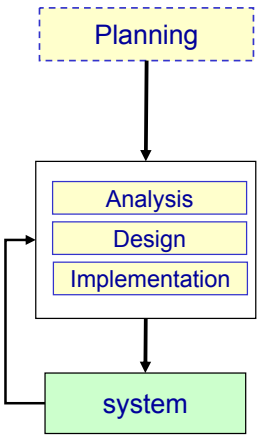
- Μεθοδολογίες που επικεντρώνουν στο προγραμματισμό (programming-centric)
- Λίγοι και εύκολα ακολουθήσιμοι κανόνες
- Στοχεύουν στην εξάλειψη του κόστους της μοντελοποίησης και τεκμηρίωσης (documentation)
- Δίνουν έμφαση στην απλή και επαναληπτική (iterative) ανάπτυξη και στην στενή σχέση με τον πελάτη
- Παραδείγματα τέτοιων μεθοδολογιών
  - eXtreme Programming (XP)
  - Scrum
  - Dynamic Systems Development Method (DSDM)
- Δείτε
  - Τον ιστότοπο της σχετικής «συμμαχίας» (www.agileAlliance.org)

Εύκαμπτη (Agile) ανάπτυξη λογισμικού

- Κύρια σημεία
  - Έμφαση στα πρόσωπα και στις επαναλήψεις αντί στις διαδικασίες και στα εργαλεία
  - Λογισμικό που δουλεύει αντί ογκώδους τεκμηρίωσης
  - Συνεργασία με τον πελάτη αντί διαπραγματεύσεων μέσω συμβολαίων
  - Ευθύνη για αλλαγές αντί της τυφλής ακολούθησης του πλάνου
- Πιο γνωστοί αντιπρόσωποι
  - eXtreme Programming (XP)
  - Aspect Oriented Software Development
  - Feature-driven Development
  - Lean Development

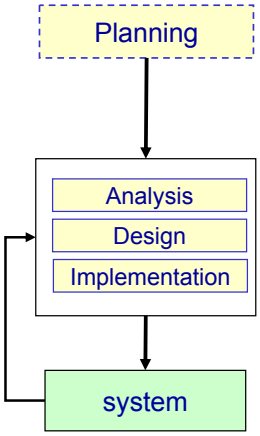


Εύκαμπτες Μεδοθολογίες Ανάπτυξης
eXtreme Programming (XP)



- Οι 4 βασικές «αξίες» του XP
  - Επικοινωνία (communication)
  - Απλότητα (simplicity)
    - **The KISS principle *(Keep It Simple, Stupid)***
  - Ανατροφοδότηση (feedback)
  - Κουράγιο (courage)
- Αρχές
  - Συνεχείς δοκιμές (continuous testing)
  - Απλή κωδικοποίηση από ζεύγη προγραμματιστών (simple coding performed by pairs of developers)
  - Στενή αλληλεπίδραση με τους τελικούς χρήστες
- Δοκιμές
  - Ο κώδικας δοκιμάζεται και τοποθετείται σε ένα ενοποιημένο περιβάλλον δοκιμών (καθημερινώς)
- Refactoring (Επαναπαραγοντοποίηση)

eXtreme Programming (XP)



Πλεονεκτήματα

- Αποφεύγει τον κίνδυνο της μη κατανόησης των πραγματικών απαιτήσεων (αφού υπάρχει στενή συνεργασία με τους πελάτες).
  - <<Δείτε τους συνήθεις κινδύνους-προβλήματα που είδαμε στο 2ο και 3ο μάθημα)
- Αποφεύγει το κόστος τεκμηρίωσης κλπ.
- Κατάλληλη για μικρές ομάδες προγραμματιστών.

Μειονεκτήματα

- Η ανάλυση και η σχεδίαση μένουν ατεκμηρίωτες. Το έργο αφήνει πίσω του μόνο κώδικα.
- Μάλλον ακατάλληλη για μεγάλα έργα

Principles behind the Agile Manifesto (http://www.agilemanifesto.org/)

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Feb 11-13, 2001

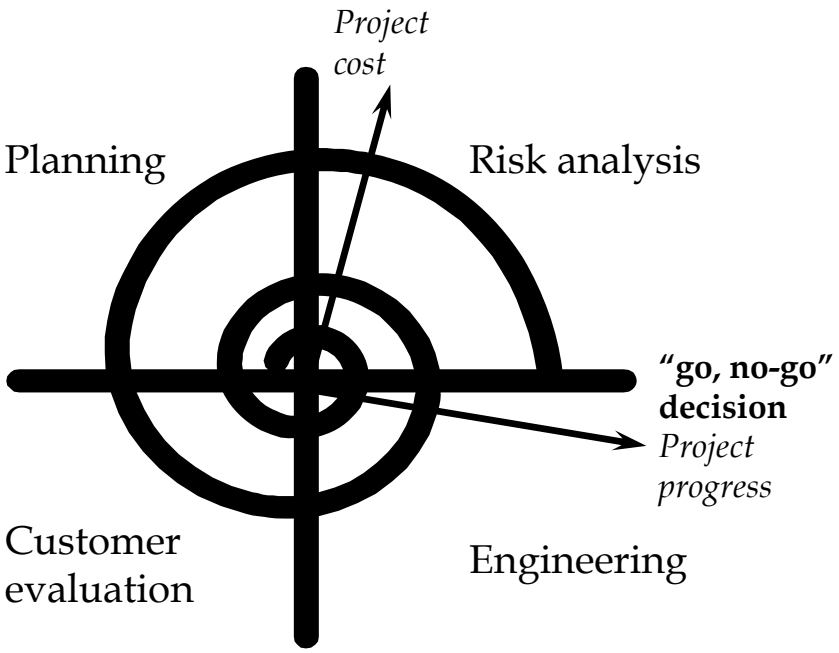
From www.agileAlliance.org:

- The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker.

