ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΠΜΣ Διοίκηση Εκπαίδευσης

Τεχνολογία Πληροφοριακών Συστημάτων – Μεθοδολογίες, Μοντέλα, Μοντελοποίηση Συστημάτων (Μέρος 1ο)

Δρ Χρήστος Πιερρακέας

Πληροφοριακό Σύστημα

Aktas (1987), Ahitur και Neumann (1990):

«Ένα σύστημα το οποίο δέχεται πληροφορίες, τις αποθηκεύει, τις ανακτά, τις μετασχηματίζει, τις επεξεργάζεται και τις διανέμει στους διάφορους χρήστες του οργανισμού, χρησιμοποιώντας υπολογιστές ή άλλα μέσα».

Συνιστώσες Πληροφοριακού Συστήματος

1. Άνθρωποι
2. Διαδικασίες
3. Δεδομένα
4. Λογισμικό
5. Υλικός εξοπλισμός

Ορισμός Κύκλου Ζωής ΠΣ

Η πορεία ενός Π.Σ., από τη στιγμή του καθορισμού του προβλήματος ή των προβλημάτων που πρέπει να επιλύσει,

μέχρι τη λειτουργία του, τη συντήρησή του και, τέλος, την απόσυρσή του,

είναι γνωστός στη βιβλιογραφία ως
Κύκλος Ζωής του Πληροφοριακού Συστήματος.

Τεχνολογία Πληροφοριακών Συστημάτων

Η **τεχνολογία πληροφοριακών συστημάτων** είναι ένας τεχνικός κλάδος που ασχολείται με όλες τις πτυχές της διαδικασίας παραγωγής του, από τα πρώτα στάδια της εξαγωγής προδιαγραφών του συστήματος μέχρι τη συντήρηση του μετά τη διάθεσή του για χρήση και μέχρι την απόσυρσή του.

Τεχνολογία Πληροφοριακών Συστημάτων είναι:

- η περιοχή εκείνη της επιστήμης της μηχανικής η οποία ασχολείται με την εύρεση και θεμελίωση μεθόδων για να περιγράφεται, να κατασκευάζεται και να συντηρείται ένα πληροφοριακό σύστημα.

Τεχνολογία Πληροφοριακών Συστημάτων

Η ανάγκη για «καλά» πληροφοριακά συστήματα είναι αυτονόητη και επιτακτική διότι…

Ο ρόλος τους …

•Στην οικονομία και την παραγωγή

•Στην ενημέρωση και την ψυχαγωγία

•Στην εκπαίδευση και αλλού

…εκπληρώνεται μόνο από Πληροφοριακά Συστήματα που:

•Κάνουν σωστά τη δουλειά τους

•Παράγονται με λογικό κόστος (και on budget),

•…σε λογικό χρόνο (και on time),

•…και είναι καλής ποιότητας (quality).

Στάδια Κύκλου Ζωής ΠΣ

1. Διερευνητική μελέτη
2. Μελέτη σκοπιμότητας
3. Ανάλυση απαιτήσεων
4. Σχεδιασμός συστήματος
5. Υλοποίηση
6. Εγκατάσταση
7. Λειτουργία-Συντήρηση

Αυτά τα στάδια μαζί αναφέρονται συχνά ως:

«συμβατική ανάλυση συστημάτων», ή

«παραδοσιακή ανάλυση συστημάτων», ή

«κύκλος ανάπτυξης και ζωής πληροφοριακών συστημάτων», ή

μοντέλο καταρράκτη

Διερευνητική μελέτη (1/2)

- Ποιο είναι το υπό εξέταση σύστημα;
- Ποιο είναι το πραγματικό πρόβλημα;
- Ποιες είναι οι υπάρχουσες εναλλακτικές λύσεις;

Ζητούμενο: Ο χρήστης θα επιλέξει μία λύση για περαιτέρω εξέταση.

Διερευνητική μελέτη (2/2)

Ο βασικός και αντικειμενικός στόχος της φάσης αυτής είναι **ο προσδιορισμός του προβλήματος**.

Η διερευνητική μελέτη είναι **ιδιαίτερα απαραίτητη** για:

- μεσαία και μεγάλα έργα με υψηλό προϋπολογισμό και μεγάλα χρονικά περιθώρια ανάπτυξης
- έργα που δεν είναι εγγυημένη η αναγκαιότητά τους
- έργα που δεν είναι καλώς και με σαφήνεια ορισμένα.
- έργα ευαίσθητα από τεχνολογικής πλευράς

Αντίθετα, η διερευνητική μελέτη **μπορεί να παραληφθεί** για:

- μικρά έργα με περιορισμένο προϋπολογισμό και μικρά χρονικά περιθώρια ανάπτυξης
- έργα που είναι με σαφήνεια διατυπωμένα
- έργα που αποτελούν ένα κομμάτι ενός μεγαλύτερου στρατηγικού σχεδίου

Μελέτη Σκοπιμότητας (1/3)

- Είναι εφικτή η υλοποίηση της λύσης;
- Υπάρχουν εναλλακτικοί τρόποι υλοποίησης;
- Με ποιο κόστος/όφελος;

Ζητούμενο: Περιγραφή της λύσης που επιλέχθηκε για υλοποίηση.

Μελέτη Σκοπιμότητας (2/3)

Χρόνος (Πόσο χρόνο χρειάζεται η υλοποίηση κάθε λύσης;)

Κόστος (Ποια είναι η εκτίμηση για το συνολικό κόστος κάθε λύσης;)

Όφελος (Ποια είναι τα οφέλη που θα προσφέρει η κάθε λύση;)

Τεχνολογία (Είναι η λύση εφικτή τεχνολογικά; Υπάρχουν δυσκολίες στην υλοποίησή της;)

Τεχνογνωσία (Υπάρχει στον οργανισμό η κατάλληλη τεχνική υποδομή; Έχουν οι υπάλληλοι που θα το χειρίζονται τις γνώσεις ώστε να το χρησιμοποιήσουν αποδοτικά;)

Μετάπτωση στο νέο σύστημα (Ποιός θα είναι ο τρόπος εισαγωγής του νέου συστήματος και αφαίρεσης του παλαιού;)

Εξέταση των παραμέτρων ΑΕΜΕΒΥ (Αύξηση Εσόδων, Μείωση Εξόδων, Βελτίωση Υπηρεσιών).

Μελέτη Σκοπιμότητας (3/3)

Οποιοδήποτε προτεινόμενο σύστημα πρέπει να εξεταστεί από άποψη:

- **Νομική** (π.χ. δεν παραβιάζει την ιδιωτικότητα)
- **Οργανωτική** (π.χ. ανασχεδιασμός υπηρεσιών)
- **Κοινωνική** (π.χ. απόλυση προσωπικού)
- **Οικονομική** (π.χ. αγορά υλικού, λογισμικού υπηρεσιών, αγαθών κλπ.)

Ανάλυση Απαιτήσεων (1/4)

- Ποιες είναι οι βασικές λειτουργίες του συστήματος;
- Υπάρχουν ειδικές απαιτήσεις;
- Ποια είναι τα κριτήρια αποδοχής των διαφόρων προϊόντων;

Ζητούμενο: Περιγραφή του *τι* πρέπει να κάνει το σύστημα, ανεξάρτητα από την τεχνολογία υλοποίησης.

Παραδείγματα ανάλυσης απαιτήσεων:

- ***Τι θα κάνει το σύστημα***: Μεταφορά ενός πακέτου από την Αθήνα στα Ιωάννινα με αεροπλάνο
- ***Τι θα κάνει το σύστημα***: Έκδοση ενός τιμολογίου με χρήση Η/Υ.

Ανάλυση Απαιτήσεων (2/4)

Η ανάγκη προσδιορισμού των απαιτήσεων, μέσα στα πλαίσια του σχεδιασμού και της ανάπτυξης ενός ολοκληρωμένου Π.Σ. αναφέρεται σε δύο επίπεδα:

1. Σε **ολόκληρο το κυρίως σύστημα**, οπότε πρέπει να εξετασθεί η συνολική δομή του οργανισμού / επιχείρησης.

- Προσδιορισμός της δομής του και καθορισμός των επιμέρους εφαρμογών, στις οποίες αναλύεται το σύστημα.
- Καθορισμός των διεπαφών μεταξύ των εφαρμογών, αφού κάθε εφαρμογή μπορεί να θεωρηθεί ως ένα υποσύστημα του κυρίως συστήματος.

2. Στον **αναλυτικό προσδιορισμό των απαιτήσεων των επιμέρους εφαρμογών**, που υποστηρίζει το Π.Σ., οπότε πρέπει να εξετασθούν όλες οι λειτουργίες που υλοποιούνται και όλα τα παραδοτέα που παράγονται από τα επιμέρους κομμάτια του συστήματος.

πχ.Πληροφοριακό Σύστημα SuperMarket – Διάρθρωση κυρίως συστήματος που αποτελείται από Σύστημα Ταμείου, Αποθήκης κλπ αλλά και επιμέρους απαιτήσεις

Ανάλυση Απαιτήσεων (3/4)

3. Η ανάλυση απαιτήσεων περιέχει πληροφορίες για το χρήστη, το σχεδιαστή, τον άνθρωπο που υλοποιεί το σύστημα και τέλος, τον ελεγκτή της υλοποίησης. Οι πληροφορίες μεταξύ άλλων αναφέρονται:

- στις λειτουργίες που πρέπει να εκτελεί το σύστημα
- στο πλαίσιο, τους περιορισμούς και τις υποθέσεις που γίνονται.
- στην απόδοση του συστήματος.
- στις ποσοτικές μετρήσεις και τα κριτήρια ελέγχου, ώστε να εξασφαλίζεται η ζητούμενη απόδοση.

4. Οι απαιτήσεις διατυπώνονται με τη βοήθεια **διαγραμματικών**, κυρίως, **τεχνικών** και άλλων εργαλείων, που υποστηρίζουν την απρόσκοπτη επικοινωνία μεταξύ των εμπλεκομένων φορέων, ενώ παράλληλα διαθέτουν την απαιτούμενη αυστηρότητα (φορμαλισμό) για την ακρίβεια της περιγραφής.

Ανάλυση Απαιτήσεων (4/4)

Η συλλογή στοιχείων γίνεται μέσω:

- Παρατήρησης
- Συνεντεύξεων
- Ερωτηματολογίων
- Εξέτασης των αρχείων και του υλικού τεκμηρίωσης
- Δειγματοληψίας

Η ανάλυση παράγει απαιτήσεις ανεξάρτητες από την τεχνολογία υλοποίησης, με την έννοια ότι αυτές δεν αναφέρονται και δεν καθορίζουν τον τρόπο με τον οποίο θα υλοποιηθεί ή θα λειτουργεί το σύστημα, αλλά τι θα κάνει, ανεξάρτητα από τον τρόπο υλοποίησης που θα επιλεγεί.

Σχεδιασμός Συστήματος (1/2)

- Ποια θα είναι η δομή του συστήματος;
- Ποιος θα είναι ο εξοπλισμός σε υλικό και λογισμικό;
- Ποιες διαδικασίες απαιτούνται;
- Με ποιον τρόπο θα πραγματοποιηθούν οι δοκιμές ελέγχου;

Ζητούμενο: Αναλυτική περιγραφή του πως θα είναι το σύστημα. Τεχνικές προδιαγραφές για το υλικό και το λογισμικό που θα χρησιμοποιηθεί.

Σχεδιασμός Συστήματος (2/2)

Ο σχεδιασμός ενός Π.Σ. πραγματοποιείται σε δύο στάδια:

στο **λογικό σχεδιασμό**, που ξεκινάει επικυρώνοντας το μοντέλο του νέου συστήματος, που έχει δημιουργηθεί στο τέλος της φάσης της ανάλυσης και

στο **φυσικό σχεδιασμό**, που αφορά το σχεδιασμό του Π.Σ. για θέματα σχετικά με τον τύπο και τη μορφή των δεδομένων, εκθέσεων, αναφορών, φυσικού σχεδιασμού του λογισμικού, κλπ

Ο λογικός σχεδιασμός ασχολείται κυρίως με τα εξής σημεία:

- τους στόχους/περιορισμούς του συστήματος.
- εναλλακτικούς τρόπους φυσικού σχεδιασμού.
- τα διάφορα θέματα σχεδιασμού: πλατφόρμες υλοποίησης, αρχιτεκτονικές για Βάσεις Δεδομένων, αρχεία, επικοινωνίες, δίκτυα, κ.λπ.
- τα χρονοδιαγράμματα.
- τις εκτιμήσεις επικοινωνιακού/λειτουργικού φόρτου.
- την τυποποίηση και τους ελέγχους που πιθανόν να υιοθετηθούν.

Υλοποίηση – Έλεγχος – Εγκατάσταση (1/2)

- Ανάπτυξη του λογισμικού
- Έλεγχος του λογισμικού – Επικύρωση (validation)
- Εγκατάσταση του υλικού (υπολογιστές κλπ)
- Έλεγχος καλής λειτουργίας του υλικού
- Εγκατάσταση του λογισμικού
- Έλεγχος καλής λειτουργίας του λογισμικού

Ζητούμενο: Τεκμηρίωση του υλικού, του λογισμικού και των διαδικασιών που υλοποιήθηκαν.

Υλοποίηση – Έλεγχος – Εγκατάσταση (2/2)

Μετά από τη φάση του σχεδιασμού ακολουθούν διάφορες διαδικασίες που οδηγούν στην υλοποίηση και εγκατάσταση του νέου συστήματος

- Αγορά και εγκατάσταση νέων συστημάτων υλικού και λογισμικού
- Ανάπτυξη προγραμμάτων
- **Έλεγχος ποιότητας**
- Τεκμηρίωση
- Μετάπτωση
- Εκπαίδευση & κατάρτιση

Ερώτημα - Άσκηση

Τι σημαίνει «καλή ποιότητα» για ένα πληροφοριακό σύστημα;

Απάντηση

Τα πληροφοριακά συστήματα έχουν ορισμένα χαρακτηριστικά που αντανακλούν την **ποιότητά** τους.

Τα γνωρίσματα αυτά δεν αναφέρονται άμεσα στο τι κάνει το σύστημα, αλλά στη συμπεριφορά που έχει όταν εκτελείται καθώς και στη δομή και στην οργάνωση του πηγαίου κώδικα και της σχετικής τεκμηρίωσης.

Παράδειγμα τέτοιων χαρακτηριστικών, (**τα οποία μερικές φορές λέγοντα μη λειτουργικά χαρακτηριστικά**), είναι για παράδειγμα ο χρόνος απόκρισης του λογισμικού σε ερωτήματα του χρήστη.

Ποιότητα Λογισμικού

Ύπαρξη προτύπων και μετρικών που μας επιτρέπουν να ελέγξουμε την ποιότητα του λογισμικού. Ενδεικτικά μια προσέγγιση που αφορά στην αξιολόγηση τεχνικών χαρακτηριστικών και λειτουργιών του λογισμικού είναι:

Λειτουργικότητα (functionality)

καταλληλότητα (suitability),

αξιοπιστία (reliability), δλδ. ανοχή βλαβών (fault tolerance), δυνατότητα ανάκαμψης (recoverability)

αποδοτικότητα (efficiency), δλδ. χρόνος απόκρισης (time behavior), συμπεριφορά πόρων (resource behavior)

ευχρηστία (usability),

ασφάλεια (security)

Δυνατότητες υποστήριξης (maintainability)

Συμβατότητα (compatibility) δλδ. portability, reusability κλπ

Λειτουργία-Συντήρηση

- Τι προσθήκες, αλλαγές, τροποποιήσεις και βελτιώσεις απαιτούνται;

Ζητούμενο: Προσπάθεια για ομαλή λειτουργία και συνεχή βελτίωση.

Το σύστημα προσαρμόζεται σε ένα δυναμικό επιχειρηματικό περιβάλλον

Βελτιώνεται προκειμένου να καλύψει τις νέες απαιτήσεις των χρηστών

Ερώτημα - Άσκηση

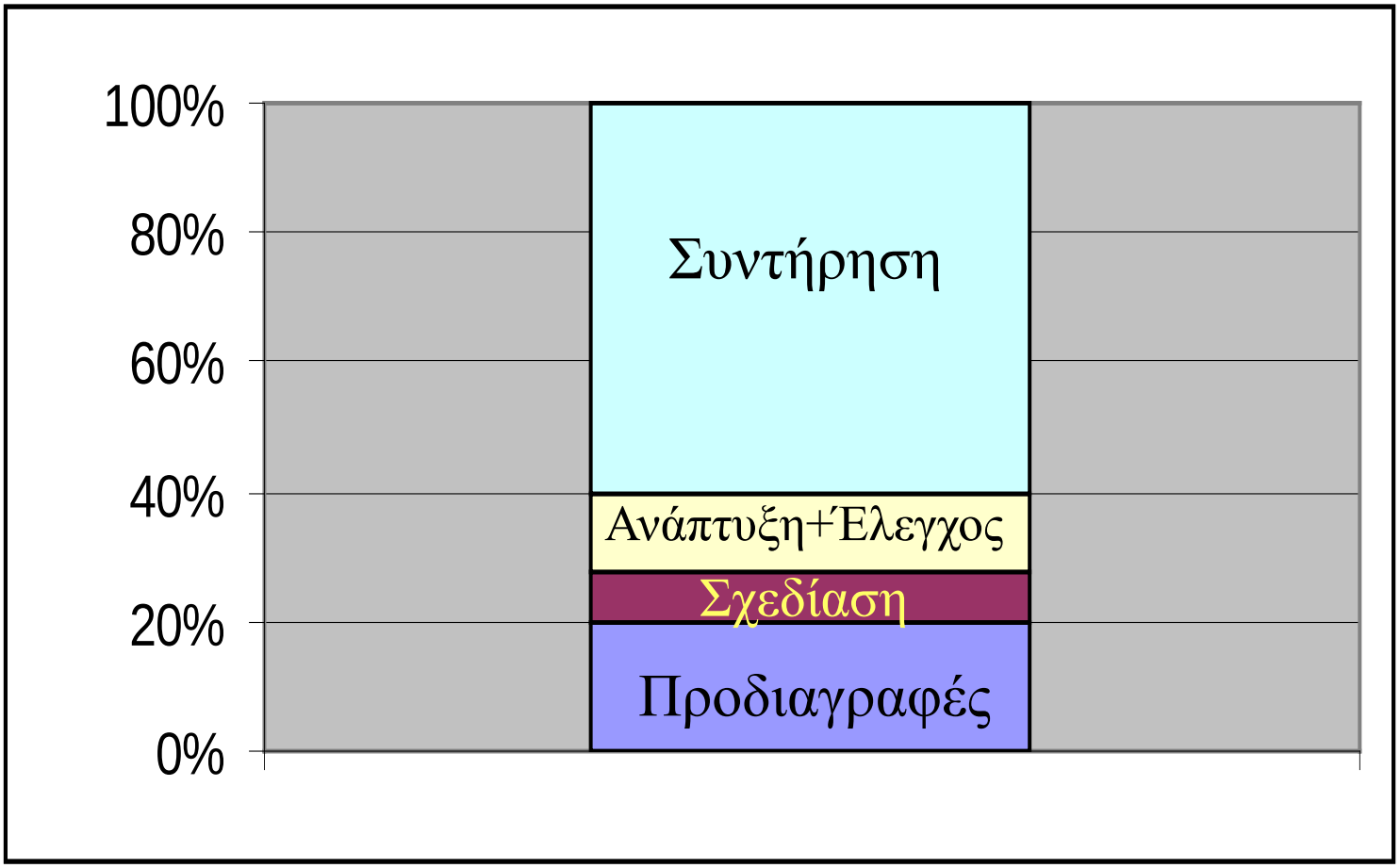
Μιλήσαμε για τον όρο συντήρηση πληροφοριακού συστήματος (information system maintenance)