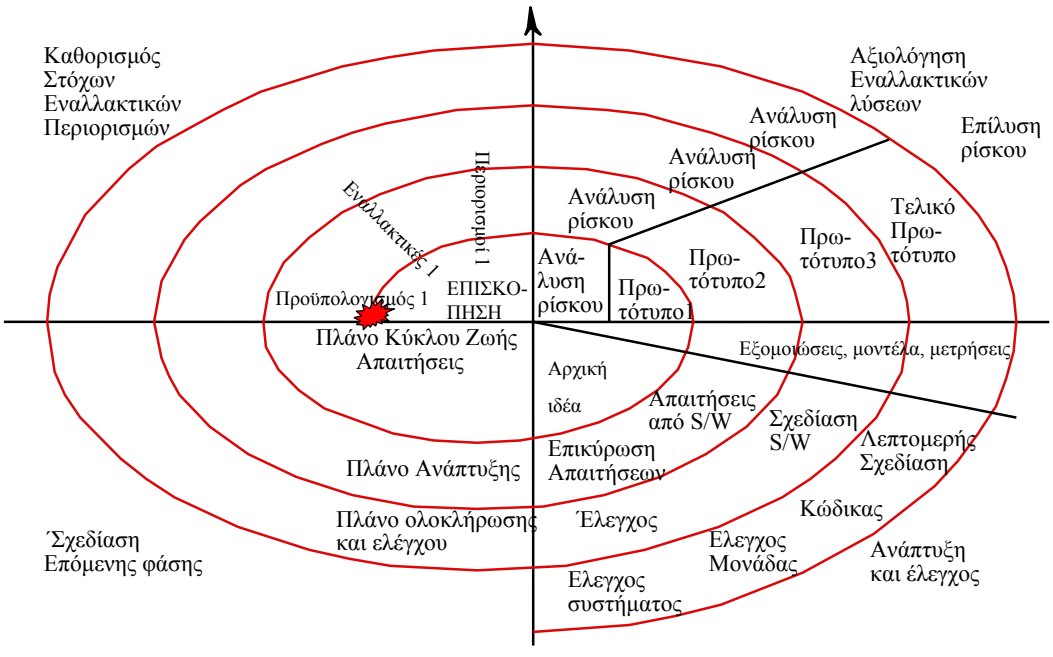
Μοντέλα Ανάπτυξης

Το μοντέλο της Σπείρας (the Spiral Model)

Μοντέλα Ανάπτυξης
Το μοντέλο της Σπείρας (the Spiral Model)



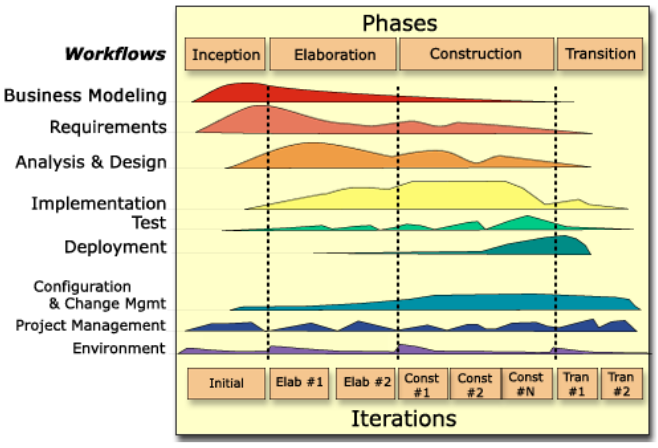
Μοντέλα Ανάπτυξης
Το μοντέλο της Σπείρας (the Spiral Model)



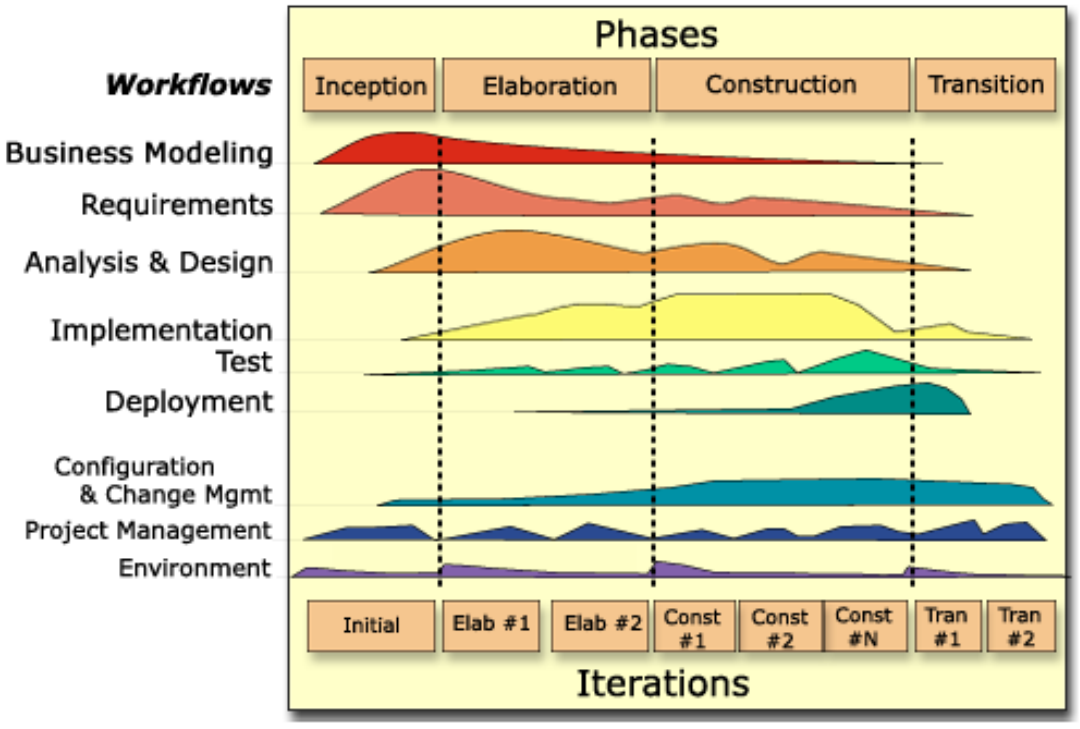
Μοντέλα Ανάπτυξης
Rational Unified Process (RUP)

Οργάνωση σε 2 διαστάσεις

- Οριζοντίως: Οι διαδοχικές φάσεις:
  - Σύλληψη (inception)
  - Επεξεργασία (elaboration)
  - Κατασκευή (construction)
  - Μετάβαση (transition)
- Καθέτως
  - Παριστάνει τις 7 δραστηριότητες



Μοντέλα Ανάπτυξης
Rational Unified Process (RUP)



Εξελικτικές Μεθοδολογίες

Πλεονεκτήματα

- Αντιμετωπίζει την ασάφεια στις απαιτήσεις
- Παρέχει τη δυνατότητα στον πελάτη να αλλάξει γνώμη πριν υπογράψει
- Μείωση χρόνου ανάπτυξης
- Τα αρχικά πρωτότυπα χρησιμοποιούνται για εξοικείωση από τους χρήστες
- Μεγαλύτερη πιθανότητα ανάπτυξης φιλικού προς το χρήστη λογισμικού
- Ο πελάτης εμπλέκεται στην ανάπτυξη του προϊόντος
- Αυξανόμενη σταδιακά ικανοποίηση του πελάτη
- Επικοινωνία χρηστών / ομάδος ανάπτυξης