Πάρτε ως παράδειγμα μια εφαρμογή μισθοδοσίας και πέστε μου σε τι αφορά

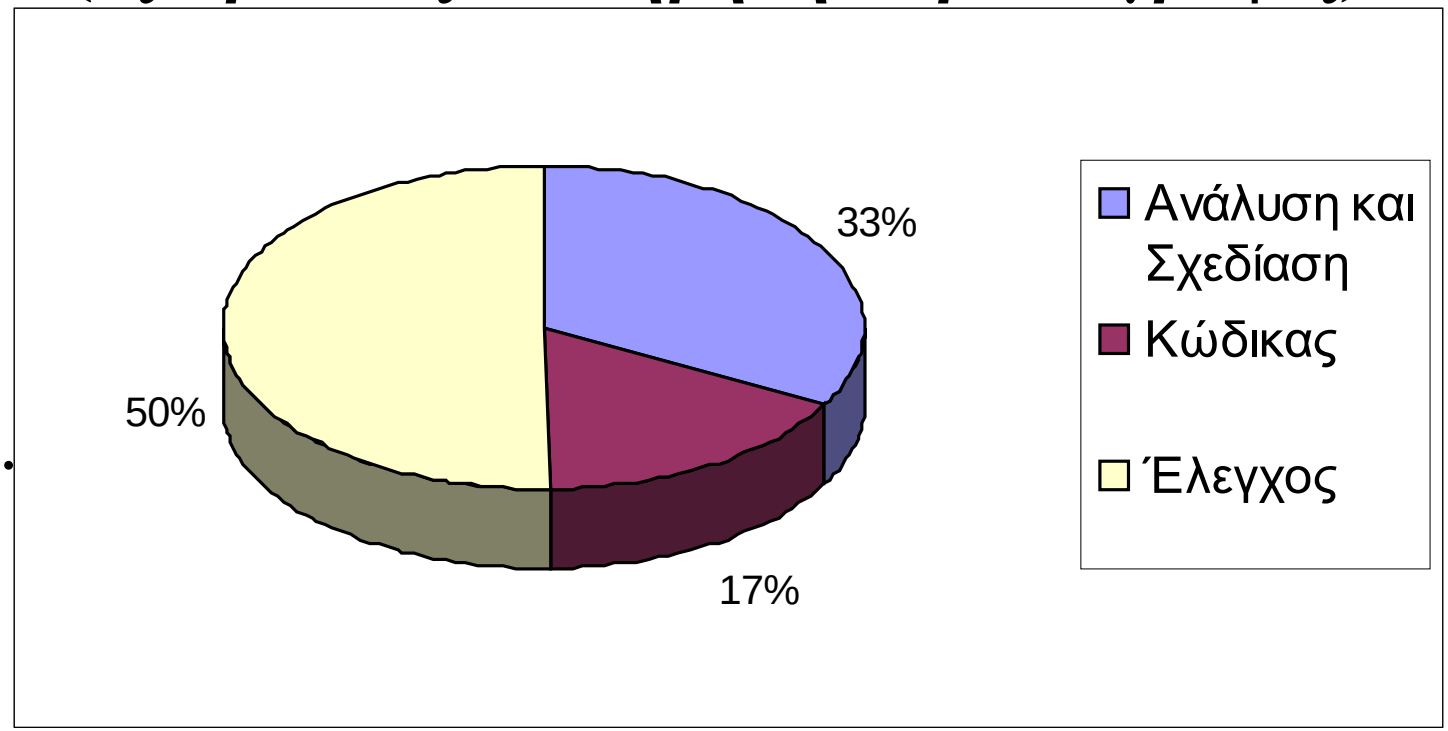
Απάντηση

Αφορά	Παράδειγμα
Διόρθωση σφαλμάτων	Το πρόγραμμα αρνείται να λειτουργήσει κάτω από συγκεκριμένες συνθήκες και παράγει ένα λάθος κατά την εκτέλεση.
Βελτιστοποίηση της απόδοσης •	Το πρόγραμμα παράγει τα επιθυμητά αποτελέσματα σε χρόνο που μπορεί να βελτιωθεί (για τον ίδιο υπολογιστή).
Αυτοματοποίηση / ενσωμάτωση νέων εργασιών	Διαπιστώνεται ότι υπάρχει μια εργασία η οποία θα ήταν χρήσιμο να προστεθεί σε αυτές που εκτελεί το πρόγραμμα.
Ενσωμάτωση μεταβολών από τον πραγματικό κόσμο	Αλλάζει το νομικό πλαίσιο της μισθοδοσίας (ή της τήρησης «βιβλίων και στοιχείων» ή καταργείται) και οι αλλαγές αυτές πρέπει να ενσωματωθούν στο λογισμικό

Κατανομή κόστους πληροφοριακού συστήματος



Κατανομή κόστους πληροφοριακού συστήματος
(εξαιρώντας συντήρηση - προδιαγραφές)



Το πρόβλημα δεν είναι που είναι τα λάθη, αλλά πόσο κοστίζει η διόρθωσή τους

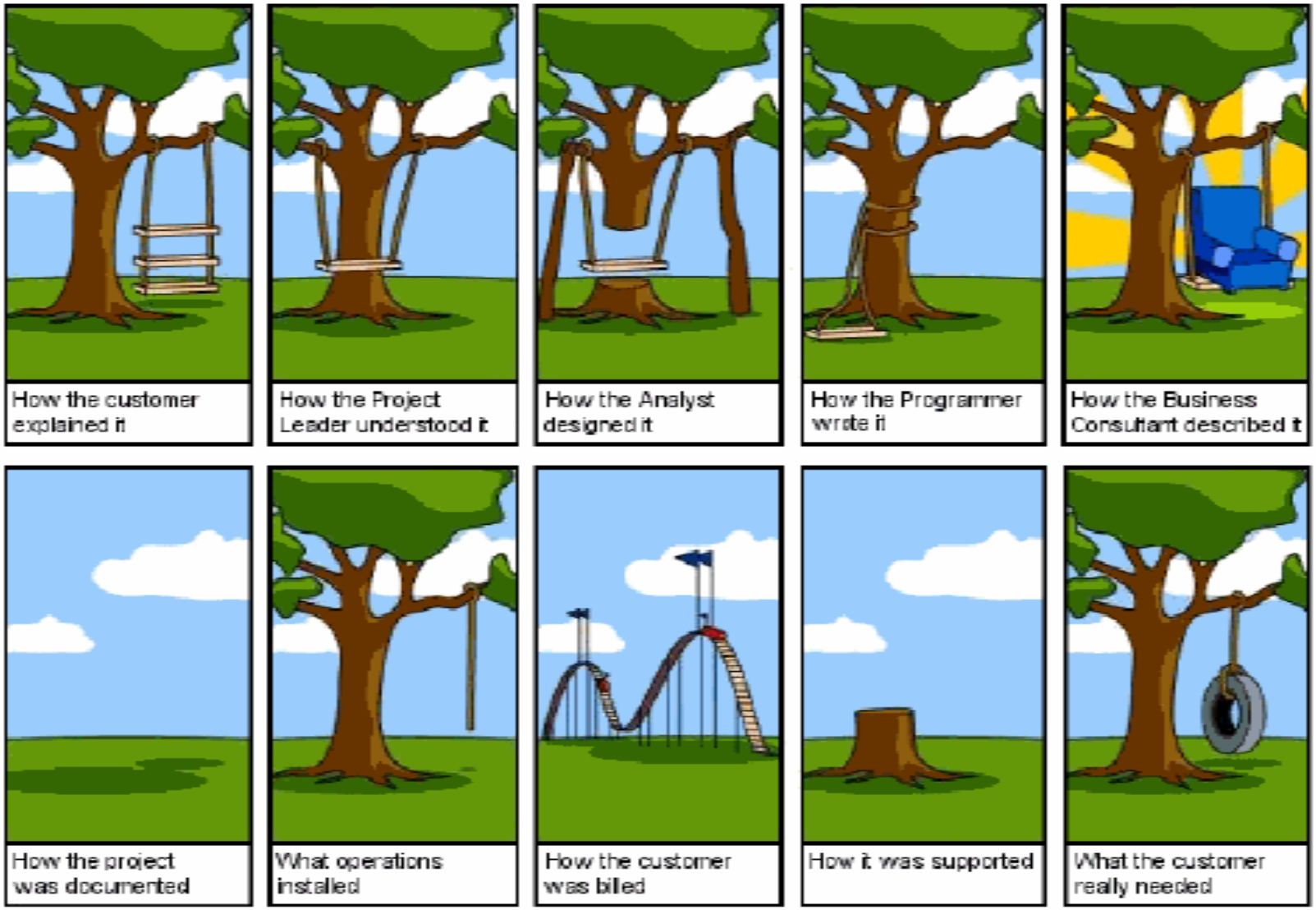
Το κόστος του λάθους αυξάνει όσο περισσότερο δεν εντοπίζεται

Μεθοδολογίες

Η χρήση μιας μεθοδολογίας βελτιώνει την πρακτική της ανάπτυξης πληροφοριακών συστημάτων και περιέχει:

- Μια σειρά **φάσεων** που αρχίζουν από τη μελέτη σκοπιμότητας και φτάνουν μέχρι και τη συντήρηση του συστήματος
- Μια σειρά **τεχνικών,** που περιλαμβάνουν τους τρόπους για να αξιολογηθούν τα κόστη και τα οφέλη των διαφορετικών λύσεων, και τις μεθόδους για το λεπτομερή σχεδιασμό του συστήματος
- Μια σειρά **εργαλείων** που βοηθούν τους αναλυτές συστημάτων
- Ένα σχέδιο **κατάρτισης** έτσι ώστε οι αναλυτές και οι άλλοι συμμετέχοντες, να μπορούν να υιοθετήσουν τα προτεινόμενα πρότυπα.

Το ζήτημα της Τεχνολογίας ΠΣ είναι:



Τεχνοτροπίες Σχεδίασης Πληροφοριακών Συστημάτων / Λογισμικού

Το πρόβλημα της ανάπτυξης πληροφοριακών συστημάτων / λογισμικού μπορεί να αντιμετωπιστεί με διάφορες στρατηγικές προσεγγίσεις. Οι διάφορες μεθοδολογίες που έχουν παρουσιαστεί μπορούν να ενταχθούν σε δύο μεγάλες κατηγορίες:

τις *προσανατολισμένες στις διαδικασίες (function oriented) και* τις *προσανατολισμένες στα αντικείμενα (object oriented).*

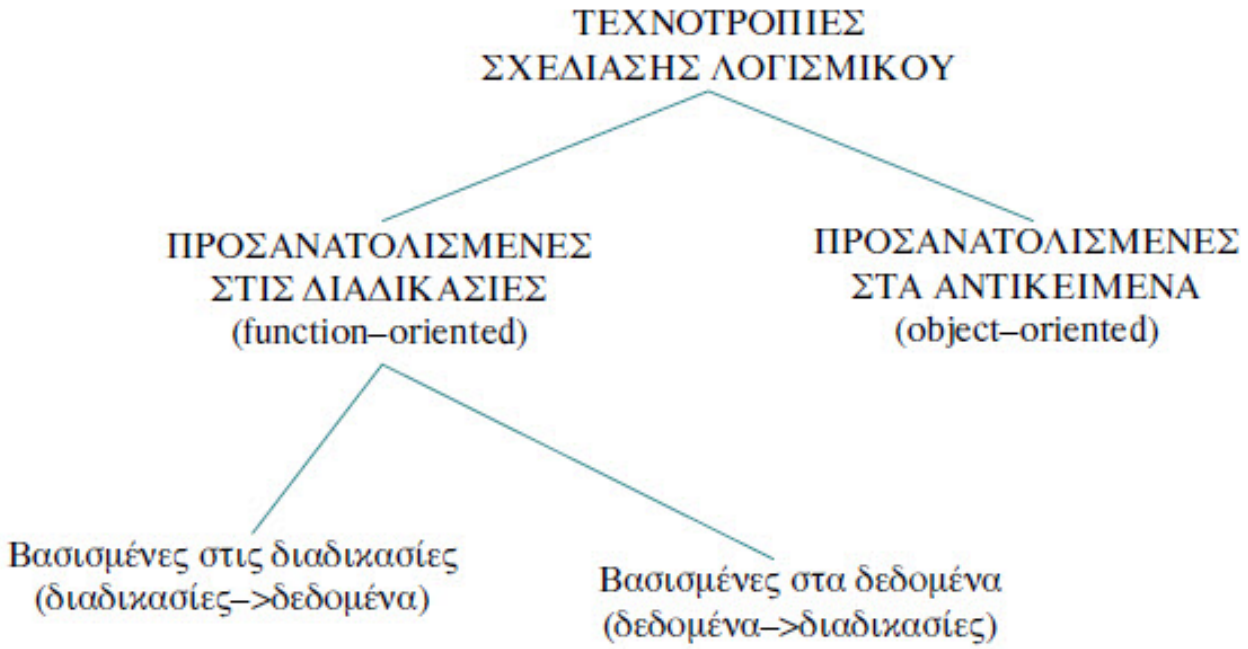
Η απάντηση στην ερώτηση «Τι θα κάνει το πληροφοριακό σύστημα / λογισμικό;» μπορεί να δοθεί με τη μορφή μιας ιεραρχίας διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού, η οποία επιδρά σε ένα σύνολο ανεξάρτητων δεδομένων είτε ως αλληλεπίδραση μεταξύ «αντικειμένων».

Τεχνοτροπίες Σχεδίασης Πληροφοριακών Συστημάτων / Λογισμικού

Η **δομημένη προσέγγιση** είναι βασισμένη σε δεδομένα και διαδικασίες που προκύπτουν από την αποσύνθεση του προβλήματος και τα οποία μελετώνται ανεξάρτητα και χρησιμοποιούνται για τη υλοποίηση της εφαρμογής λογισμικού.

Η **αντικειμενοστρεφής προσέγγιση (object oriented)** βασίζεται στην ιδέα ότι στον πραγματικό κόσμο δεδομένα και διαδικασίες μπορούν να ιδωθούν ενιαία με βάση το πεδίο ευθύνης κάποιων «αντικειμένων». Κάθε αντικείμενο παρέχει στο περιβάλλον του ένα σύνολο υπηρεσιών της ευθύνης του. Η συνεργασία του συνόλου των αντικειμένων του πεδίου μιας εφαρμογής λογισμικού παράγει το επιθυμητό αποτέλεσμα.

Τεχνοτροπίες Σχεδίασης



Η δομημένη προσέγγιση και οι Τεχνικές της (1/2)

Οι πρώτες μεθοδολογίες περιελάμβαναν τεχνικές όπως:

διαφόρων τύπων διαγράμματα ροής (π.χ. για να βοηθήσουν τον αναλυτή να κατανοήσει τη ροή των εγγράφων μέσα σε ένα τμήμα)

διαγράμματα οργάνωσης (π.χ. για να αναπαραστήσουν την ιεραρχία των υπαλλήλων σε μια επιχείρηση)

προδιαγραφές εγγράφων (παρέχουν τις λεπτομέρειες των εγγράφων που χρησιμοποιούνται στο σύστημα)

διαγράμματα πλέγματος (περιγράφουν πως αλληλεπιδρούν τα επιμέρους υποσυστήματα)

Η δομημένη προσέγγιση και οι Τεχνικές της (2/2)

Οι περισσότερες από τις παραπάνω τεχνικές έχουν αντικατασταθεί στις σύγχρονες μεθοδολογίες, οι οποίες συνήθως βασίζονται σε:

Διαγράμματα Ροής Δεδομένων

- ροές δεδομένων από το περιβάλλον στο σύστημα,
- ροές δεδομένων από το σύστημα στο περιβάλλον,
- διαδικασίες που αλλάζουν τα δεδομένα στα πλαίσια του συστήματος
- αποθήκευση δεδομένων στο σύστημα,
- τα όρια και την εμβέλεια του συστήματος

Διάγραμμα οντοτήτων – συσχετίσεων

Λεξικά δεδομένων και

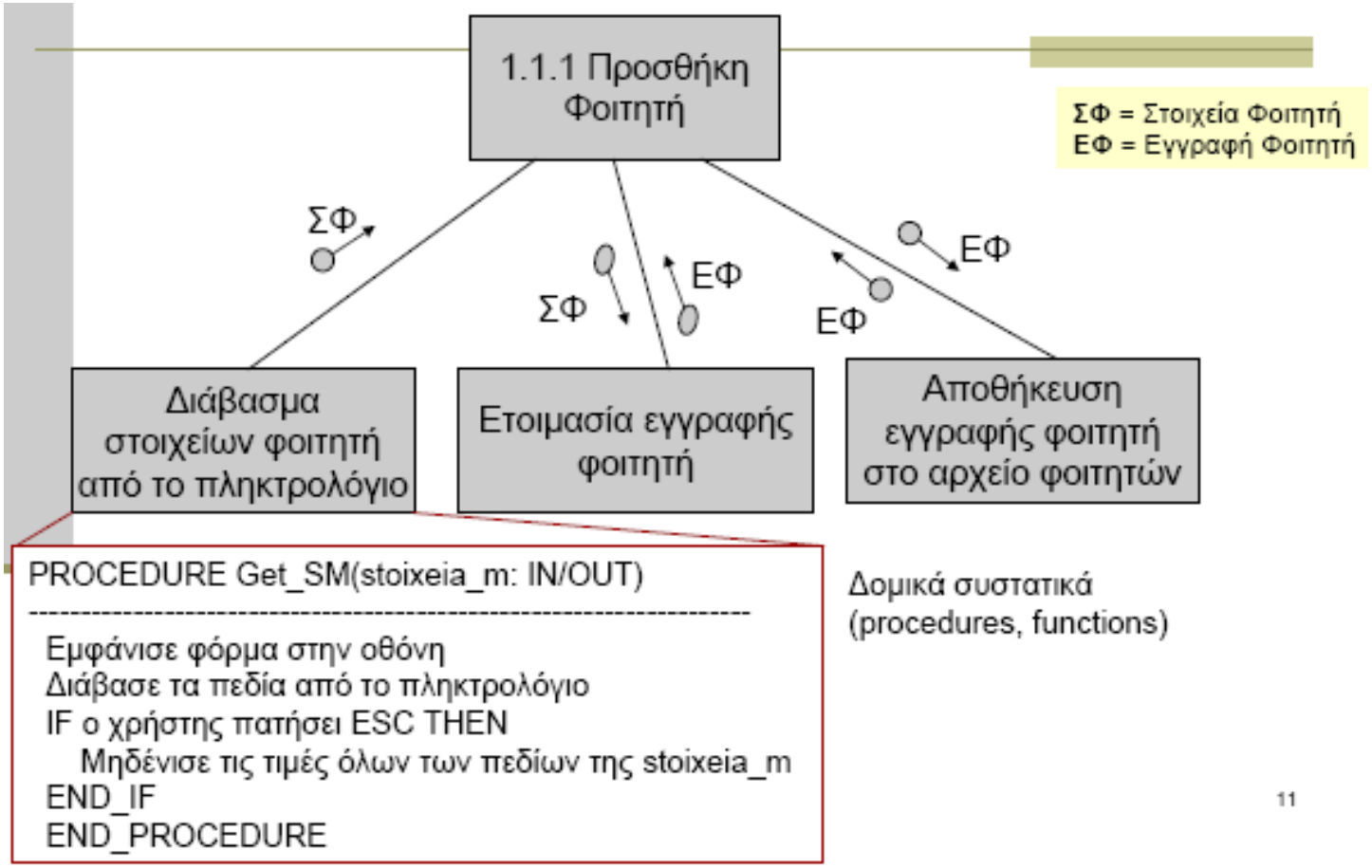
Διαγράμματα Δομής

Δομημένη προσέγγιση

Οι μεθοδολογίες που ακολουθούν αυτή την προσέγγιση προτείνουν τρόπους αποσύνθεσης του συστήματος από πάνω προς τα κάτω (top-down) σε μια ιεραρχία διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού.