Προβλήματα

- Έλλειψη παρατήρησης στη διαδικασία (αδυναμία πρόβλεψης πλήθους επαναλήψεων)
- Συστήματα «λιτά» δομημένα λόγω συχνών αλλαγών
- Εστίαση στη λειτουργικότητα – Λιγότερη έμφαση στις μη λειτουργικές απαιτήσεις
- Ειδικές ικανότητες μπορεί να απαιτηθούν (π.χ. γλώσσες για rapid prototyping)
- Υπερβολικός ενθουσιασμός από τον πελάτη. Ενδεχόμενη υποεκτίμηση του χρόνου ανάπτυξης
- Η δυνατότητα για συνεχείς τροποποιήσεις μειώνουν το βαθμό ικανοποίησης

Εξελικτικές Μεθοδολογίες

Κατάλληλη για

- Μικρά ή μεσαίου μεγέθους συστήματα
  - Π.χ. όταν το αρχικό πρωτότυπο πλησιάζει το τελικό προϊόν
- Τμήματα μεγάλων συστημάτων
  - π.χ. η επαφή χρήσης ενός συστήματος εναέριας κυκλοφορίας
- Συστήματα με μικρό χρόνο ζωής
- Συστήματα όπου υπάρχει αδυναμία έκφρασης των απαιτήσεων εξαρχής

Ακατάλληλη για

- Λογισμικό ενσωματωμένων συστημάτων
- Λογισμικό πραγματικού χρόνου
- Επιστημονικό λογισμικό

Πώς επιλέγουμε μια μεθοδολογία;

Επιλέγοντας την Κατάλληλη Μεθοδολογία

Δεν υπάρχει μια που να είναι η καλύτερη για κάθε περίπτωση.
Κριτήρια που θα μπορούσαν να φανούν χρήσιμα είναι:

- ***Διαύγεια (Σαφήνεια) Απαιτήσεων Χρηστών (Clarity of User Requirements)***
- ***Εξοικείωση με την Τεχνολογία (Familiarity with Technology)***
- ***Πολυπλοκότητα Συστήματος (System Complexity)***
- ***Αξιοπιστία Συστήματος (System Reliability)***
  - ***e.g. a missile control system***
- ***Βραχυπρόθεσμο Χρονοδιάγραμμα (Short Time Schedules)***
- ***Ορατότητα Βαθμού Προόδου (Schedule Visibility)***

Επιλέγοντας την Κατάλληλη Μεθοδολογία

Ικανότητα Ανάπτυξης ΠΣ με

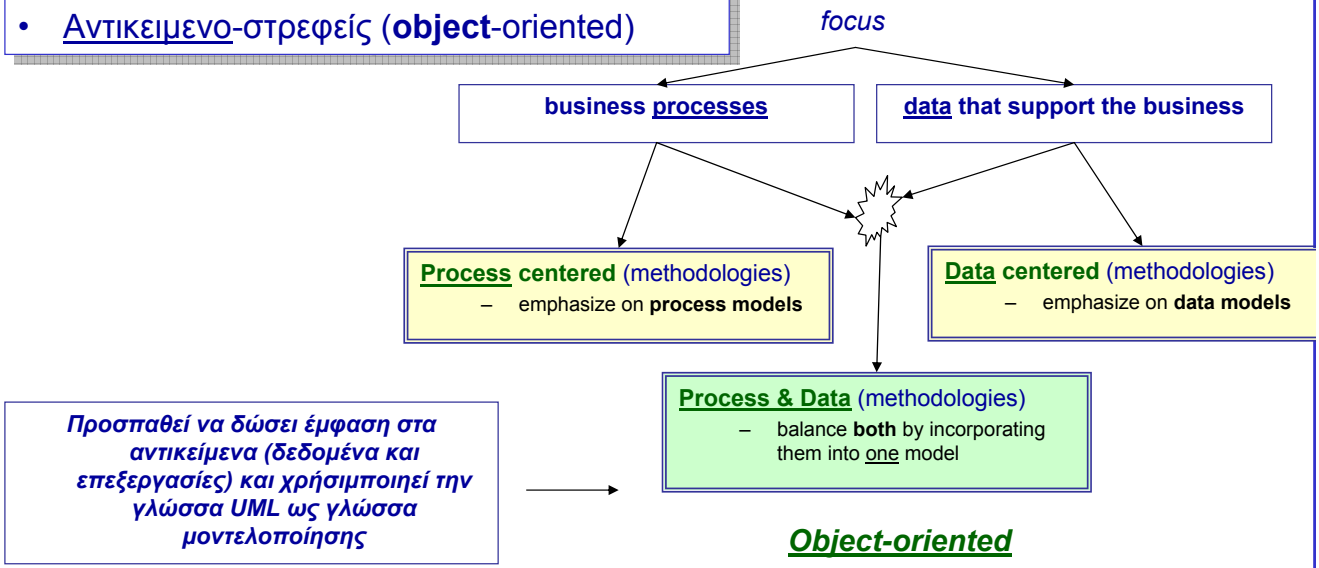
	Waterfall	*Parallel*	*Phased*	*Prototyp.*	*Thr. Prot.*	*XP*
with unclear requirements	**Poor**	Poor	**Good**	Poor	**V.Good**	**V.Good**
with Unfamiliar Technology	**Poor**	Poor	**Good**	Poor	**V.Good**	Poor
that are Complex	**Good**	Good	Good	Poor	**V.Good**	Poor
that are Reliable	**Good**	Good	Good	Poor	**V.Good**	Good
with a Short Time Schedule	**Poor**	Good	**V.Good**	**V.Good**	Good	**V.Good**
with Schedule Visibility	**Poor**	Poor	**V. Good**	**V. Good**	Good	Good

Χωρίς να είναι τίποτα απόλυτο

Αντικειμενοστρεφείς (object-oriented) Μεθοδολογίες Ανάλυσης και Σχεδίασης Πληροφοριακών Συστημάτων

Υπενθύμιση (από μάθημα 2)
(μεθοδολογίες ανάλυσης και σχεδίασης πληροφοριακών συστημάτων)

Ανάλογα με το που δίδεται έμφαση:
- Επεξεργασιο-κεντρικές (**process** centered)
- Δεδομενο-κεντρικές (**data** centered)
- Αντικειμενο-στρεφείς (**object**-oriented)



Αντικειμενοστρεφής μεθοδολογία
Ιστορία

- Όπως ο δομημένος προγραμματισμός οδήγησε στην δομημένη μεθοδολογία σχεδίασης, έτσι και ο αντικειμενοστρεφής προγραμματισμός οδήγησε στην αντικειμενοστρεφή μεθοδολογία σχεδίασης πληροφοριακών συστημάτων

Κύρια Σημεία

- Στοχεύει στην επιτάχυνση της διαδικασίας ανάπτυξης λογισμικού
- Είναι ευέλικτη (διευκολύνει την αλλαγή απαιτήσεων, την σταδιακή βελτίωση και επέκταση, τον έλεγχο, την συντήρηση)
- Έχει έμφυτη την αρχή της επανάληψης (επιστροφή σε προηγούμενες φάσεις)
- Εκμεταλλεύεται τις αρχές και τεχνικές του αντικειμενοστραφούς προγραμματισμού
  - Αφαίρεση (abstraction), ενθυλάκωση (encapsulation), κληρονομικότητα (inheritance), πολυμορφισμός (polymorphism), αναχρησιμοποίηση (reuse)

Ένα σημαντικό πλεονέκτημα για την Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων

Η μετάβαση από φάση σε φάση είναι ευκολότερη διότι σε κάθε φάση χρησιμοποιείται η ίδια «γλώσσα»:

- *Τα αντικείμενα του πραγματικού κόσμου μοντελοποιούνται ως αντικείμενα (στη φάση ανάλυσης), κατόπιν μεταφράζονται άμεσα σε αντικείμενα σχεδίασης (στη φάση της σχεδίασης), και τέλος υλοποιούνται ως αντικείμενα μιας αντικειμενοστρεφούς γλώσσας προγραμματισμού ή μιας αντικειμενοστραφούς βάσης δεδομένων (OODB).*

Αντικειμενοστρεφής Ανάλυση και Σχεδίαση σε σχέση με (Γενικές) Μεθοδολογίες Ανάπτυξης Λογισμικού

- Οι αντικειμενοστραφείς προσεγγίσεις για ανάπτυξη ΠΣ θα μπορούσαν να χρησιμοποιήσουν οποιαδήποτε από τις γενικές μεθοδολογίες που είδαμε
  - Καταρράκτη, παράλληλη, πολυφασική, προτυποποίηση, RAD, XP, κλπ.
- Κυρίως σχετίζονται όμως με πολυφασικές και εξελικτικές (RAD) μεθοδολογίες

Object-Oriented Analysis and Design

Σύμφωνα με τον δημιουργό της UML, οποιαδήποτε μοντέρνα αντικειμενοστραφής μεθοδολογία ανάπτυξης ΠΣ πρέπει να

- **Καθοδηγείται από τις Περιπτώσεις Χρήσεις (Use-case Driven)**
- **Αρχιτεκτονικοκεντρική (Architecture Centric)**
  - Η αρχιτεκτονική λογισμικού πρέπει να οδηγεί την προδιαγραφή (specification) , κατασκευή (construction) και τεκμηρίωση (documentation) του συστήματος
  - Πρέπει να υποστηρίζονται τουλάχιστον 3 διαφορετικές όψεις της αρχιτεκτονικής:
    - **Λειτουργική (Functional)**
    - **Στατική (Static)**
    - **Δυναμική (Dynamic)**
- **Επαναληπτική και Αυξητική (Iterative and Incremental)**
  - Το λογισμικό δεν πρέπει να εκδίδεται μονομιάς (σαν ένα Big Bang) στο τέλος του έργου. Πρέπει να υπάρχουν αρκετές επαναλήψεις και κάθε επανάληψη να φέρνει το σύστημα όλο και εγγύτερα στις πραγματικές ανάγκες του χρήστη

(Επανάληψη από HY252)
Οι βασικές έννοιες του αντικειμοστρεφισμού
(object-orientation)

Βασικές Έννοιες

Οι πλέον βασικές έννοιες της αντικειμενοστρεφούς προσέγγισης είναι

- Το **αντικείμενο** (object)
- Οι **μέθοδοι** (methods)
- Τα **μηνύματα** (messages)

Οι τρεις αυτές έννοιες βοηθούν στην κατανόηση των

- **Ενθυλάκωση** (encaspulation)
- **Τμηματοποίηση** (modularity)
- **Αφαίρεση** (abstraction)
- **Πολυμορφισμός** (polymorphism)

Βασικές Έννοιες (ΙΙ)

Τα **αντικείμενα** αποτελούν τους δομικούς λίθους του συστήματος. Σχετικές έννοιες:

- **Κλάσεις** (classes): αρχέτυπα αντικειμένων
- **Στιγμιότυπα** (instances): μέλη κλάσεων

Η **οργάνωση των κλάσεων** χρησιμοποιείται για να γίνει το μοντέλο του φυσικού κόσμου. Σε αυτό βοηθούν οι έννοιες:

- **Ιεραρχίες** (hierarchies)
- **Κληρονομικότητα** (inheritance)

αντικείμενο

Αντικείμενο

- Λογική ενότητα η οποία συνδυάζει δεδομένα και επεξεργασίες για να εκπληρώσει το ρόλο της μέσα στο σύστημα
- Σύμφωνα με τον Booch (1991) για να οριστεί ένα αντικείμενο θα πρέπει να έχει:
  - **Κατάσταση** (state), η οποία καθορίζεται από τις ιδιοκτησίες του και τις τιμές σε κάθε μια από αυτές
  - **Συμπεριφορά** (behavior), η οποία αναφέρεται στο πως δρα το αντικείμενο μέσα στο περιβάλλον του, δηλαδή πως αντιδρά σε κάθε ενεργοποίηση των μεθόδων του
  - **Ταυτότητα**, δηλαδή το μοναδικό χαρακτηριστικό το οποίο αντιστοιχεί σε μια οντότητα του πραγματικού κόσμου

μέθοδοι, μηνύματα, ενθυλάκωση

Μέθοδοι

- Οι πράξεις που εκτελεί το αντικείμενο όταν λαμβάνει ένα μήνυμα. Οι πράξεις μπορούν να εκτελούνται μόνο αν είναι γνωστές στο αντικείμενο

Μηνύματα

- Είναι οι αιτήσεις που στέλνονται από ένα αντικείμενο σε ένα άλλο έτσι ώστε το τελευταίο να εκτελέσει μια λειτουργία

Ενθυλάκωση

- Η συγκέντρωση, ομαδοποίηση δεδομένων και πράξεων στην ίδια δομή. Με αυτόν τον τρόπο «κρύβονται» οι εσωτερικές λεπτομέρειες της υλοποίησης και αυτό αποτελεί σημαντικό πλεονέκτημα και για την επαναχρησιμοποίηση
  - Οι εσωτερικές λεπτομέρειες περιλαμβάνουν δεδομένα για την κατάσταση των ενοτήτων και τις διαδικασίες που καθορίζουν τη συμπεριφορά τους

τμηματοποίηση, αφαίρεση, πολυμορφισμός

Τμηματοποίηση (modularity)

- Ο χωρισμός πολύπλοκων συστημάτων σε ενότητες (modules)

Αφαίρεση (abstraction)

- Η απλοποιημένη περιγραφή και καθορισμός του συστήματος ώστε να δίδεται έμφαση σε ορισμένες λεπτομέρειες και να αγνοούνται άλλες

Πολυμορφισμός

- Η ιδιότητα που επιτρέπει να σταλεί το ίδιο μήνυμα σε διαφορετικά αντικείμενα και το κάθε αντικείμενο να εκτελέσει μια λειτουργία που αντιστοιχεί στην κλάση στην οποία ανήκει

κλάση, στιγμιότυπο, κληρονομικότητα

Κλάση

- Αρχέτυπο ή καλούπι (template) αντικειμένων. Περιλαμβάνει τον ορισμό, τη δήλωση των δεδομένων και τις πράξεις που γίνονται σε αυτά.