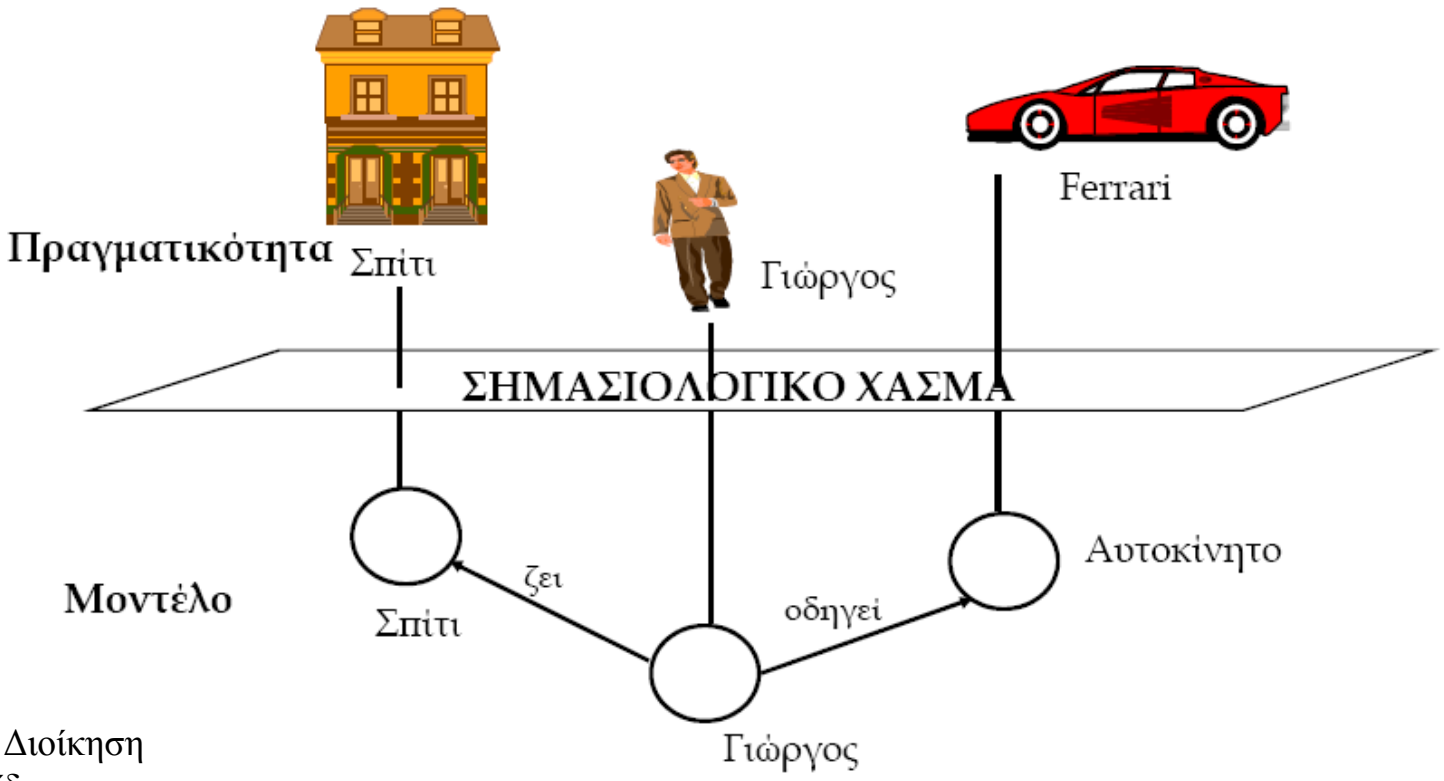
Όσο κατεβαίνει κανείς στην ιεραρχία αυτή, τόσο μεγαλύτερη λεπτομέρεια συναντά, μέχρις ότου φτάσει στις απλές δομικές μονάδες, δηλαδή τις εντολές της γλώσσας προγραμματισμού.

Δομημένη προσέγγιση (σχεδίαση)



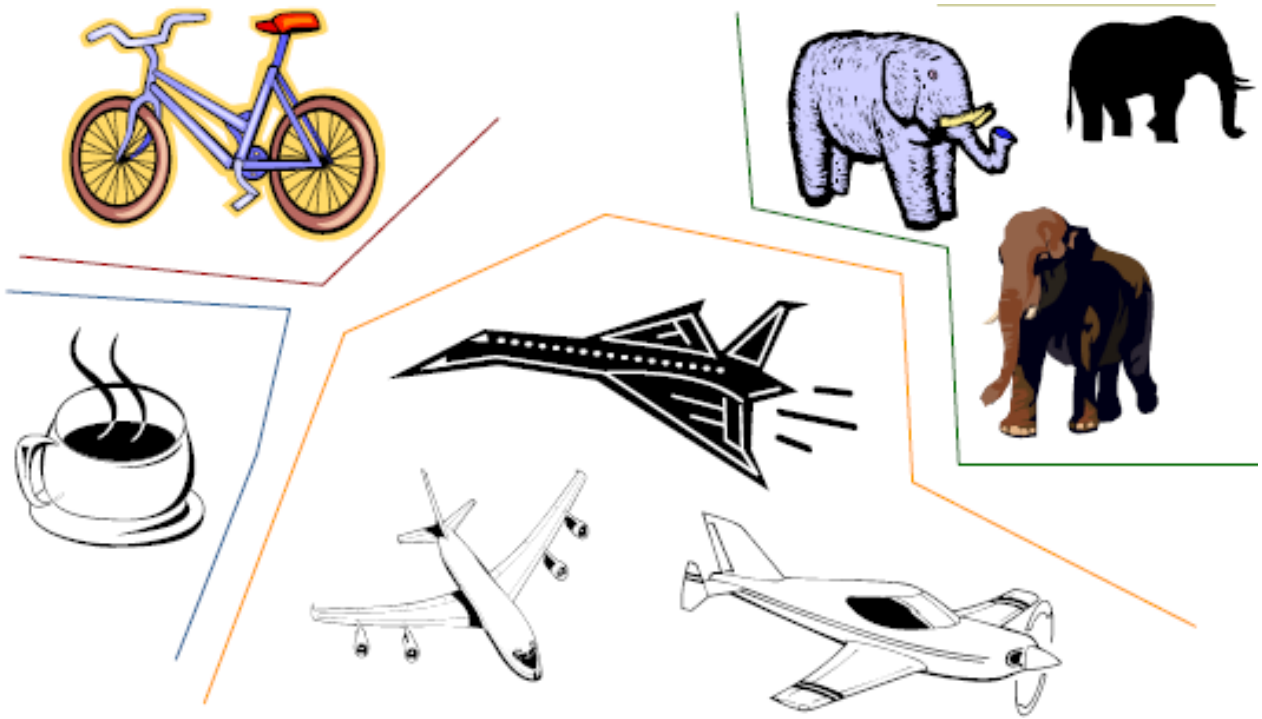
Αντικειμενοστρεφής προσέγγιση

Μοντελοποιεί τα αντικείμενα, τα οποία αναπαριστούν κάτι από τον πραγματικό κόσμο, συμπεριλαμβανομένων των ανθρώπων, των δεδομένων, και των διαδικασιών, καθώς και της αλληλεπίδρασης των αντικειμένων.



Αντικείμενο

Τα αντικείμενα είναι στοιχεία ή οντότητες που χρησιμοποιούνται για την αναπαράσταση του (πραγματικού) κόσμου που μας περιβάλλει (ποδήλατο, σκύλος).



Αντικείμενο λογισμικού: Μοντελοποιεί, αναπαριστά, αντιστοιχεί σε κάποιο αντικείμενο του πραγματικού κόσμου.

Αντικείμενο λογισμικού

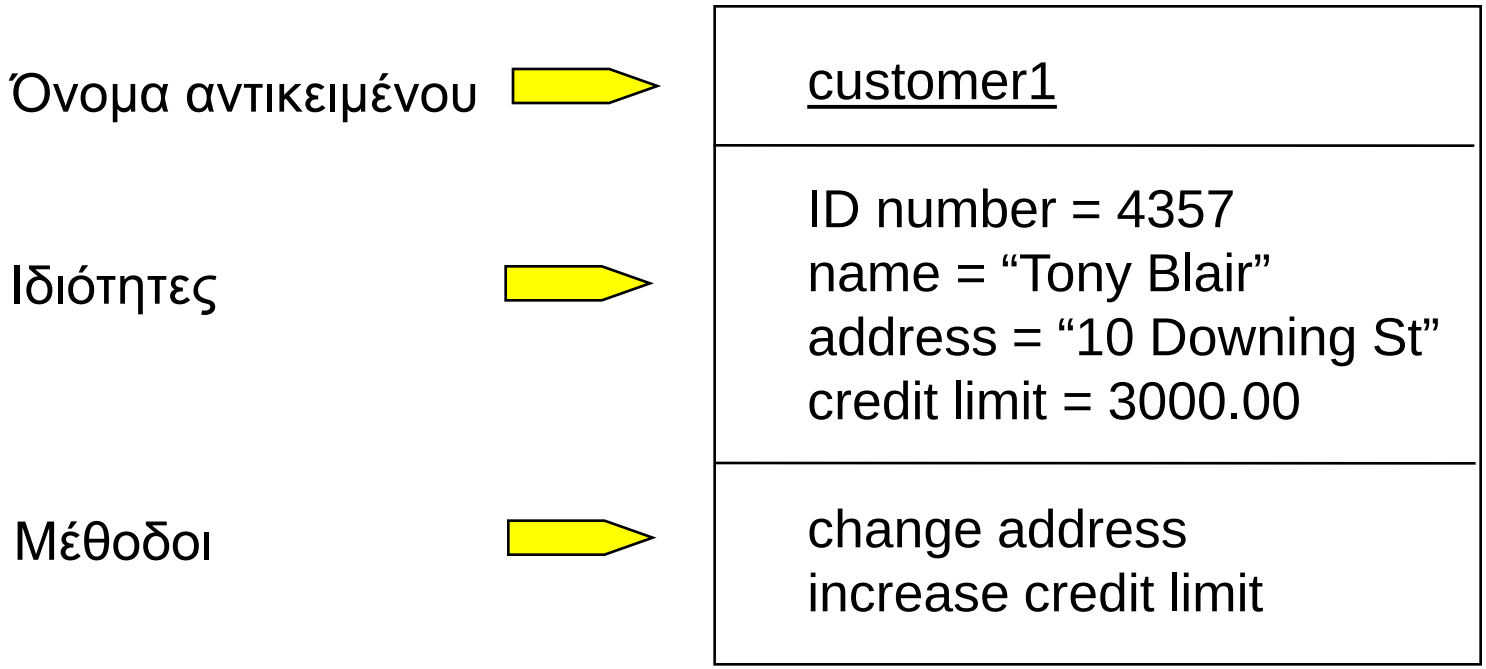
Αντικείμενο λογισμικού: Μοντελοποιεί, αναπαριστά, αντιστοιχεί επίσης και σε άλλες έννοιες του πραγματικού κόσμου όπως δομές δεδομένων (π.χ. λίστα, ουρά, κλπ.) ή σε κάτι πιο αφαιρετικό.

Τα αντικείμενα, έχουν **χαρακτηριστικά και κατάσταση** (που δηλώνονται με μεταβλητές) καθώς και **συμπεριφορά** (που υλοποιείται με μεθόδους).

Ένα αντικείμενο μοντελοποιείται με **ιδιότητες** ή **πεδία** (attributes) και **μεθόδους / λειτουργίες** (methods / operations), που **ανήκουν στο αντικείμενο**.

Παράδειγμα αντικειμένου

Για παράδειγμα, ένα αντικείμενο που παριστάνει έναν πελάτη μπορεί να μοντελοποιηθεί ως εξής:



Ποιάς εφαρμογής αντικείμενο θα μπορούσε να είναι το παραπάνω;

Κλάση ή Τάξη (Class)

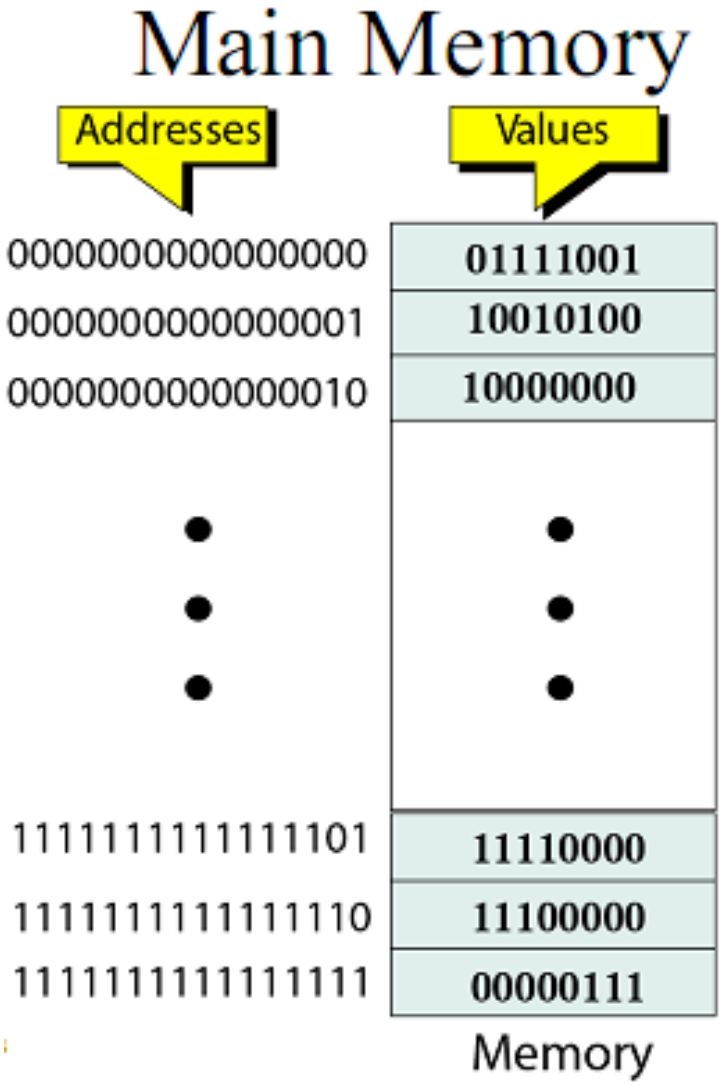
Μια συλλογή από παρόμοια αντικείμενα ονομάζεται **κλάση** (class).
Όλα τα αντικείμενα μιας κλάσης έχουν τα ίδια χαρακτηριστικά και τις ίδιες μεθόδους, αλλά οι **τιμές** των χαρακτηριστικών (είναι δυνατό να) διαφέρουν.
Μια κλάση είναι ένα **εργοστάσιο** (factory) που παράγει αντικείμενα.
Ένα συγκεκριμένο αντικείμενο αποτελεί **στιγμιότυπο (instance)** μιας κλάσης.
Ο μηχανισμός της κλάσης επιτρέπει τη δημιουργία νέων τύπων.
Είναι ένα **περίγραμμα** / μια **φόρμα** (template) για τον προσδιορισμό των ιδιοτήτων και των μεθόδων ενός συνόλου από αντικείμενα.
Η σχέση ενός αντικειμένου με μια κλάση είναι παρόμοια με τη σχέση μιας μεταβλητής με τον τύπο της μεταβλητής σε μια γλώσσα προγραμματισμού.

Πριν δούμε το παράδειγμα ενός μοντέλου μιας κλάσης

Μεταβλητή

Είναι μια υπολογιστική οντότητα που **παριστάνει ένα μέγεθος** (έννοια) που σχετίζεται με το πρόβλημα που λύνει μια εφαρμογή και έχει ένα συγκεκριμένο **τύπο** δεδομένων.