Στιγμιότυπο (instance)

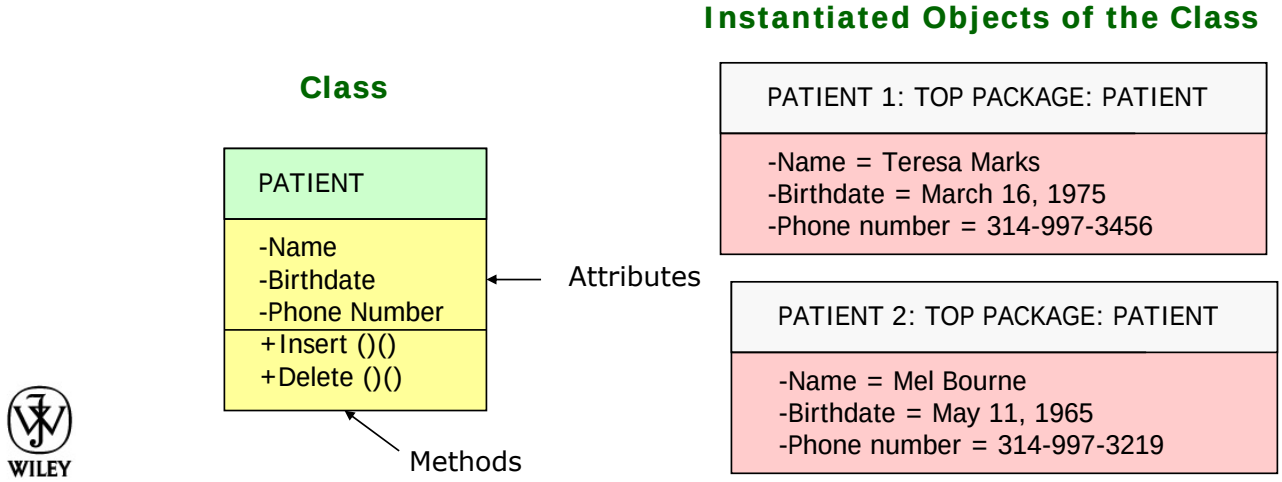
- Η συγκεκριμένη εμφάνιση που προσδιορίζει με ακρίβεια ένα μέλος της κλάσης.

Κληρονομικότητα (inheritance)

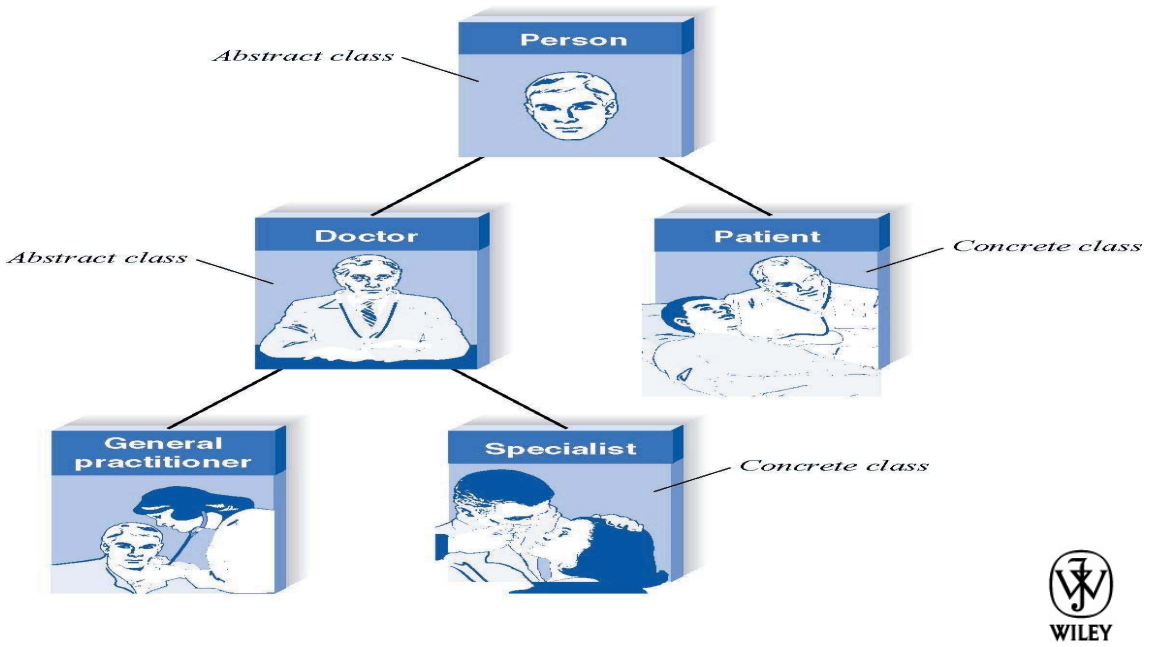
- Η δυνατότητα να ορίζει κανείς νέα αντικείμενα που είναι απόγονοι κάποιου άλλου (υπαρκτού) αντικειμένου. Ο απόγονος ενός αντικειμένου έχει τη δυνατότητα χρήσης όλων των δεδομένων και μεθόδων που κληρονομεί από τον πρόγονο του ενώ παράλληλα διατηρεί τα δεδομένα και τις μεθόδους που ο ίδιος ορίζει

Basic Characteristics of OO Systems

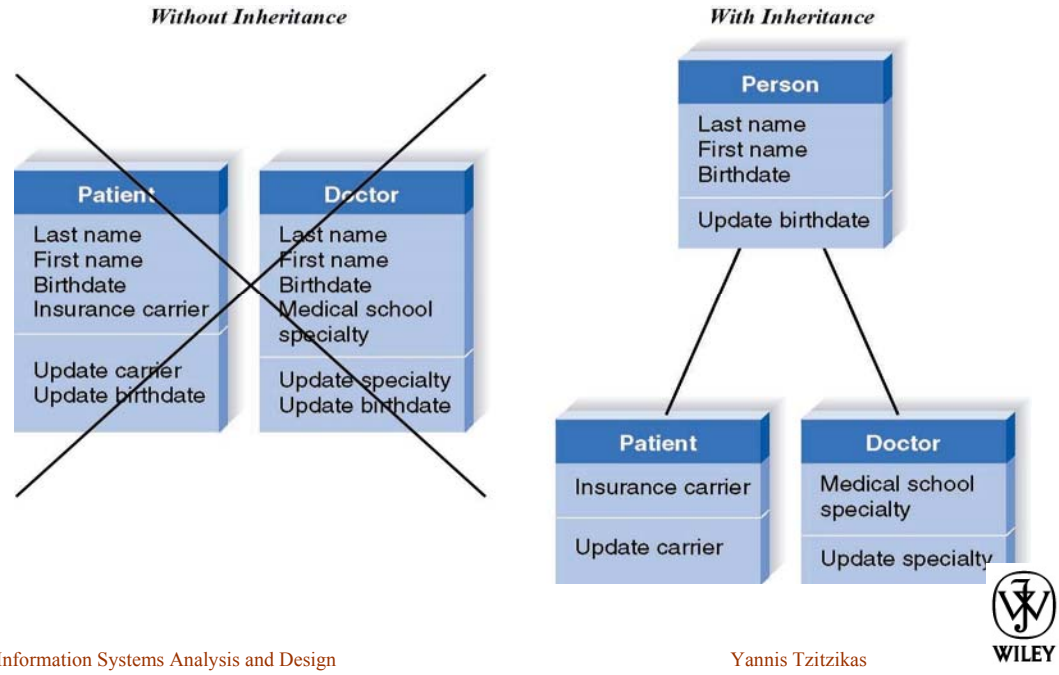
- *Classes* -- template to define objects
- *Instances* -- specific examples of class members
- *Objects* -- building block of the system
- *Attributes* -- describe data aspects of the object
- *Methods* -- the processes the object can perform
- *Messages* -- instructions sent to or received from other objects



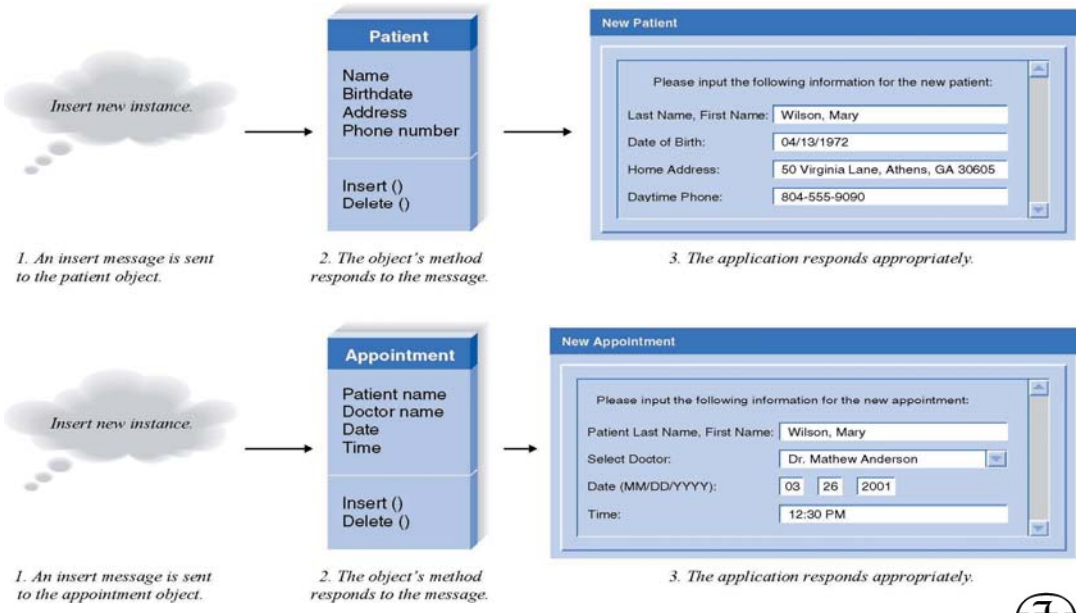
Class Hierarchy



Inheritance



Polymorphism and dynamic binding



Characteristics of some widely user oo languages

Feature	C++	Eiffel	Java
strong typing	Optional	yes	yes
static(s)/dynamic(d) typing	S	S	S+D
garbage collection	no	yes	yes
multiple inheritance	yes	yes	no
pure objects	no	yes	no
dynamic loading	no	no	yes
standardized class libraries	no	yes	yes
correctness constructs	no	yes	no

Μοντελο-οδηγούμενη Αρχιτεκτονική

MDA: Model Driven Architecture

(http://www.omg.org/mda/)

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA) Σκεπτικό

- Ανάλυση, Σχεδίαση και Υλοποίηση με τον Παραδοσιακό Τρόπο:
  - Απαιτεί τη συλλογή, μοντελοποίηση και υλοποίηση γνώσης που προέρχεται από το Πεδίου Εφαρμογής **ΚΑΙ** από την πλατφόρμα/τεχνολογία που θα χρησιμοποιηθεί για την υλοποίηση.
    - Τα δύο παραπάνω περιπλέκονται
- Ανάλυση, Σχεδίαση και Υλοποίηση βάσει MDA:
  - Προσπαθεί να ξεκαθαρίσει τα πράγματα
    - Ο ειδικός του πεδίου της εφαρμογής => δίδει (συλλέγει/αναλύει/μοντελοποιεί) εμπειρογνωμοσύνη του πεδίου
    - Ο ειδικός της πλατφόρμας => δίδει εμπειρογνωμοσύνη πλατφόρμας
    - Προγραμματιστής Εφαρμογής => δίδει τεχνολογική εμπειρογνωμοσύνη
  - Προτείνει διαφορετικών τύπων **μοντέλα** και **μετασχηματισμούς** μεταξύ αυτών.

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
Στόχος

- Να διαχωρίσει αυτά που αφορούν τη λειτουργικότητα και συμπεριφορά της εφαρμογής, από την τεχνολογία στην οποία θα αναπτυχθεί και θα παραδοθεί.
- Η υλοποίηση πρέπει να μπορεί να αλλάξει χωρίς να πρέπει να αλλάξει η ουσιαστική συμπεριφορά του συστήματος το οποίο βλέπει ο τελικός χρήστης

(Εν δυνάμει) Πλεονεκτήματα

- Μειωμένο κόστος και χρόνος ανάπτυξης
- Γρήγορη και ομαλή ενοποίηση των παλαιών με τις νέες τεχνολογίες

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
Βασικές Έννοιες

Platform Independent Model (PIM)

– Μοντέλο της εφαρμογής εκφρασμένο με τρόπο ανεξάρτητο της πλατφόρμας ανάπτυξης

Transformation
(μετασχηματισμός)

Platform Specific Model (PSM)

– Μοντέλο της εφαρμογής εκφρασμένο βάσει μιας συγκεκριμένης πλατφόρμας υλοποίησης

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
PIM (*Platform Independent Model*)

- The *Platform Independent Model* (PIM) specifies the business functionality of the application completely, but does not include anything that is specific for a certain technology or language.
- For instance, a PIM may specify that the application is split into a number of components, but it does not specify whether the application uses .NET or J2EE.
- The PIM is often written in UML and is completely independent from any details that may be different across the two platforms.