Τα δεδομένα που εισάγονται / επεξεργάζονται σε ένα πληροφοριακό σύστημα αποθηκεύονται (προσωρινά) στη μνήμη (RAM)



Τύποι δεδομένων

Απλοί

Ακέραιος – int (integer),
Πραγματικός – float, double (floating point),
Χαρακτήρας – char (character),
Λογικός – boolean
void (no value)

Σύνθετοι

Πίνακας – Array (σύνολο ομοειδών τιμών)
Πίνακας 4 θέσεων ακεραίων είναι ο: [7,9,10,3]
Πίνακας με 3 γραμμές και 4 στήλες επίσης ακεραίων είναι ο:

a[i][j] (γρ,στ)

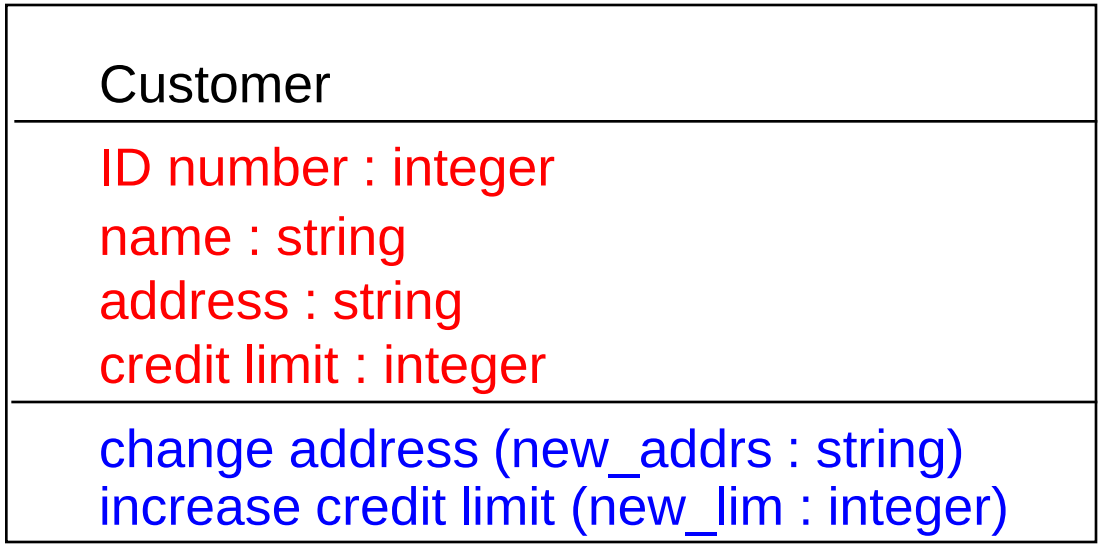
$$\begin{pmatrix} 7 & 9 & 10 & 3 \\ 9 & -1 & 8 & 5 \\ -1 & 7 & 7 & 9 \end{pmatrix}$$

Συμβολοσειρά – String (πίνακας χαρακτήρων)

Κλάση ή Τάξη (Class)

Μια κλάση μοντελοποιείται:

- Το όνομα της κλάσης.
- Το όνομα (και τον τύπο) κάθε ιδιότητας.
- Το όνομα (και τις παραμέτρους) κάθε μεθόδου.

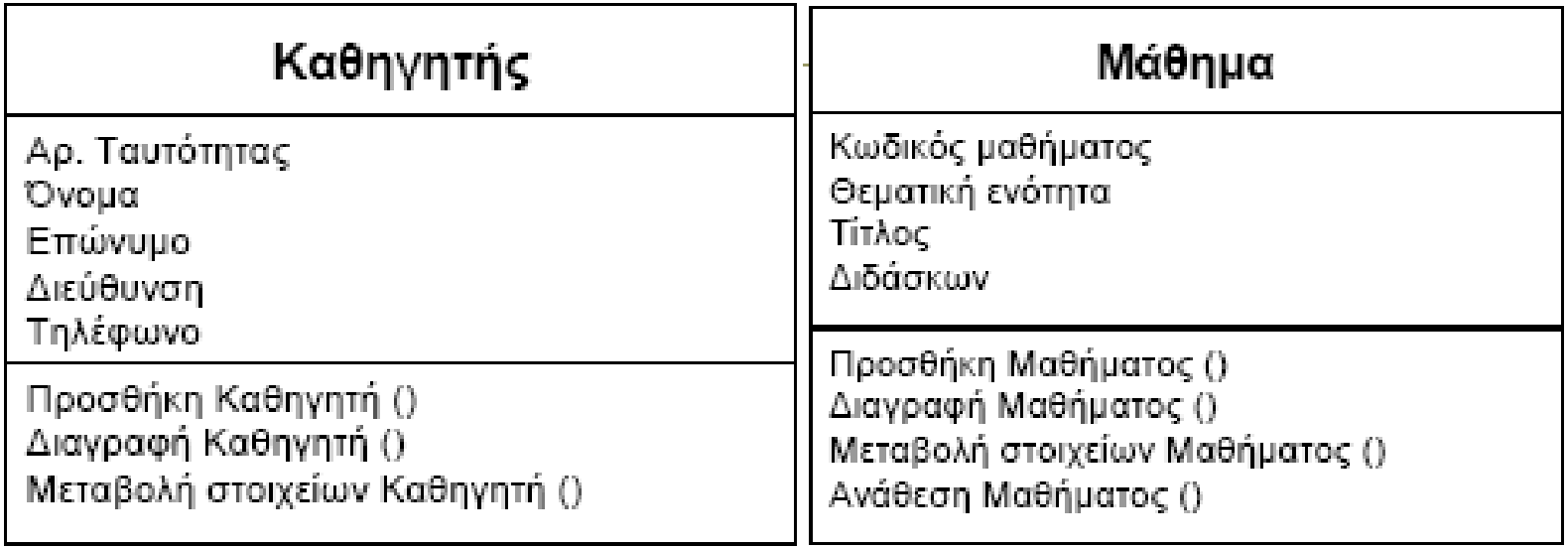


Άσκηση

Δημιουργήστε τις κλάσεις Καθηγητής, Μάθημα (περιλαμβάνοντας ιδιότητες (attributes) και μεθόδους (methods)) για μια εφαρμογή e-Μαθητολογίου.

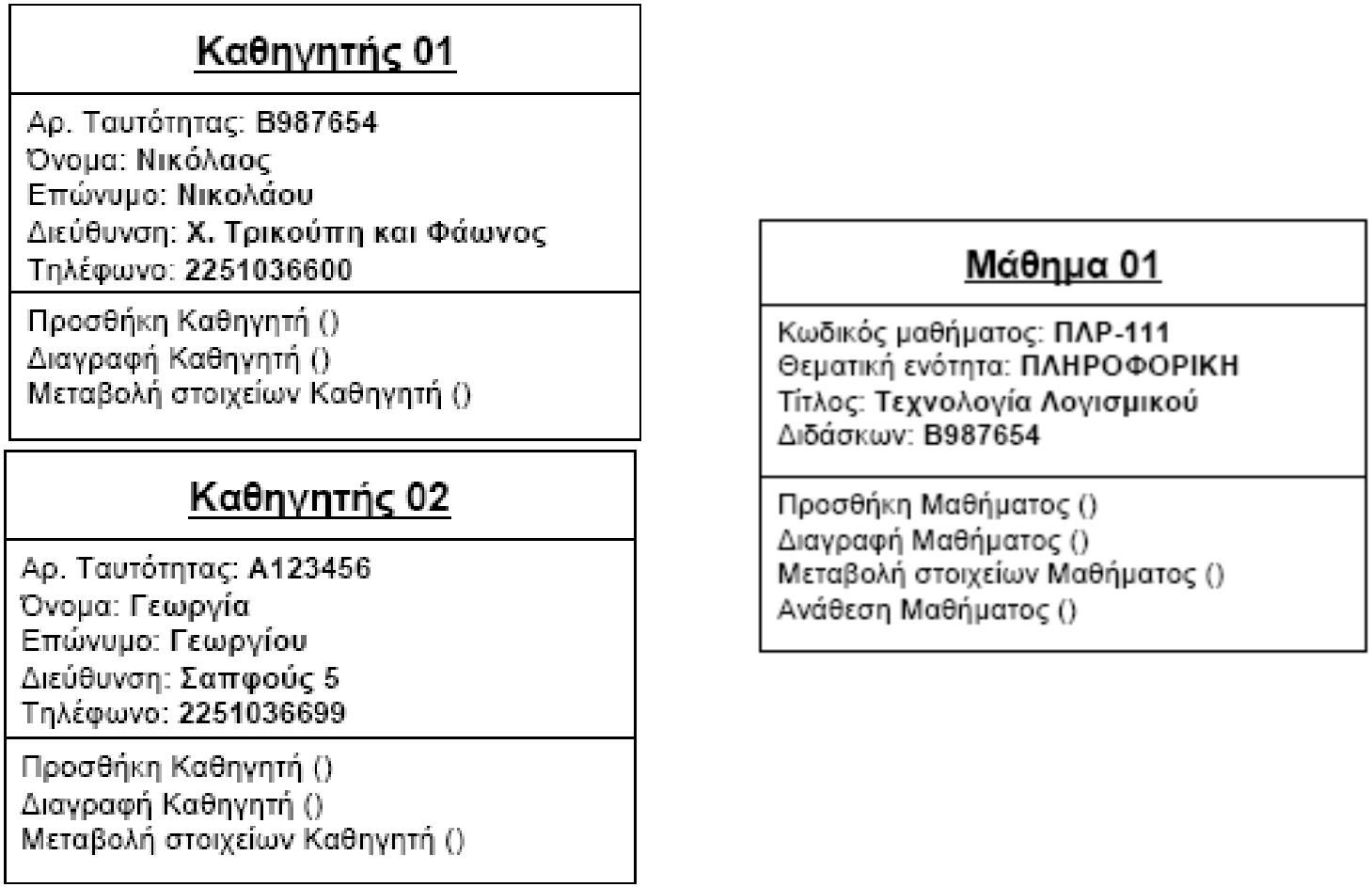
Αντικειμενοστρεφής Σχεδίαση

Απάντηση

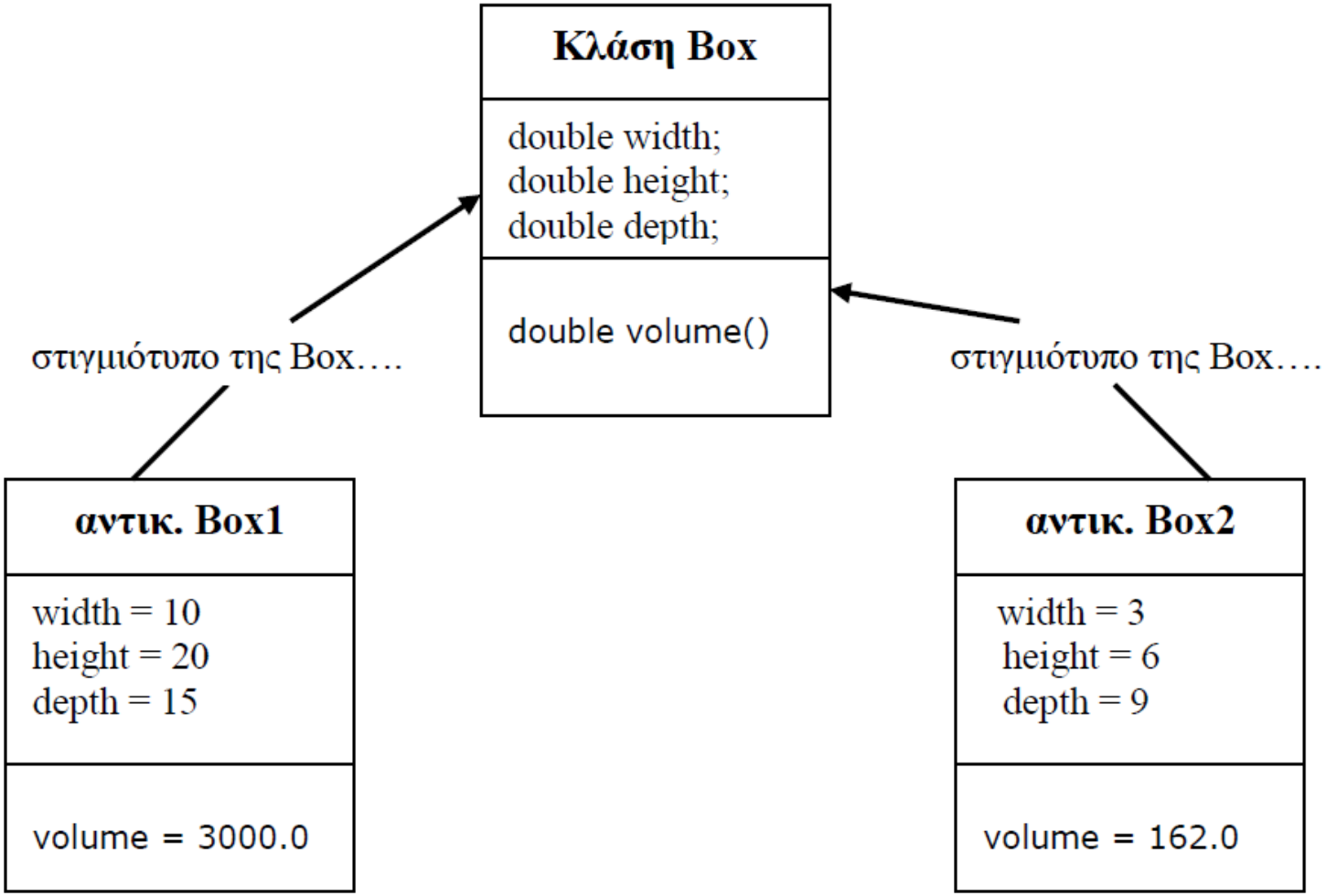


Αντικειμενοστρεφής Σχεδίαση

Στιγμιότυπα των κλάσεων Καθηγητής, Μάθημα



Αντικειμενοστρεφής Σχεδίαση



Unified Modeling Language (UML)

Η UML είναι μία γραφική γλώσσα μοντελοποίησης για την ανάλυση και τη σχεδίαση πληροφοριακών συστημάτων / λογισμικού με χρήση αντικειμενοστραφών τεχνικών.

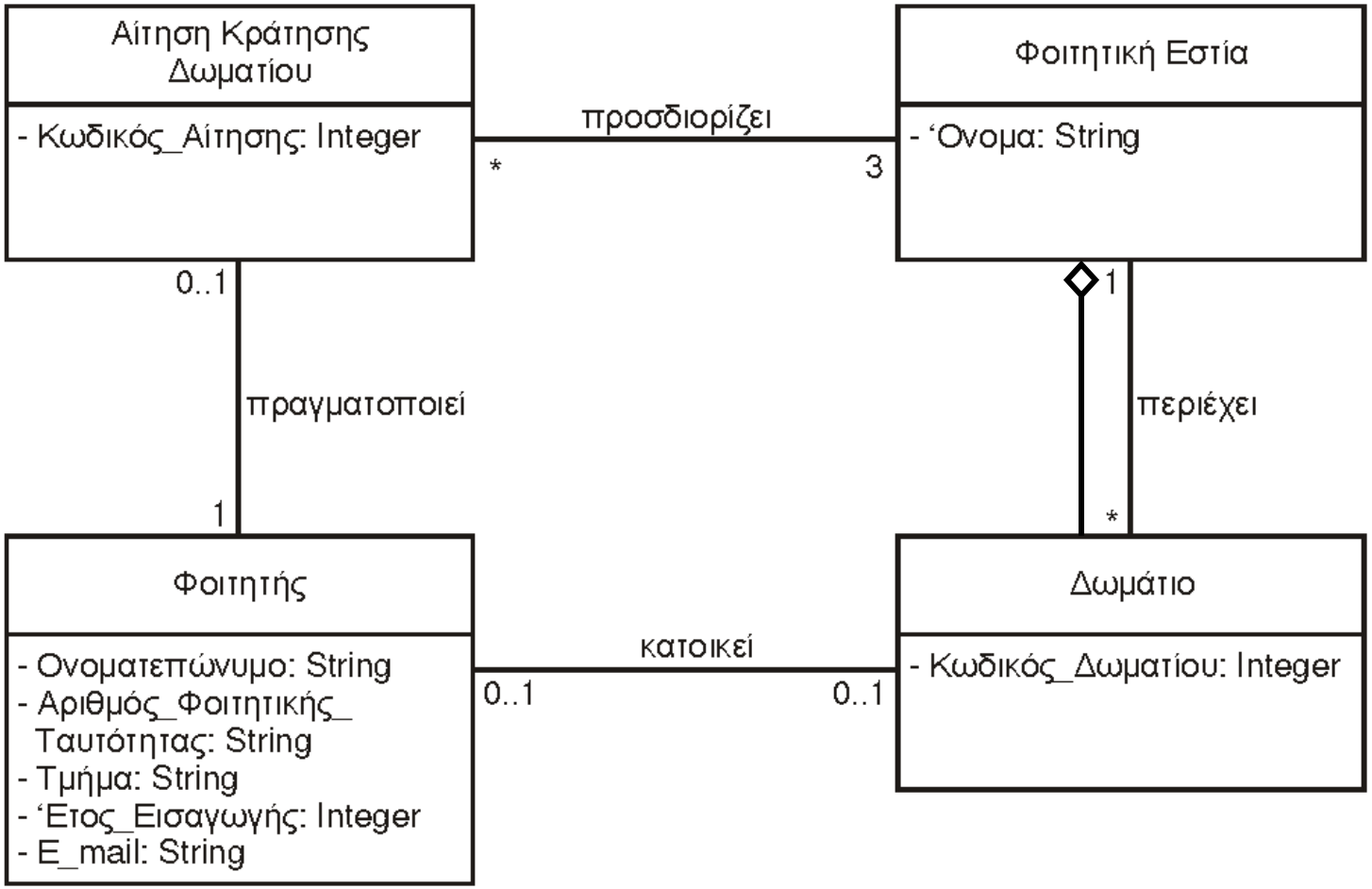
Είναι ένα σύνολο κανόνων και σημασιολογιών που μπορούν να χρησιμοποιηθούν για να περιγράψουν τόσο τη δομή όσο και τη λογική ενός πληροφοριακού συστήματος.

Η UML δεν είναι γλώσσα προγραμματισμού.

Σήμερα, υποστηρίζεται από το OMG (Object Management Group) και αποτελεί πρότυπο.