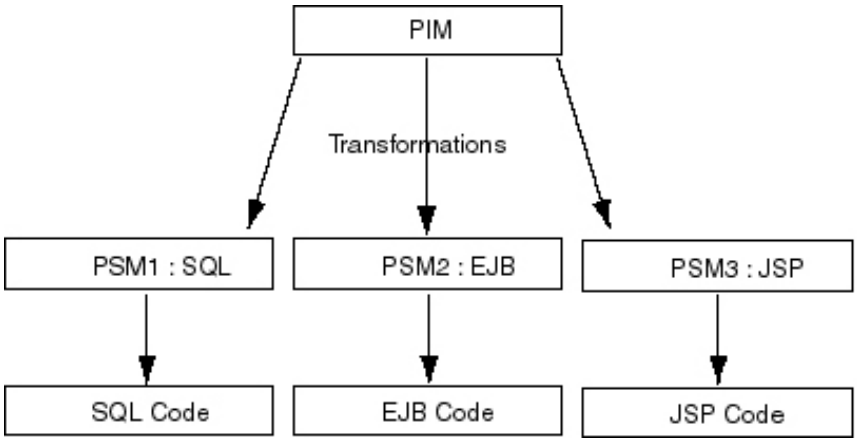
Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
PIMs and PSMs

- The best practice for (medium/large-scale systems) is to define a PIM first, then *refine it to a* PSM.
- This approach aids
  - ***Separation of concerns***: Important to understand the abstract architecture of a system to iron out problems in *that* first, then tackle platform-specific issues
  - ***Maintenance***: while platforms change over time, the abstract architecture of system will change less frequently – it is important to maintain a platform independent reference specification to ensure platform specific changes still implement the original abstract requirements
  - ***Interoperability:*** Systems are often implemented using several platforms – it is not possible to provide an overall design using a single PSM

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
Από ένα PIM σε ένα PSM: Μετασχηματισμοί

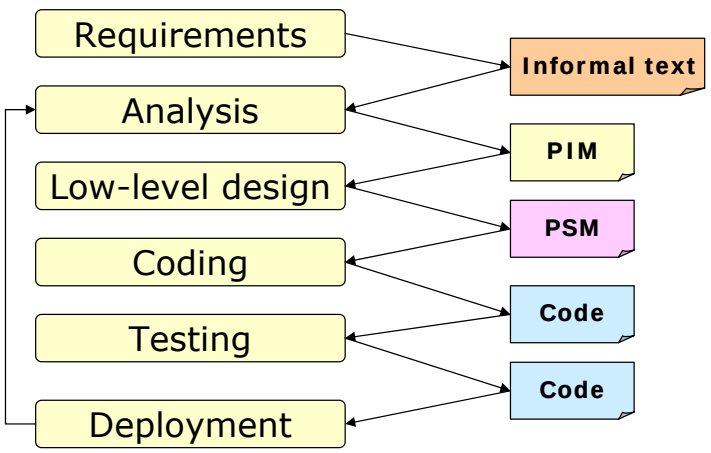
MDA tools will take a PIM and use a transformation definition to transform the PIM into one or more *Platform Specific Models* (PSMs). A PSM contains all the detail that is needed for the chosen platform.

It is crucial that the PSM is generated automatically—that is, there is no (or little) user effort needed to produce a PSM from a PIM. Usually the PSM is an almost direct reflection of the program code.



Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
Ανάλυση και Σχεδίαση ΠΣ βάσει της MDA

- According to MDA, the software development process is driven by **modelling**, rather than **coding**
- MDA involves the development of an initial PIM and successive refinement through PSMs to eventual code implementation
- At its simplest, the MDA software development lifecycle is as follows:



PIM used to define abstract specification of system – refined or transformed into a PSM that includes platform specific details – the PSM is closer to implementation code, so is more easily implemented

Μοντελο-οδηγούμενη Αρχιτεκτονική (MDA)
Ανάλυση και Σχεδίαση ΠΣ βάσει της MDA

Απώτερος στόχος είναι η πλήρως αυτοματοποιημένη παραγωγή του PSM από το PIM βάσει των κατάλληλων μετασχηματισμών.

- Μετασχηματισμός: αυτόματη παραγωγή ενός μοντέλου από ένα άλλο, βάσει ενός Ορισμού Μετασχηματισμού
- Ορισμός Μετασχηματισμού: ένα σύνολο Κανόνων Μετασχηματισμού
- Κανόνας Μετασχηματισμού: περιγραφή του πως ένα ή περισσότερα δομικά στοιχεία του μοντέλου πηγής (source model) μετατρέπονται σε ένα (ή περισσότερα) δομικά στοιχεία του μοντέλου στόχου (target model)

Σημερινή Κατάσταση

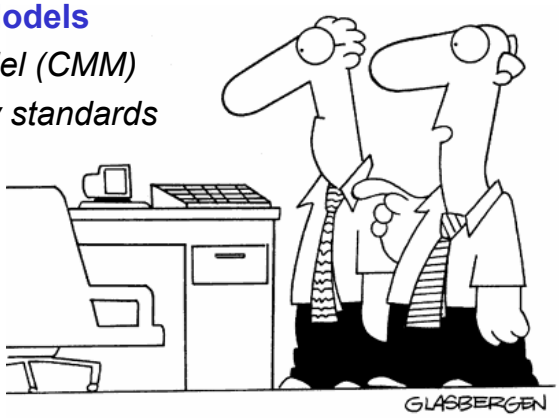
- Το QVT (Query/View/Transformation) είναι ένα νέο standard για τον ορισμό μετασχηματισμών (χρησιμοποιεί και την OCL που θα μάθουμε σε αυτό το μάθημα)
- Απώτερος στόχος είναι η ενσωμάτωση εργαλείων μετασχηματισμού στα εργαλεία μοντελοποίησης (που θα χρησιμοποιήσετε στο μάθημα αυτό)

(Περισσότερα ίσως προλάβουμε να πούμε προς το τέλος του εξαμήνου)

Μοντέλα Αποτίμησης και Βελτίωσης της Διαδικασίας (παραγωγής λογισμικού)

Process Improvement Models

- *Capability Maturity Model (CMM)*
- *ISO 9000 family of quality standards*



"We installed little monitors because they make all of our problems look smaller."