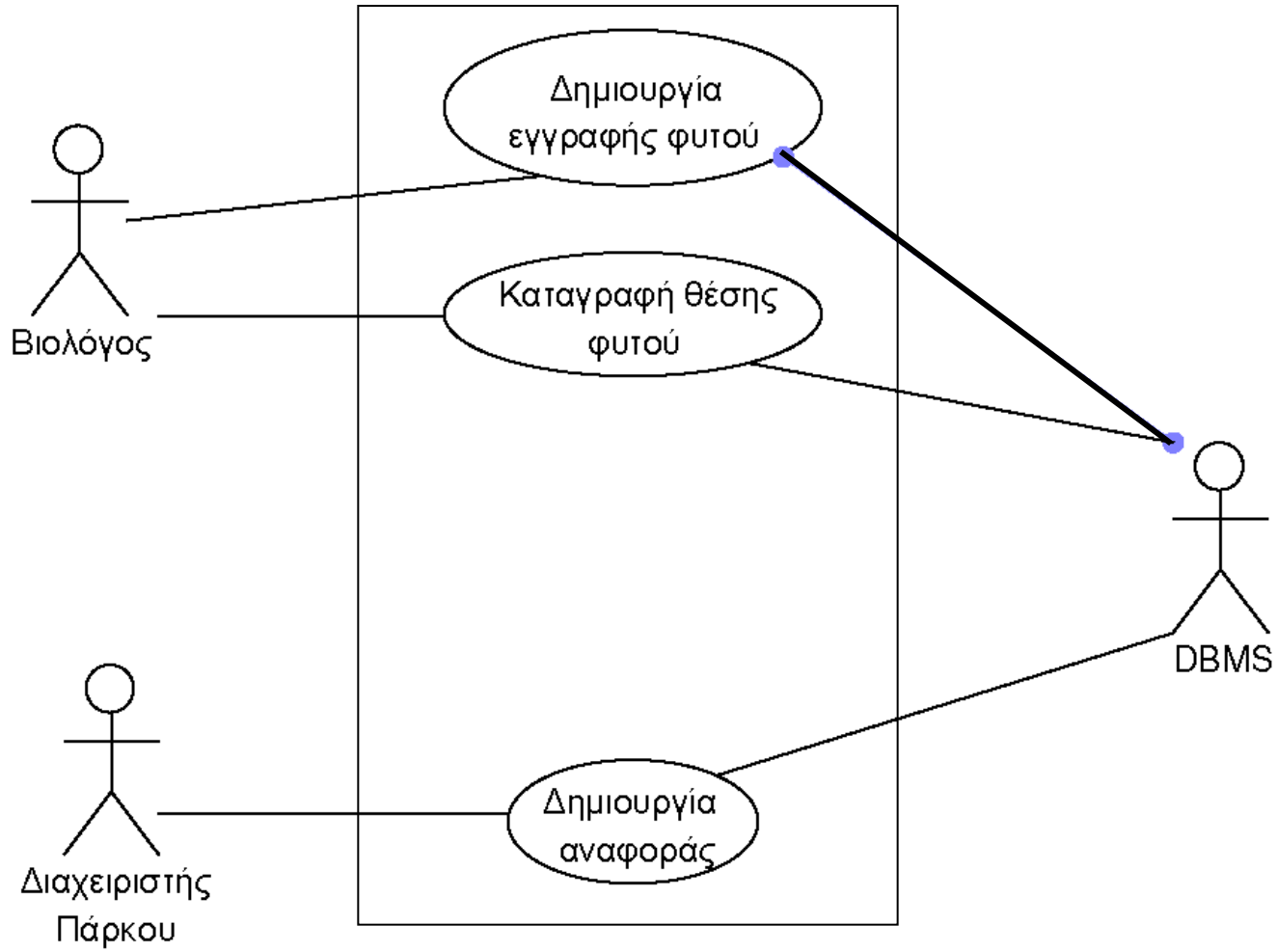
Διάγραμμα Κλάσεων

Σχεδιασμός εφαρμογής Web για την κράτηση δωματίου σε φοιτητικές εστίες από φοιτητές ενός πανεπιστημίου.



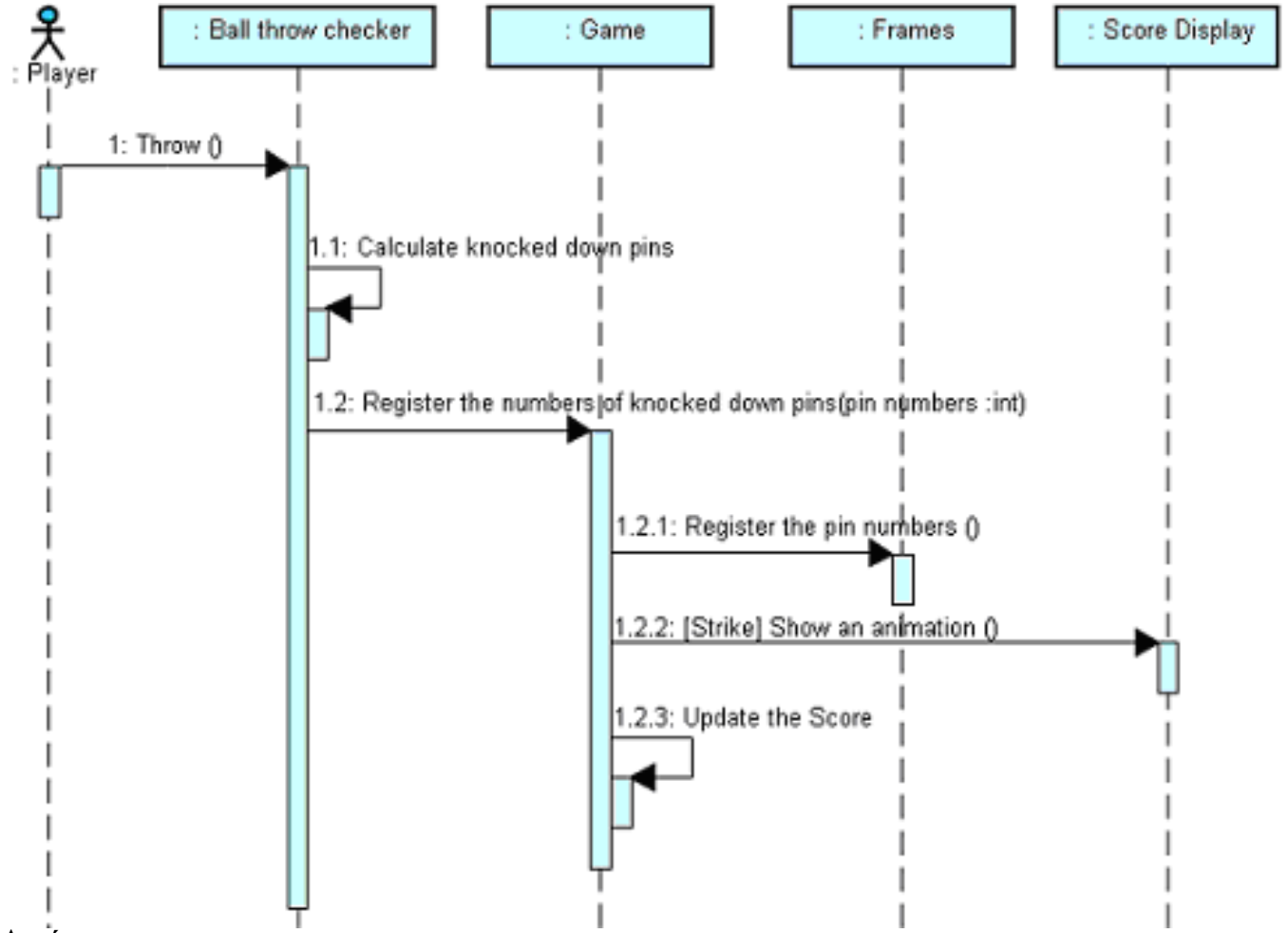
Διάγραμμα Περιπτώσεων Χρήσης

Σχεδιασμός εφαρμογής που θα επιτρέπει σε βιολόγους να καταγράφουν τη θέση που έχουν φυτά σε ένα εθνικό πάρκο.



Διάγραμμα Ακολουθίας (Sequence Diagram)

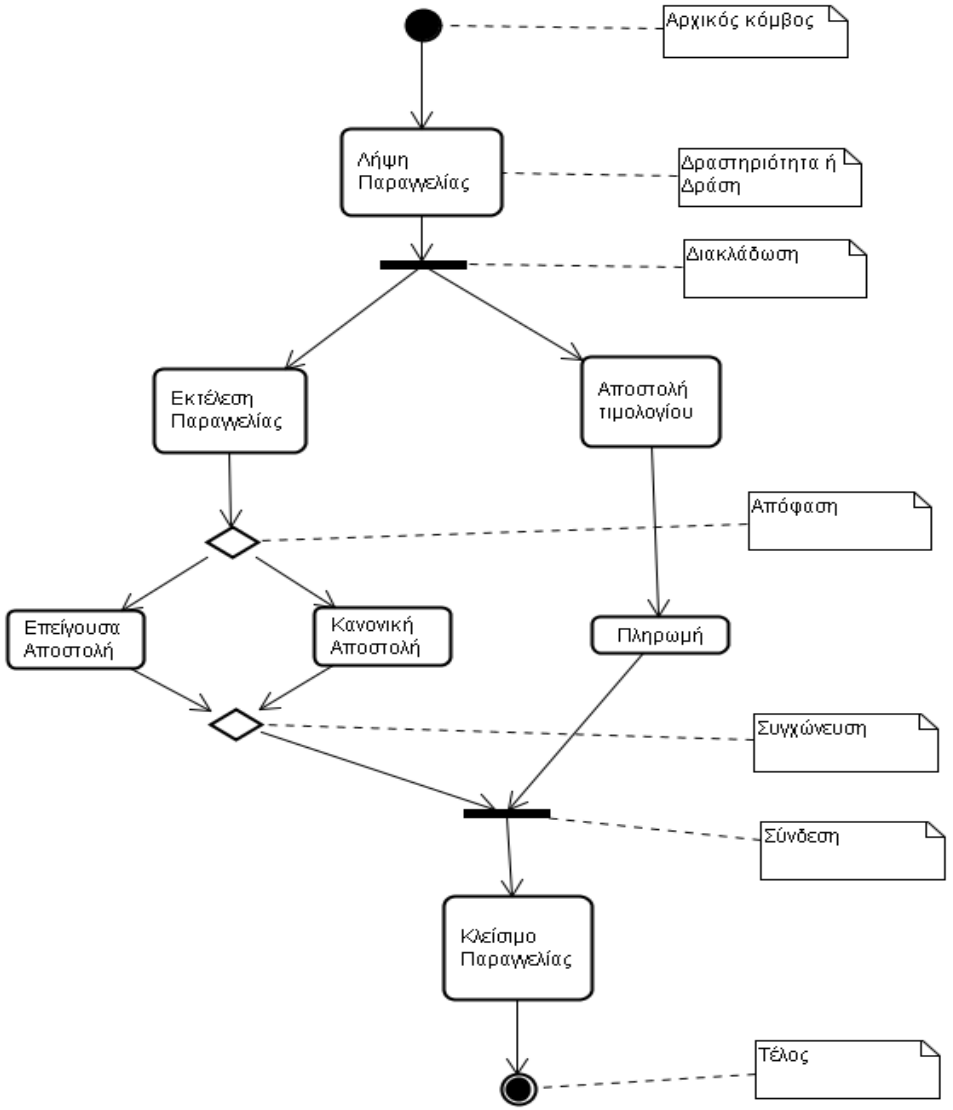
Σχεδιασμός εφαρμογής για το παιχνίδι Bowling για την Περίπτωση Χρήσης «Εμφάνισε σκορ».



Διάγραμμα Δραστηριοτήτων (Activity Diagram)

Σχεδιασμός εφαρμογής Διαχείρισης Παραγγελιών.

Χρησιμοποιούνται για την περιγραφή επιχειρησιακών διαδικασιών – ροών εργασιών που απαιτούν αποσαφήνιση.



Ταξινόμηση Διαδικασιών Ανάπτυξης Πληροφοριακών Συστημάτων

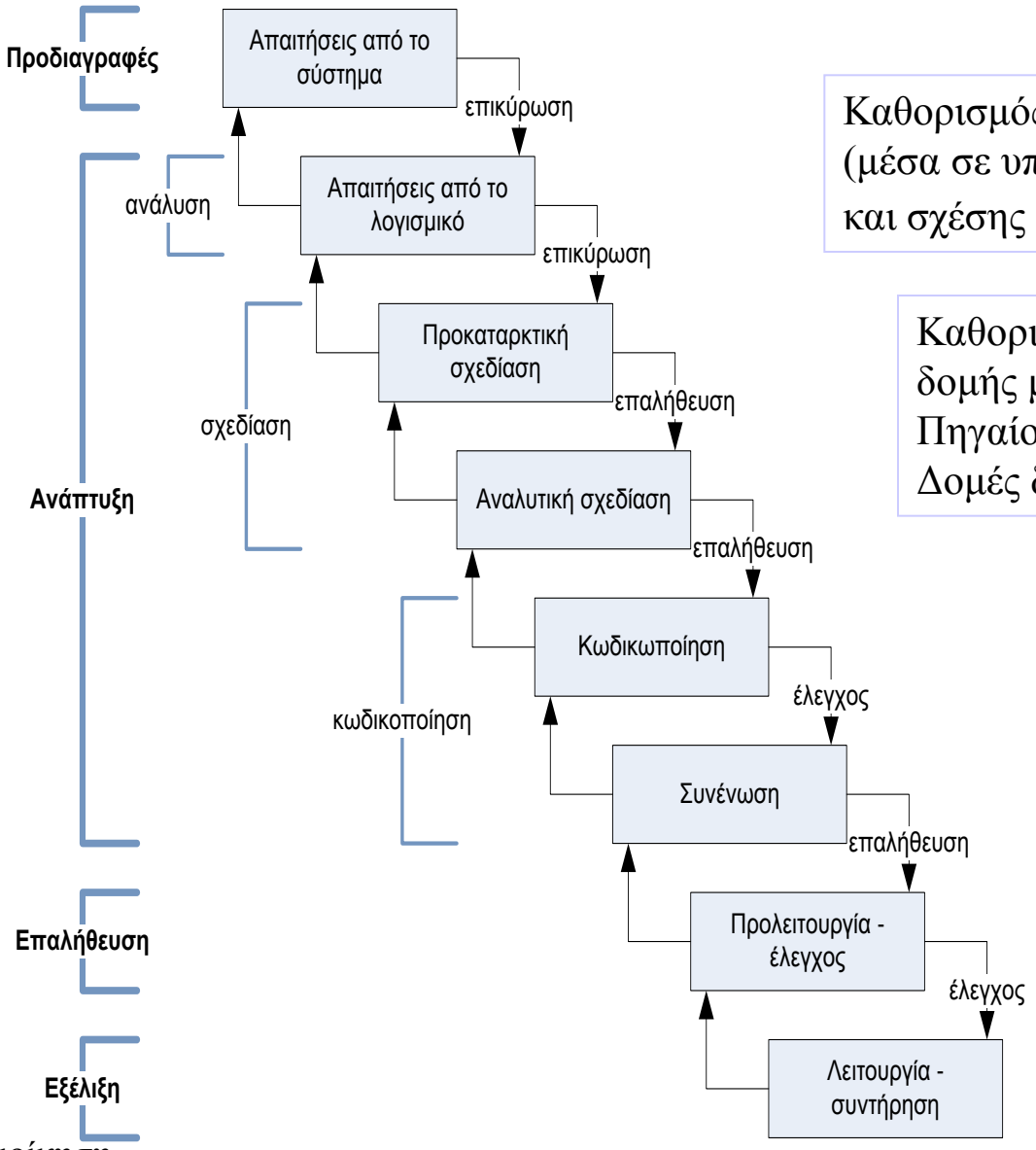
Προδιαγραφή, δηλαδή καθορισμός των εργασιών που θα επιτελεί το πληροφοριακό σύστημα / λογισμικό, καθώς και των περιορισμών και των παραδοχών που ισχύουν.

Ανάπτυξη, δηλαδή κατασκευή του λογισμικού. Εδώ, σε όλα τα μοντέλα κύκλου ζωής μπορούμε να διακρίνουμε τρεις επιμέρους φάσεις: την ανάλυση, τη σχεδίαση και τη συγγραφή του πηγαίου κώδικα (source code), την οποία στη συνέχεια θα ονομάζουμε και κωδικοποίηση.

Επαλήθευση, δηλαδή επιβεβαίωση της ικανοποίησης των προδιαγραφών και της μη ύπαρξης σφαλμάτων.

Εξέλιξη, δηλαδή επαύξηση των λειτουργικών χαρακτηριστικών του λογισμικού ή τροποποίηση υπαρχουσών, προκειμένου να ικανοποιούνται οι μεταβαλλόμενες ανάγκες.

Το μοντέλο του Καταρράκτη (Waterfall)



Καθορισμός μονάδων (μέσα σε υποσυστήματα) και σχέσης μεταξύ τους

Καθορισμός εσωτερικής δομής μονάδων – Μονάδες Πηγαίου Κώδικα (αλγόριθμοι, Δομές δεδομένων κτλ)

Γενικά για το μοντέλο του Καταρράκτη

Τα στάδια αυτού του μοντέλου παριστάνονται με τη μορφή μιας γραμμικής ακολουθίας, σαν καταρράκτης που οδηγεί από το ένα στο άλλο. Για να ξεκινήσει κάθε στάδιο της ανάπτυξης θα πρέπει να έχει ολοκληρωθεί το προηγούμενό του.

Σύμφωνα με αυτό το μοντέλο, με κάθε δραστηριότητα της διεργασίας σχετίζονται **ορόσημα (milestones)** και πρότυπα παραδοτέων προϊόντων, έτσι ώστε οι υπεύθυνοι των έργων να μπορούν να χρησιμοποιούν το μοντέλο για να κάνουν εκτιμήσεις για το χρόνο ολοκλήρωσης του έργου τους ανά πάσα χρονική στιγμή.

Η απλότητά του διευκολύνει και την ενημέρωση των πελατών, οι οποίοι δεν είναι εξοικειωμένοι με την ανάπτυξη συστημάτων, επειδή ορίζει ξεκάθαρα τα ενδιάμεσα προϊόντα που είναι απαραίτητα προκειμένου να ξεκινήσει το επόμενο στάδιο της ανάπτυξης.

Επεξήγηση των φάσεων του μοντέλου

Ανάλυση και προσδιορισμός των απαιτήσεων: Οι υπηρεσίες, στόχοι και περιορισμοί του συστήματος καθορίζονται μετά από συζήτηση με τους ανθρώπους που θα χρησιμοποιήσουν το σύστημα. Μετά ορίζονται με τρόπο κατανοητό και από τις δύο πλευρές

Σχεδιασμός συστήματος και λογισμικού: Καθορίζεται μια γενική αρχιτεκτονική του συστήματος χωρίζοντας τις απαιτήσεις του συστήματος σε hardware ή software απαιτήσεις. Ο software σχεδιασμός έχει να κάνει με την παρουσίαση του λειτουργιών λογισμικού που απαιτούνται και που πρέπει να μεταφραστούν σε εκτελέσιμα προγράμματα

Υλοποίηση και έλεγχος ενοτήτων (units): Κατά τη διάρκεια αυτού του σταδίου, το λογισμικό υλοποιείται ως ένα σύνολο από προγράμματα και ενότητες προγραμμάτων. Κάθε ενότητα πρέπει να ελεγχθεί ώστε να διαπιστωθεί ότι πληροί τις προδιαγραφές.

Επεξήγηση των φάσεων του μοντέλου

Ολοκλήρωση και έλεγχος του συστήματος: Σε αυτό το στάδιο το σύστημα χτίζεται από τις επιμέρους ενότητες και ελέγχεται ως ολοκληρωμένο πλέον σύστημα. Μετά τον έλεγχο μπορεί να παραδοθεί στο χρήστη

Λειτουργία και συντήρηση: Το σύστημα εγκαθίσταται και χρησιμοποιείται. Συντήρηση σημαίνει διόρθωση των λαθών που δεν είχαν ανακαλυφθεί σε προηγούμενα στάδια του σχεδιασμού, βελτίωση των προγραμμάτων και βελτίωση των υπηρεσιών του συστήματος όσο παρουσιάζονται νέες απαιτήσεις

Ερώτημα - Άσκηση

Μια σημαντική παράμετρος που καταδεικνύει τη σημασία των μοντέλων κύκλου ζωής είναι το κόστος. Πως θεωρείτε ότι αυτό μπορεί να εξηγηθεί;

Κόστος

Το **κόστος αναθεώρησης αποφάσεων ή/και διόρθωσης σφαλμάτων** είναι τόσο μεγαλύτερο, όσο μεγαλύτερη είναι και η απαιτούμενη οπισθοδρόμηση της διαδικασίας που αυτή συνεπάγεται.

Το κόστος αυτό δεν αφορά μόνο οικονομικούς πόρους που αποδίδονται στο έργο, αλλά και **χρόνο καθυστέρησης**, που δεν είναι πάντα διαθέσιμος σε πραγματικές συνθήκες.

Επίσης, είναι συχνό φαινόμενο οι **παρενέργειες** στο υπόλοιπο σύστημα λογισμικού (**side effects**), οι οποίες μπορούν να μεταβάλλουν προς το χειρότερο τα ποιοτικά του χαρακτηριστικά και δεν είναι εύκολο να εντοπιστούν από την αρχή.

Ερώτημα - Άσκηση

Αναφέρατε 5 περιπτώσεις ρίσκου για την υλοποίηση ενός project ανάπτυξης ενός πληροφοριακού συστήματος. Τι θα κάνατε;

Παραδείγματα ρίσκου

Μη συγκεκριμένες απαιτήσεις συστήματος

Λάθος κατανόηση μιας απαίτησης

Αλλαγή ορισμένων από τις απαιτήσεις

Αποχώρηση (ασθένειες) ατόμων από την ομάδα ανάπτυξης

Το λογισμικό πληροί τις προδιαγραφές αλλά έχει πολύ μεγάλο χρόνο εκτέλεσης

Εμφάνιση νέων εργαλείων ανάπτυξης

Αλλαγή του υλικού

Μείωση προϋπολογισμού

Βλάβες στο υλικό της ομάδας ανάπτυξης

Ερώτημα - Άσκηση

Αναφέρετε τουλάχιστον δύο χαρακτηριστικά του μοντέλου του καταρράκτη τα οποία μπορούν να θεωρηθούν μειονεκτήματα, με βάση τα όσα είπαμε προηγούμενα. Όπως είπαμε από την αρχή, το μοντέλο του καταρράκτη αντιμετωπίζει ολόκληρη την εφαρμογή λογισμικού σε κάθε βήμα.

Απάντηση

1. Δεν είναι δυνατό να υπάρχει μια πρώτη εικόνα του συστήματος λογισμικού που κατασκευάζεται, παρά μόνο σε προχωρημένη φάση της ανάπτυξης.

2. Όσο μεγαλώνει η έκταση του συστήματος που κατασκευάζεται, τόσο δυσκολότερη γίνεται η μετάβαση από τη μια φάση στην επόμενη και η αποφυγή σφαλμάτων, που δεν είναι δυνατό να εντοπιστούν παρά σε πολύ προχωρημένες φάσεις της ανάπτυξης.

3.Όσο αργότερα στην ανάπτυξη εντοπίζεται ένα σφάλμα, τόσο μεγαλύτερες είναι οι επιπτώσεις που η διόρθωσή του μπορεί να έχει σε όρους κόστους οπισθοδρομήσεων και επανάληψης τουλάχιστον μέρους της διαδικασίας, παρενεργειών και δημιουργίας και νέων σφαλμάτων, καθυστερήσεων κτλ.

Ερώτημα - Άσκηση

Θα διαλέγατε το μοντέλο του Καταρράκτη για την ανάπτυξη των παρακάτω εφαρμογών; Γιατί;

1. Ένα διαδραστικό σύστημα που δίνει πληροφορίες για δρομολόγια τρένων
2. Λογιστικό σύστημα πανεπιστημίου που αντικαθιστά υπάρχον

Απάντηση

1. Όχι

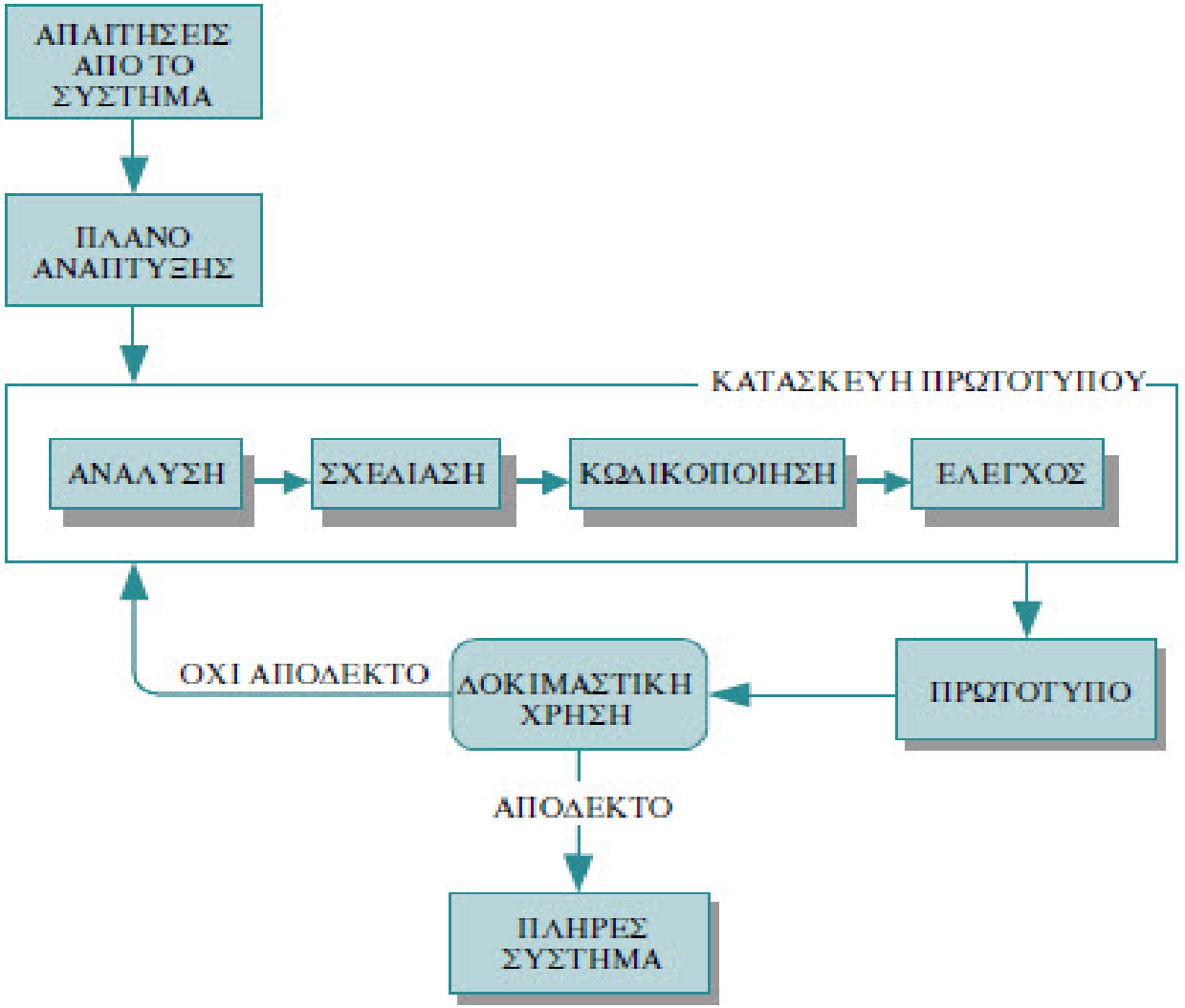
Σύνθετο interface, μη ξεκαθαρισμένες απαιτήσεις, μεγάλο σύστημα, μη εμπειρία σε τέτοια ΠΣ , σχεδιαστικά ζητήματα, επικύρωση από χρήστες ….

2. Ναι

Οι απαιτήσεις είναι πολύ συγκεκριμένες

Υπάρχει εμπειρία σε αυτά τα συστήματα

Το μοντέλο πρωτοτυποποίησης



Το μοντέλο πρωτοτυποποίησης

Ανάπτυξη λογισμικού σε στάδια

Σε κάθε στάδιο φτιάχνεται ένα πρωτότυπο (Prototype)

Οι διαδικασίες ανάπτυξης επαναλαμβάνονται για κάθε πρωτότυπο (επαναληπτικό μοντέλο)

Το πρωτότυπο περιλαμβάνει τις βασικές μόνο λειτουργίες που προορίζεται να εκτελεί το ΠΣ.

Κάθε πρωτότυπο δοκιμάζεται από τον πελάτη-χρήστη και βελτιώνεται με νέα έκδοση πρωτοτύπου (νέο πρωτότυπο)

Το μοντέλο πρωτοτυποποίησης

Ένα σημαντικό πλεονέκτημα του μοντέλου αυτού είναι η δυνατότητα απόκτησης άποψης για την εφαρμογή λογισμικού νωρίτερα απ' ό,τι στο μοντέλο του καταρράκτη.

Αυτό μπορεί να γλιτώσει την ανάπτυξη από καθυστερήσεις (και συνεπαγόμενα κόστη) ή ακόμη και από ολική αποτυχία, τα οποία θα επέρχονταν, αν ο κατασκευαστής αναγκαζόταν να οπισθοδρομήσει την ανάπτυξη, ενώ αυτή είχε προχωρήσει πολύ.

Παράλληλα, ιδιαίτερη σημασία αποκτά η διοίκηση του έργου, η οποία πρέπει να εξασφαλίζει την υλοποιησιμότητα του πρωτοτύπου και την εύκολη τροποποίησή του.

Κάθε κατασκευή πρωτοτύπου μπορεί να θεωρηθεί ως ένα μικρό έργο λογισμικού το οποίο κατασκευάζεται με διαδικασίες που μπορούν να ακολουθούν άλλα μοντέλα κύκλου ζωής, όπως αυτό του καταρράκτη.

Το μοντέλο πρωτοτυποποίησης

Με βάση τις παραπάνω παρατηρήσεις, το μοντέλο πρωτοτυποποίησης χρησιμοποιείται στην ανάπτυξη εφαρμογών λογισμικού για τις απαιτήσεις από τις οποίες δεν υπάρχει βεβαιότητα στην αρχή της ανάπτυξης **(υψηλός κίνδυνος)**, οπότε δεν μπορούν να συμφωνηθούν και να παγιοποιηθούν.

Τέτοιες είναι εφαρμογές που κατασκευάζονται για πρώτη φορά ή που είναι στενά εξαρτημένες από τον πελάτη, χωρίς να υπάρχει αποδεκτό προηγούμενο παράδειγμα.

Ωστόσο, το μέγεθος των εφαρμογών αυτών δεν μπορεί να είναι ιδιαίτερα μεγάλο, διότι ο χρόνος ανάπτυξης κάθε πρωτοτύπου μεγαλώνει και η απαιτούμενη ευελιξία μειώνεται.

Ερώτημα – Άσκηση

Αναφέρατε (επιγραμματικά) 2 βασικά πλεονεκτήματα του μοντέλου πρωτοτυποποίησης.

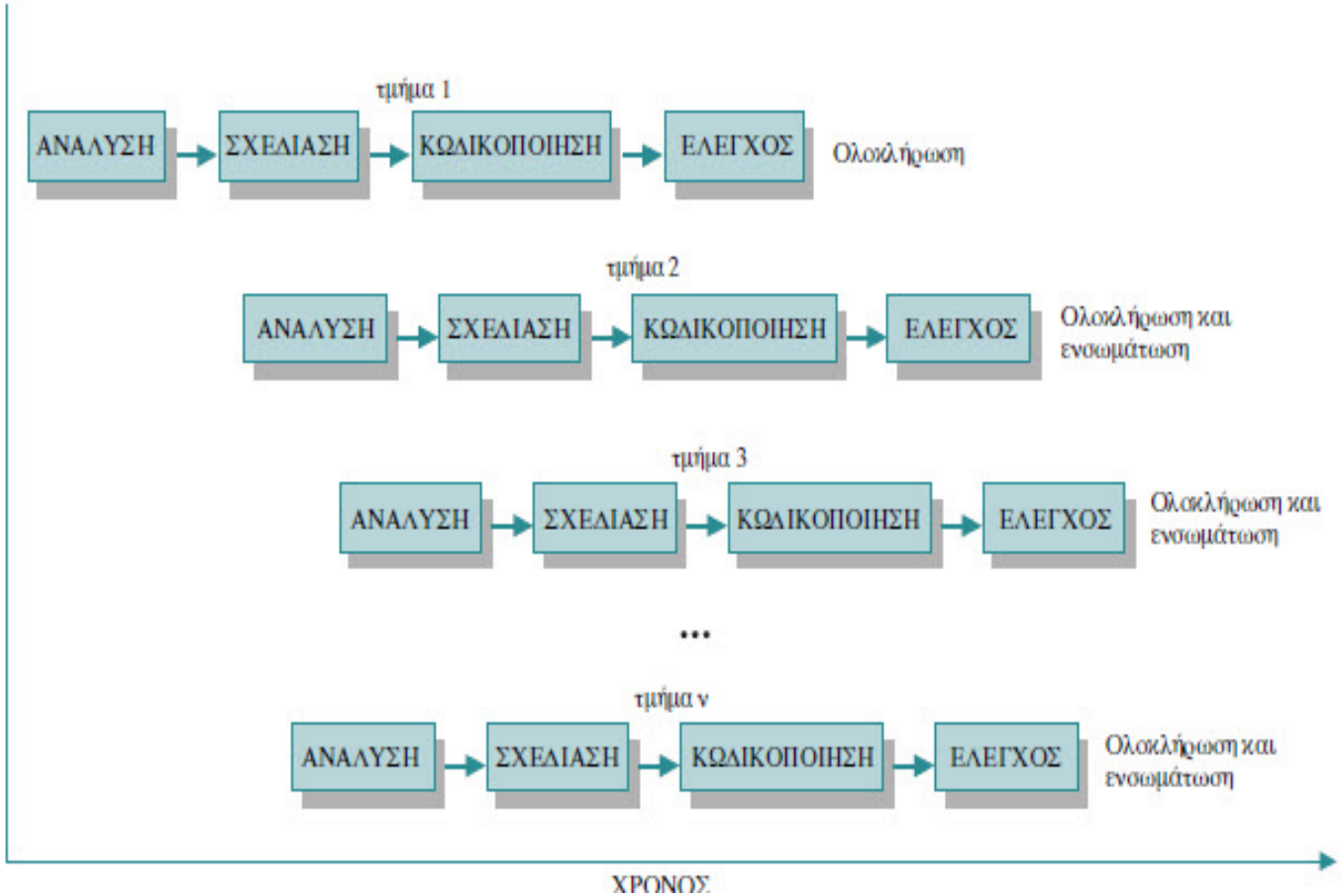
Απάντηση

Πλεονέκτημα είναι η γρήγορη ανίχνευση αναγκών – προβλημάτων πριν την ανάπτυξη μεγάλου μέρους του Λογισμικού.

Αποκτάται άποψη για την τελική μορφή της εφαρμογής σε σύντομο χρονικό διάστημα.

Μειώνονται οι καθυστερήσεις και τα αντίστοιχα κόστη.

Το μοντέλο λειτουργικής επαύξησης



Το μοντέλο λειτουργικής επαύξησης

Το μοντέλο της λειτουργικής επαύξησης (incremental model) συνδυάζει την ακολουθιακή ανάπτυξη του μοντέλου του καταρράκτη με την τμηματική ανάπτυξη του μοντέλου της πρωτοτυποποίησης.

Κεντρική ιδέα είναι η κατάτμηση του υπό κατασκευή λογισμικού σε τμήματα που αναπτύσσονται ανεξάρτητα, ακολουθώντας το καθένα ακολουθιακή ανάπτυξη σύμφωνα με το μοντέλο του καταρράκτη, όπως φαίνεται στο σχήμα.

Κατά την αρχική φάση ανάλυσης και σχεδίασης αποφασίζονται τα τμήματα στα οποία θα κατατμηθεί η εφαρμογή, η ανάπτυξη των οποίων γίνεται στη συνέχεια ανεξάρτητα και παράλληλα.

Όταν ολοκληρώνεται η ανάπτυξη κάθε τμήματος, αυτό ενσωματώνεται στο σύνολο της εφαρμογής, διαδικασία η οποία δικαιολογεί και τον τίτλο «λειτουργική επαύξηση».