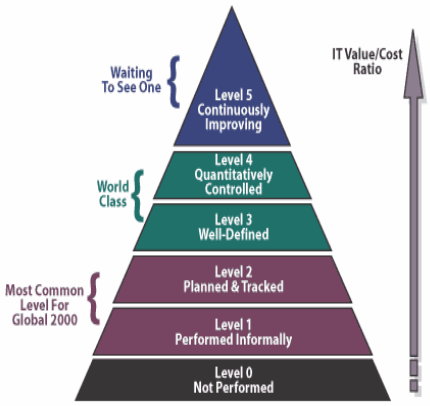
Process Improvement Models
Μοντέλα Βελτίωσης Διαδικασίας

- Κάθε οργανισμός που σχετίζεται με παραγωγή κώδικα επιθυμεί να βελτιώσει τη διαδικασία ανάπτυξής του.
- Για να τη βελτιώσει πρέπει πρώτα να μάθει ποια είναι τα προβλήματα της.
- Έναν τρόπο αποτίμησης (μέτρησης) της διαδικασίας ορίζουν τα εξής Μοντέλα Βελτίωσης Διαδικασίας (Process Improvement Models):
  - *Capability Maturity Model (CMM)*
  - *ISO 9000 family of quality standards*

Μοντέλα Βελτίωσης Διαδικασίας (Process Improvement Models)
Capability Maturity Model (CMM)

- "The stairway to software excellence"
- Ένας δημοφιλής τρόπος για αποτίμηση και βελτίωση της διαδικασίας (process assessment and improvement)
- Είναι ουσιαστικά ένα ερωτηματολόγιο που ο Οργανισμός Πληροφορικής πρέπει να συμπληρώσει
  - Κατόπιν ακολουθεί μια διαδικασία επαλήθευσης και επιβεβαίωσης και στο τέλος ο οργανισμός κατατάσσεται σε ένα από τα 5 επίπεδα του CMM (όσο μεγαλύτερο τόσο το καλύτερο)

CMM: Μοντέλο Ωριμότητας Ικανοτήτων (Δυνατοτήτων/Δυναμικού)



SEI (Software Engineering Institute), U. of Carnegie Mellon

Μοντέλα Βελτίωσης Διαδικασίας (Process Improvement Models)
Τα 5 επίπεδα του CMM

Level 5 Optimizing
Improve process itself
Continuous process improvement in place.

Level 4 Managed
Improve process control
Metrics used to control the process.

Level 3 Defined
Improve process definition
Both management and engineering processes are codified and followed.

Level 2 Repeatable
Improve process discipline
Repeatable project management; consistent time and effort predictions for similar projects.

Level 1 Initial
Unpredictable and undisciplined process that depends on the current staff.

5: **Continuous quality improving**

4: Measurement

3:Method and tools

2:Project mgmt

1: **Chaos**

Μοντέλα Βελτίωσης Διαδικασίας (Process Improvement Models)
Process maturity levels in CMM

Level 5 Optimizing
Improve process itself
Continuous process improvement in place.

Level 4 Managed
Improve process control
Metrics used to control the process.

Level 3 Defined
Improve process definition
Both management and engineering processes are codified and followed.

Level 2 Repeatable
Improve process discipline
Repeatable project management; consistent time and effort predictions for similar projects.

Level 1 Initial
Unpredictable and undisciplined process that depends on the current staff.

- Most organizations are at level 1; some at level 2; very few are known to be at level 5
- experience shows that it takes years to progress one level up the maturity scale

CMM Level 2

- For CMM level 2 an organization must provide positive answers to all these questions (and more)
  - Does the software quality assurance function have a **management reporting channel** separate from the software development project mgmt?
  - Is there a **software configuration control function** for each project that involves software development?
  - Is a formal process used in the management review of each software development prior to main contractual commitments ?
  - Is a **formal procedure** used to produce **software development schedules** ?
  - Are **formal procedures** applied to **estimate software development cost**?
  - Are **statistics** on **software code** and **test errors** gathered ?
  - Does senior management have a mechanism for the **regular review** of the status of software development projects ?
  - Is a mechanism for **controlling changes to the software requirements** ?

Επίπεδο CMM και μερικά στατιστικά

Η διάρκεια, η προσπάθεια, οι αστοχίες και το κόστος ενός έργου 200K γραμμών κώδικα, σε σχέση με το επίπεδο CMM του οργανισμού που ανέλαβε την εκπόνηση του έργου:

	Organization's **CMM Level**	Project **duration** (months)	Project **person months**	Number of **Defects**	Median **Cost**
Οργανισμός X	1	30	600	61	5.5 M $
Οργανισμός Y	2	18	153	12	1 M $
Οργανισμός Z	3	15	80	7	0.5 M$

Οι οικογένεια των προτύπων ποιότητας ISO 9000

- ISO: International Organization for Standardization (Διεθνής Οργανισμός Τυποποίησης)
- Τα πρότυπα ISO
  - Αφορούν στην **διαχείριση ποιότητας** (**quality management**) και στη διαδικασία παραγωγής προϊόντων με ποιότητα
  - Είναι εφαρμόσιμα σε **οποιαδήποτε βιομηχανία** ή επιχείρηση, συμπεριλαμβανομένης και της ανάπτυξης λογισμικού
- Σκεπτικό
  - Αν μια **διαδικασία** (**process)** είναι σωστή τότε τα παράγωγα της (προϊόντα ή υπηρεσίες) θα είναι επίσης «καλά»
- Τα πρότυπα του ISO δεν επιβάλλουν κάποια συγκεκριμένη διαδικασία. Παρέχουν μόνο μοντέλα του τι πρέπει να επιτευχθεί, χωρίς όμως να προτείνουν πώς πρέπει να γίνονται οι διάφορες δραστηριότητες.

Πηγές – Αναφορές - Μελέτη

- **Systems Analysis and Design with UML Version 2.0** (2nd edition) by A. Dennis, B. Haley Wixom, D. Tegarden, Wiley, 2005. CHAPTER 1,2
- **Requirements Analysis and System Design** (2nd edition) by Leszek A. Maciaszek, Addison Wesley, 2005, CHAPTER 1
- Shari Lawrence Pfleeger. Τεχνολογία Λογισμικού: Θεωρία και Πράξη, 1. Κλειδάριθμος, Αθήνα, 2003, Κεφάλαιο 1,2.
- **Object-Oriented Systems Analysis and Design Using UML** (2nd edition) by S. Bennett, S. McRobb, R. Farmer, McGraw Hil, 2002, Chapter 2, 4
- Ε. Κιουντούζης, Μεθοδολογίες Ανάλυσης και Σχεδιασμού Πληροφοριακών Συστημάτων,Εκδόσεις Α.Σταμούλη, Αθήνα 2002, Κεφ. 11
- Shari Lawrence Pfleeger. Τεχνολογία Λογισμικού: Θεωρία και Πράξη, 1. Κλειδάριθμος, Αθήνα, 2003, Κεφάλαιο 6