Το μοντέλο λειτουργικής επαύξησης

Πλεονεκτήματα της ιδέας είναι η δυνατότητα παράλληλης ανάπτυξης, η οποία τελικά καταλαμβάνει μικρότερο χρόνο, καθώς και ο διαδοχικός εμπλουτισμός των λειτουργικών χαρακτηριστικών του λογισμικού.

Τα βασικά μειονεκτήματα του μοντέλου είναι τα ακόλουθα:

- Η αρχική κατάτμηση και γενική σχεδίαση του συστήματος αποκτά ιδιαίτερη βαρύτητα. Σφάλματα σε αυτή μπορούν να έχουν σημαντικές επιπτώσεις στο λογισμικό που θα κατασκευαστεί στη συνέχεια.
- Σε περίπτωση μεταβολής των λειτουργικών απαιτήσεων κατά την ανάπτυξη του ημιτελούς συστήματος, μπορεί η αρχιτεκτονική αυτού να μεταβληθεί σε βαθμό που να κλονιστεί η ανάπτυξη των υπόλοιπων τμημάτων αυτού.

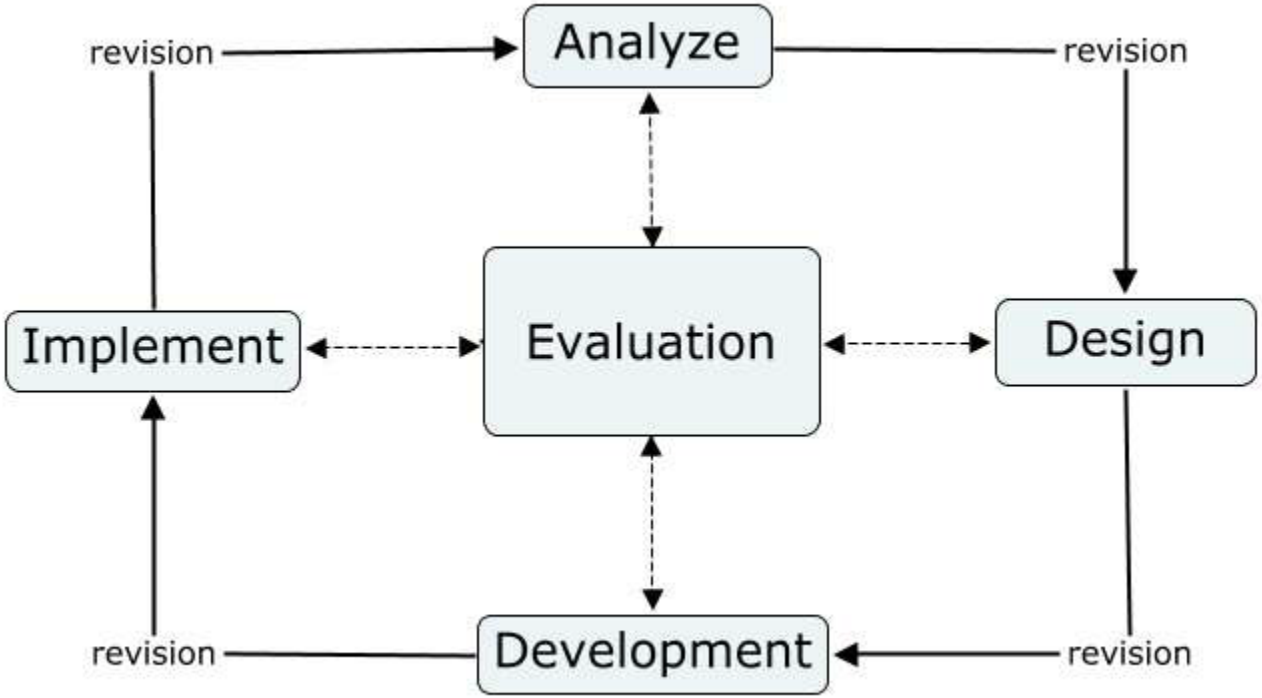
ADDIE process ή model

Ο διδακτικός σχεδιασμός συστημάτων (Instuctional Systems Design, ISD) ή μια διδακτική παρέμβαση στην εκπαίδευση και την κατάρτιση χρησιμοποιούν συνήθως μια γενικευμένη σχεδιαστική προσέγγιση, γνωστή ως **ADDIE process ή model** για να καθορίσουν τα χαρακτηριστικά για μια σειρά από ενέργειες που υποστηρίζουν τη διαδικασία μάθησης.

Αποτελείται από πέντε διαδοχικά στάδια-φάσεις, που είναι τα εξής: **Ανάλυση – Σχεδιασμός – Ανάπτυξη – Εφαρμογή – Αξιολόγηση**

Πολλά από τα σύγχρονα μοντέλα διδακτικού σχεδιασμού αποτελούν παραλλαγές αυτού του μοντέλου. Η μεθοδολογία αυτή μπορεί να εφαρμοστεί στο σχεδιασμό της εκπαιδευτικής διαδικασίας ως εξής:

ADDIE process ή model



Ανάλυση – Analysis

Στη φάση αυτή:

γίνεται λεπτομερής καθορισμός του προβλήματος,

καθορίζονται οι διδακτικοί στόχοι και

αναγνωρίζεται το μαθησιακό περιβάλλον και οι προϋπάρχουσες γνώσεις του μαθητή

Μερικές ερωτήσεις-οδηγός:

Ποιο είναι το αποτέλεσμα της διαδικασίας;

Ποιοι είναι οι μαθητές και ποια είναι τα χαρακτηριστικά τους;

Ποιοι μαθησιακοί περιορισμοί υφίστανται;

Τι χρονικοί περιορισμοί υπάρχουν για την ολοκλήρωση του έργου;

Σχεδιασμός – Design

Η φάση του σχεδιασμού έχει να κάνει με

- μαθησιακούς στόχους, εργαλεία αξιολόγησης, ασκήσεις, περιεχόμενο, σχεδιασμό και επιλογή πολυμεσικών υλικών
- καταγραφή-τεκμηρίωση της διδακτικής στρατηγικής, των οπτικοακουστικών μέσων, και της τεχνολογίας
- εφαρμογή στρατηγικών σύμφωνα με τα επιθυμητά αποτελέσματα (γνωσιακά, συναισθηματικά, ψυχοκινητικά)

και

- Εκτίμηση χρόνου και κόστους σε ανθρώπινους και υλικούς πόρους
- Σχεδιασμός της διεπιφάνειας χρήσης (user interface)
- Δημιουργία πρωτοτύπου
- Φυσικός σχεδιασμός (visual design ή graphic design)

Ανάπτυξη – Development

Στη φάση της ανάπτυξης δημιουργούνται και συνενώνονται τα στοιχεία της φάσης σχεδιασμού. Στη φάση αυτή χρησιμοποιούνται εργαλεία όπως σενάρια χρήσης, πρωτότυπα οθονών κ.α.

Αναπτύσσονται και ολοκληρώνονται οι σχετικές τεχνολογίες

Γίνεται αποσφαλμάτωση υπορουτινών

Πιθανές τροποποιήσεις στο σχεδιασμό ανάλογα με την αλληλεπίδραση παιδαγωγών-μηχανικών ανάπτυξης και τη σχετική ανάδραση που προκύπτει

Εφαρμογή – Implementation

Στη διαδικασία της εφαρμογής αναπτύσσονται διαδικασίες εκπαίδευσης των συμμετεχόντων (εκπαιδευτές – μαθητές)

Για τους **εκπαιδευτικούς** δίνεται έμφαση

στο περιεχόμενο του μαθήματος

στα αποτελέσματα

στις χρησιμοποιούμενες μεθόδους διδακτικής παρέμβασης και

στις διαδικασίες ελέγχου

Για τους **μαθητές** δίνεται έμφαση

στην εκπαίδευση στα νέα εργαλεία

στην εξοικείωσή τους για την είσοδο στο σύστημα και

στην ανάπτυξη σχετικών υποστηρικτικών υλικών (help files)

Ελέγχονται επίσης όλα τα σχετικά υλικά (βιβλία, εξοπλισμός κλπ)

Αξιολόγηση – Evaluation

Η αξιολόγηση αποτελείται από 2 φάσεις:

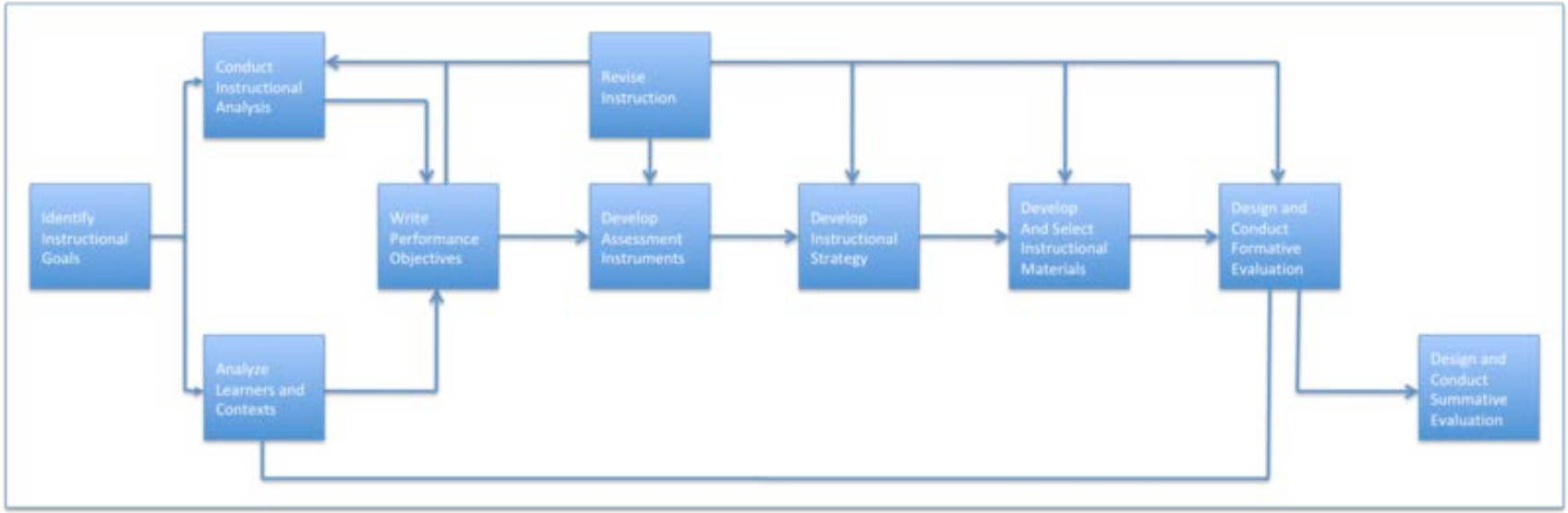
Διαμορφωτική αξιολόγηση (Formative evaluation) η οποία λαμβάνει χώρα σε κάθε στάδιο του μοντέλου

Αθροιστική ή τελική αξιολόγηση (Summative evaluation) η οποία αποτελεί στοχευμένη δράση μετά το τέλος του κύκλου ανάπτυξης

- Μαθησιακή αποτελεσματικότητα
- Αλληλεπίδραση μαθητή-περιβάλλοντος
- Τεχνική αρτιότητα

Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Το μοντέλο **ADDIE** αποτελεί ένα γενικό μοντέλο το οποίο μπορεί να εφαρμοστεί σε πλήθος περιστάσεων, αλλά έχει δεχτεί και ισχυρή κριτική ότι ουσιαστικά δεν αποτελεί μοντέλο αλλά μία πρακτική των σχεδιαστών ή ότι είναι πολύ γραμμικό και στατικό και δεν μπορεί να ανταποκριθεί στις εξελίξεις στην εκπαιδευτική τεχνολογία, όπως για παράδειγμα το μοντέλο **Rapid Prototyping**, το οποίο βασίζεται σε περισσότερο σύγχρονες προσεγγίσεις που σχετίζονται με την ανάπτυξη εκπαιδευτικού υλικού / λογισμικού.



Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Ο προσδιορισμός των εκπαιδευτικών σκοπών (Identify Instructional Goals)

Στο βήμα αυτό ο εκπαιδευτής/σχεδιαστής πρέπει να καθορίσει τις νέες πληροφορίες, δεξιότητες, γνώση που πρέπει να προσλάβουν οι εκπαιδευόμενοι μέσω της εκπαιδευτικής παρέμβασης αυτής και να τις εκφράσει ως γενικούς στόχους, οι οποίοι πρέπει να επιτευχθούν μέσω της παρέμβασης.

Η διεξαγωγή εκπαιδευτικής Ανάλυσης (Conduct Instructional Analysis)

Μετά την αναγνώριση των εκπαιδευτικών στόχων για την παρέμβαση, διεξάγεται η ανάλυσή τους, προκειμένου να διευκρινιστούν οι γνώσεις και οι δεξιότητες που θα περιληφθούν στην εκπαιδευτική παρέμβαση και που απαιτούνται για την ολοκληρωμένη επεξεργασία του κάθε στόχου. Στο βήμα αυτό, επίσης, είναι απαραίτητο να προσδιορίζονται οι προαπαιτούμενες γνώσεις, ικανότητες και δεξιότητες από την πλευρά των εκπαιδευομένων, προκειμένου να μπορούν να παρακολουθήσουν την εκπαιδευτική παρέμβαση αυτή με επιτυχία.

Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Ανάλυση Εκπαιδευομένων και Πλαισίων (Analyze Learners and Contexts)
Παράλληλα με την εκπαιδευτική ανάλυση είναι σημαντικό να διευκρινίζονται και τα ιδιαίτερα χαρακτηριστικά των εκπαιδευομένων, των γνώσεων, δεξιοτήτων και ικανοτήτων τους, να προσδιορίζονται οι παράμετροι του περιβάλλοντος της εκπαιδευτικής παρέμβασης και οι συνθήκες και τα χαρακτηριστικά του περιβάλλοντος στο οποίο θα χρησιμοποιηθούν οι γνώσεις αυτές (π.χ. σχολική εκπαίδευση, πανελλήνιες εξετάσεις, επιχειρησιακό περιβάλλον, τριτοβάθμια εκπαίδευση).

Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Καθορισμός στόχων επίδοσης (Write Performance Objectives)
Με βάση την ανάλυση του προηγούμενου σταδίου είναι απαραίτητο να καθοριστεί, όσο το δυνατόν πιο συγκεκριμένα, τι θα πρέπει να είναι ικανοί να κάνουν οι εκπαιδευόμενοι με την ολοκλήρωση των στόχων εκπαιδευτικής παρέμβασης, οι συνθήκες με τις οποίες μπορεί να επιδειχτεί η ικανότητα αυτή και τα κριτήρια της επιτυχημένης επίδοσης των εκπαιδευομένων. Κατά το στάδιο αυτό αναπτύσσονται, δηλαδή, συγκεκριμένα μαθησιακά αποτελέσματα τα οποία πρέπει να επιτύχουν οι εκπαιδευόμενοι.
Ανάπτυξη Εργαλείων Αξιολόγησης (Develop Assessment Instruments)
Σύμφωνα με τους στόχους επίδοσης που έχουν καθοριστεί κατά το προηγούμενο βήμα αναπτύσσονται στη συνέχεια και οι κατάλληλες μέθοδοι και τρόποι αξιολόγησης της επίδοσης των εκπαιδευομένων. Ιδιαίτερη έμφαση πρέπει να δοθεί στην ύπαρξη διαφορετικών μεθόδων αξιολόγησης, ώστε να ανταποκρίνονται στις διαφορετικές ανάγκες όλων των εκπαιδευομένων, αλλά και να αξιολογούν ποικιλοτρόπως την επίδοσή τους (πχ. ερωτήσεις πολλαπλών απαντήσεων, σύντομες αναφορές, εργασίες κλπ.).

Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Ανάπτυξη Εκπαιδευτικής Στρατηγικής (Develop Instructional Strategy)
Η έννοια της εκπαιδευτικής στρατηγικής αναφέρεται στα στοιχεία αυτά τα οποία θα χρησιμοποιηθούν, προκειμένου να επιτευχθεί ο εκπαιδευτικός στόχος. Η στρατηγική αυτή για την επίτευξη των μαθησιακών στόχων που έχουν τεθεί, αναπτύσσεται στο στάδιο αυτό.

Σχεδιασμός και Επιλογή Εκπαιδευτικών Υλικών (Develop and Select Instructional Materials)
Κατά το βήμα αυτό, λαμβάνοντας υπόψη τα συμπεράσματα από τα προηγούμενα βήματα της διαδικασίας, αναπτύσσεται το εκπαιδευτικό υλικό. Το υλικό αυτό μπορεί είτε να αναπτυχθεί εξ αρχής είτε να επιλεγεί από ήδη υπάρχον εκπαιδευτικό υλικό είτε να παραχθεί συνδυάζοντας ήδη έτοιμο υλικό με υλικό που αναπτύσσεται για τη συγκεκριμένη εκπαιδευτική παρέμβαση.

Το μοντέλο εκπαιδευτικού σχεδιασμού των Dick & Carey

Σχεδιασμός και Διενέργεια Διαμορφωτικής Αξιολόγησης (Formative Evaluation)

Σημαντικό είναι με την ολοκλήρωση των παραπάνω βημάτων να διενεργείται διαμορφωτική αξιολόγηση της εκπαιδευτικής παρέμβασης, προκειμένου να διαπιστωθούν τα κενά, τα σφάλματα και οι ελλείψεις ή να προσδιοριστούν τα σημεία αυτά που απαιτούν βελτίωση, προκειμένου να επιτυγχάνεται αποτελεσματικότερα ο στόχος της.

Αναθεώρηση Εκπαιδευτικής Παρέμβασης (Revise Instruction)

Αξιοποιώντας τα συμπεράσματα της διαμορφωτικής αξιολόγησης, αναδιαμορφώνονται τα στοιχεία τα οποία κρίνεται απαραίτητο, προκειμένου να βελτιωθεί η αποτελεσματικότητα της εκπαιδευτικής παρέμβασης. Η διαδικασία αυτή είναι μια επαναλαμβανόμενη διαδικασία, μέχρι να επιτευχθεί ένα ικανοποιητικό ή το επιθυμητό αποτέλεσμα.

Σχεδιασμός και Διενέργεια Τελικής Αξιολόγησης (Summative Evaluation)

Κανονικά δεν ανήκει στη διαδικασία σχεδιασμού. Η τελική αξιολόγηση έχει ως στόχο να συγκεντρώσει τα απαραίτητα δεδομένα για το βαθμό επιτυχίας μιας ολοκληρωμένης εκπαιδευτικής παρέμβασης.

Συζήτηση

Τα παραπάνω μοντέλα έχουν

- Ίδιες διαδικασίες ανάπτυξης λογισμικού
- Ακολουθιακή (σειριακή) εκτέλεση σε όλο ή σε μέρος του λογισμικού
- Προκαθορισμένες φάσεις και διαδικασίες από το μοντέλο
- Ένας κύκλος ανάπτυξης

Το Σπειροειδές μοντέλο

Φάσεις και διαδικασίες ανάπτυξης **μη** προκαθορισμένες…εξειδικεύονται στο χώρο εφαρμογής

Πολλοί κύκλοι ανάπτυξης…Κάθε κύκλος περιέχει νέα λειτουργικά χαρακτηριστικά

Μελέτη σκοπιμότητας και ανάλυση κινδύνων πριν από την έναρξη κάθε κύκλου

Το Σπειροειδές μοντέλο

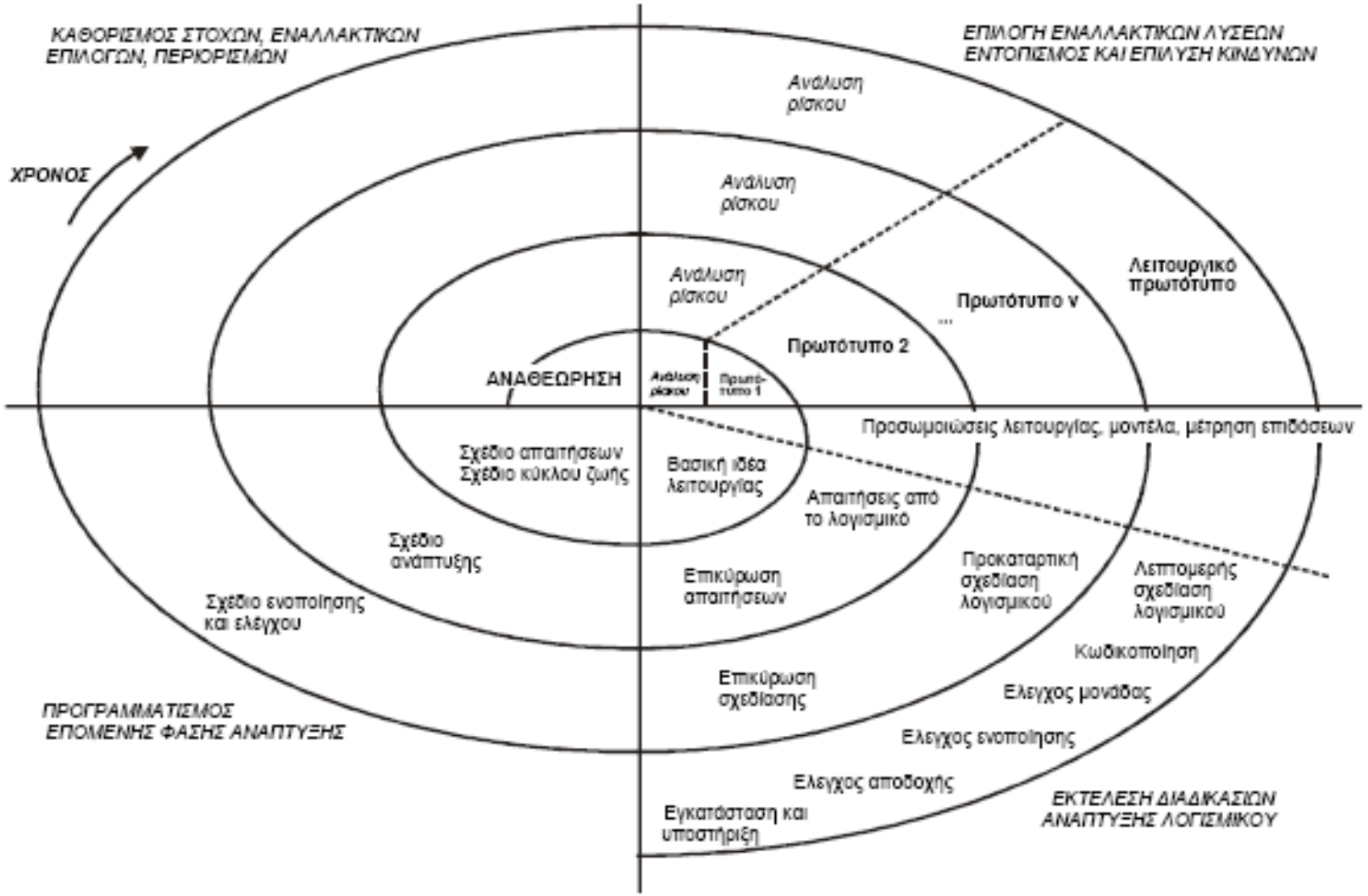
4 κατηγορίες εργασιών

- Προσδιορισμός στόχων
- Εντοπισμός – επίλυση κινδύνων
- Εκτέλεση διαδικασιών ανάπτυξης – Επαλήθευση
- Προγραμματισμός επόμενης φάσης

Σπειροειδές μοντέλο

- **Προσδιορισμός στόχων**
  - Καθορίζονται τα αντικείμενα εργασιών κάθε επανάληψης.
  - Καταγράφονται οι περιορισμοί επί του προϊόντος, αλλά και επί της διαδικασίας για την οποία κατασκευάζεται ένα αναλυτικό πλάνο διοίκησης.
  - Καταγράφονται οι κίνδυνοι που εμπεριέχει η διαδικασία και οι εναλλακτικές λύσεις, όπου υπάρχουν.
- **Επιλογή εναλλακτικών λύσεων, εντοπισμός και επίλυση κινδύνων**
  - Αναλύονται οι κίνδυνοι που έχουν καταγραφεί και αποτιμάται κάθε εναλλακτική λύση.
  - Λαμβάνονται αποφάσεις για τη συνέχιση ή όχι της ανάπτυξης, για το μοντέλο που θα ακολουθηθεί στη συγκεκριμένη επανάληψη, για την κατασκευή ή όχι πρωτοτύπου κ.ά.
- **Εκτέλεση** των βημάτων **της διαδικασίας ανάπτυξης λογισμικού** που έχει επιλεγεί για το τμήμα εκείνο του συστήματος που αφορά η τρέχουσα επανάληψη.
- Τέλος, μετά την επαλήθευση των αποτελεσμάτων-ενδιάμεσων προϊόντων λογισμικού γίνεται **προγραμματισμός της επόμενης φάσης ανάπτυξης**.

Το Σπειροειδές μοντέλο (Boehm 1988)



Το Σπειροειδές μοντέλο

Άρα σύμφωνα με αυτά που αναφέρθηκαν στο Σπειροειδές Μοντέλο:

Διαφορετικές διαδικασίες ανάπτυξης μπορούν να επιλεγούν για διαφορετικά τμήματα του λογισμικού

Ουσιαστικά είναι μια γενίκευση των μοντέλων **Λειτουργικής Επαύξησης + Πρωτοτυποποίησης**

Συνεχής καθορισμός λεπτομερειών υλοποίησης **(σε αντίθεση με τα προηγούμενα μοντέλα)**

Δύσκολη η εφαρμογή του μοντέλου

Ερώτημα - Άσκηση

Σχολιάστε την καταλληλότητα του σπειροειδούς μοντέλου στην υλοποίηση μικρών έργων ανάπτυξης λογισμικού. Το θεωρείτε κατάλληλο για τέτοιες υλοποιήσεις ή όχι;

Απάντηση

Το κόστος για την εκτέλεση των ενεργειών προγραμματισμού εργασιών, εκτίμησης του ρίσκου κτλ. κάθε άλλο παρά αμελητέο μπορεί να χαρακτηριστεί. Είναι αντιληπτό ότι το κόστος αυτό έχει μια ελάχιστη τιμή, πέραν της οποίας δεν μπορεί να μειωθεί, όσο και να μειώνεται το μέγεθος της εφαρμογής λογισμικού που αναπτύσσεται.

Στις περιπτώσεις μικρών εφαρμογών λογισμικού το κόστος αυτό είναι δυσανάλογα μεγάλο σε σχέση με το καθαρό κόστος των ενεργειών ανάπτυξης και, ως εκ τούτου, μπορεί η χρήση του σπειροειδούς μοντέλου να μην αποτελεί την καλύτερη επιλογή από οικονομική άποψη.

Αλλαγή φιλοσοφίας

Τέλος της δεκαετίας του 80 αρχή της δεκαετίας του 90.

Προσέγγιση βασισμένη στην «Αντικειμενοστρεφή» τεχνολογία.

Οι έννοιες «Ανάλυση, Σχεδίαση, Κωδικοποίηση» επικαλύπτονται

Παραγωγή επαναχρησιμοποιήσιμων μονάδων

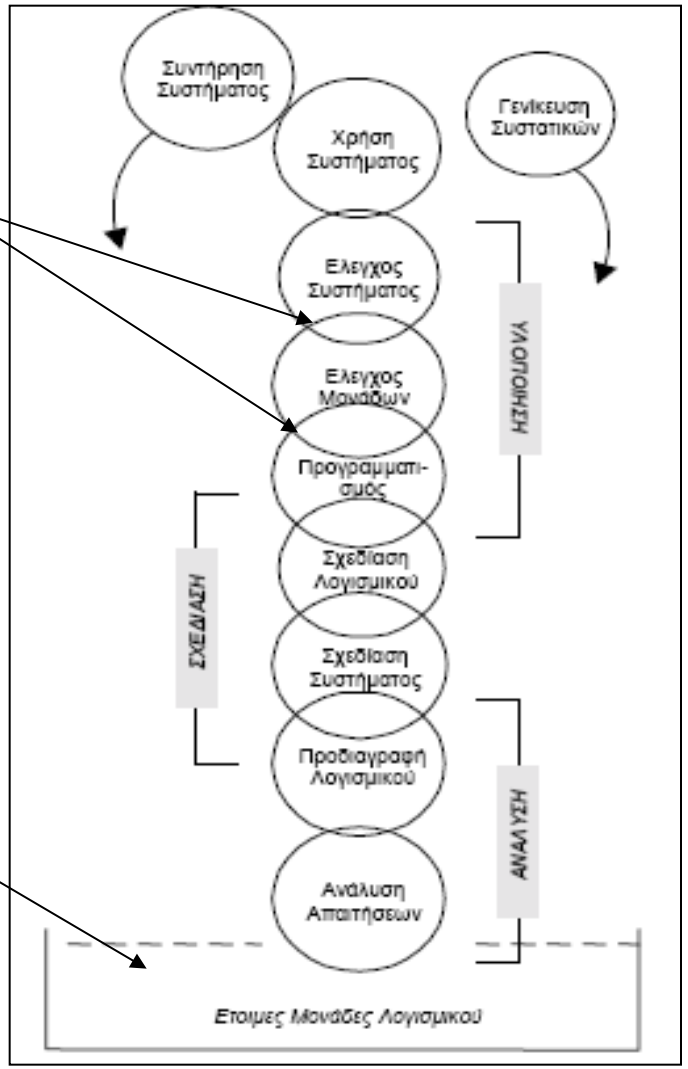
Χρήση μονάδων από μελλοντικά συστήματα

Μοντέλο του Πίδακα (fountain model)

Επικαλύψεις φάσεων

Τονίζει τα επιθυμητά χαρακτηριστικά της μεθοδολογίας κατασκευής λογισμικού σύμφωνα με την Αντικειμενοστραφή λογική

Δεξαμενή Συστατικών



Μοντέλο του Πίδακα (fountain model)

Κατά την ανάπτυξη παρατηρούνται επικαλύψεις των φάσεων «ανάλυση, σχεδίαση, κωδικοποίηση», οι οποίες φαίνονται με την επικάλυψη των κύκλων στο σχήμα.

Κατά το τέλος της ανάπτυξης, ορισμένα από τα συστατικά λογισμικού που έχουν παραχθεί ενσωματώνονται σε μια «δεξαμενή» συστατικών και διατίθενται για να χρησιμοποιηθούν στην ανάπτυξη και νέων συστημάτων. Η ιδέα του μοντέλου κύκλου ζωής του πίδακα τονίζει περισσότερο τα επιθυμητά χαρακτηριστικά της μεθοδολογίας κατασκευής του λογισμικού σύμφωνα με την αντικειμενοστρεφή λογική

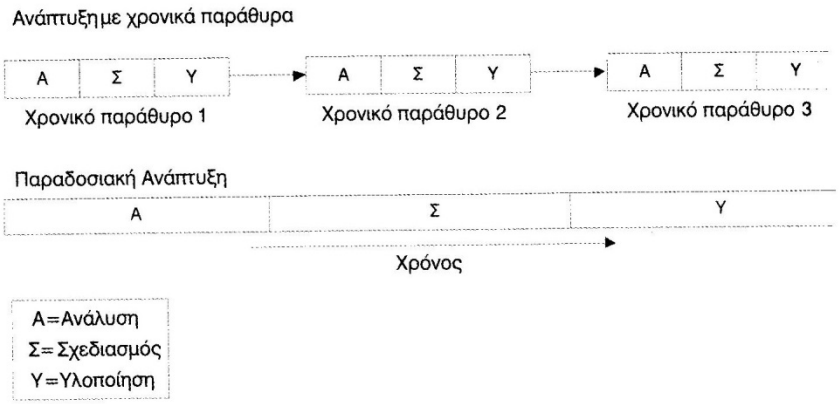
Ταχεία Ανάπτυξη Εφαρμογών (RAD)

Η Ταχεία Ανάπτυξη Εφαρμογών ή Rapid Application Development (RAD) ήρθε ως συνέχεια του Σπειροειδούς Μοντέλου και υιοθετεί τα παρακάτω 7 χαρακτηριστικά:

- Αυξητική ανάπτυξη

Δεν μπορούν να καθοριστούν όλες οι απαιτήσεις που εξελίσσονται και αλλάζουν

- Εφαρμογή χρονικών παραθύρων



- Αρχή Pareto (Κανόνας 80/20)

Το 80% της λειτουργικότητας του συστήματος μπορεί να παραδοθεί με το 20% της προσπάθειας που απαιτείται για όλο το σύστημα.

Ταχεία Ανάπτυξη Εφαρμογών (RAD)

- Κανόνες MoSCoW

The Must have, the Should have, the Could have, the Won't have

- Ημερίδες κοινής ανάπτυξης της εφαρμογής

Κοινή ανάπτυξη – Joint Application Development (με χρήστες)

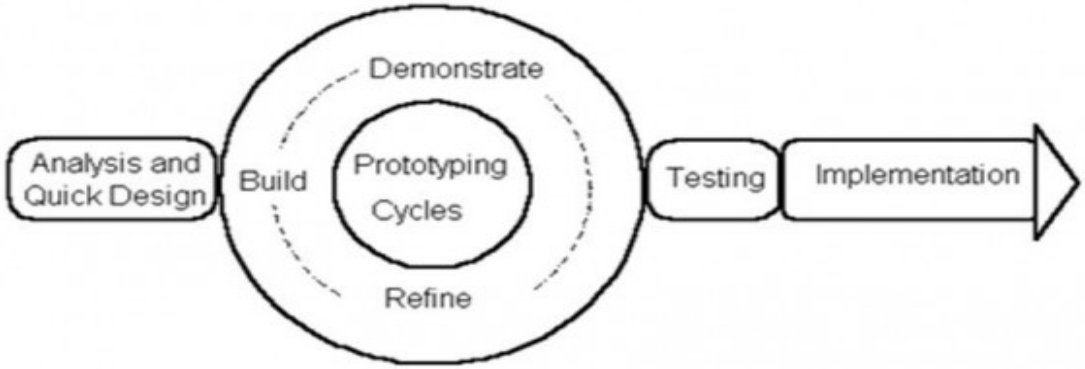
- Χρήση πρωτοτύπων

- Χρήση εργαλείων λογισμικού

Επαναχρησιμοποίηση κώδικα, χρήση και τροποποίηση υπαρχόντων εμπορικών πακέτων κλπ

Rapid prototyping είναι ένα από τα σύγχρονα μοντέλα που ανήκουν στην ομάδα μοντέλων RAD και χρησιμοποιείται στον διδακτικό σχεδιασμό ως προφανής βελτίωση του ADDIE model.

Το μοντέλο Ταχείας Προτυποποίησης (rapid prototyping)



Πλεονεκτήματα:

α) έχουν μικρούς χρόνους υλοποίησης,

β) απαιτούν μικρές ομάδες ανάπτυξης,

γ) δημιουργούνται απευθείας από τους ειδικούς του γνωστικού αντικειμένου με χρήση authoring tools + templates,

δ) στοχεύουν κατά κύριο λόγο στην αντιμετώπιση άμεσων εκπαιδευτικών ή επιμορφωτικών αναγκών και, τέλος,

ε) το εκπαιδευτικό υλικό έχει μικρό κύκλο ζωής και αναπτύσσεται εσωτερικά από τον εκπαιδευτικό φορέα ή οργανισμό.

Το μοντέλο Ταχείας Προτυποποίησης (rapid prototyping)

Μειονεκτήματα:

α) το φαινόμενο «***quick and dirty solution***»: μια πολύ γρήγορη υλοποίηση ενός αρχικού πρωτοτύπου, αλλά χαμηλής ποιότητας, μπορεί να δράσει αρνητικά στην όλη διαδικασία και να θέσει σε αμφισβήτηση την αποτελεσματικότητα του εγχειρήματος και του τελικού προϊόντος και

β) το φαινόμενο «***non-fast prototyping case***», κατά το οποίο η φάση της προτυποποίησης καθυστερεί υπερβολικά και δεν συνεισφέρει τίποτε στη διαδικασία του εκπαιδευτικού σχεδιασμού, γιατί χρονικά δεν επιτρέπει την ανατροφοδότηση και την αναπροσαρμογή του μαθήματος.

Σύγχρονα μοντέλα ανάπτυξης

Μεταγενέστερα μοντέλα κύκλου ζωής λογισμικού προσπαθούν να δώσουν μια γενική κατεύθυνση εφαρμογής των υπαρχουσών ιδεών, αφήνοντας σημαντικούς βαθμούς ελευθερίας στον κατασκευαστή που τα ακολουθεί.

Μια περιγραφή ενός σύγχρονου μοντέλου κύκλου ζωής λογισμικού περιέχει μόνο γενικές κατευθύνσεις, οι οποίες εξειδικεύονται στο εκάστοτε περιβάλλον ανάπτυξης, πρόβλημα κτλ.

Άρα:

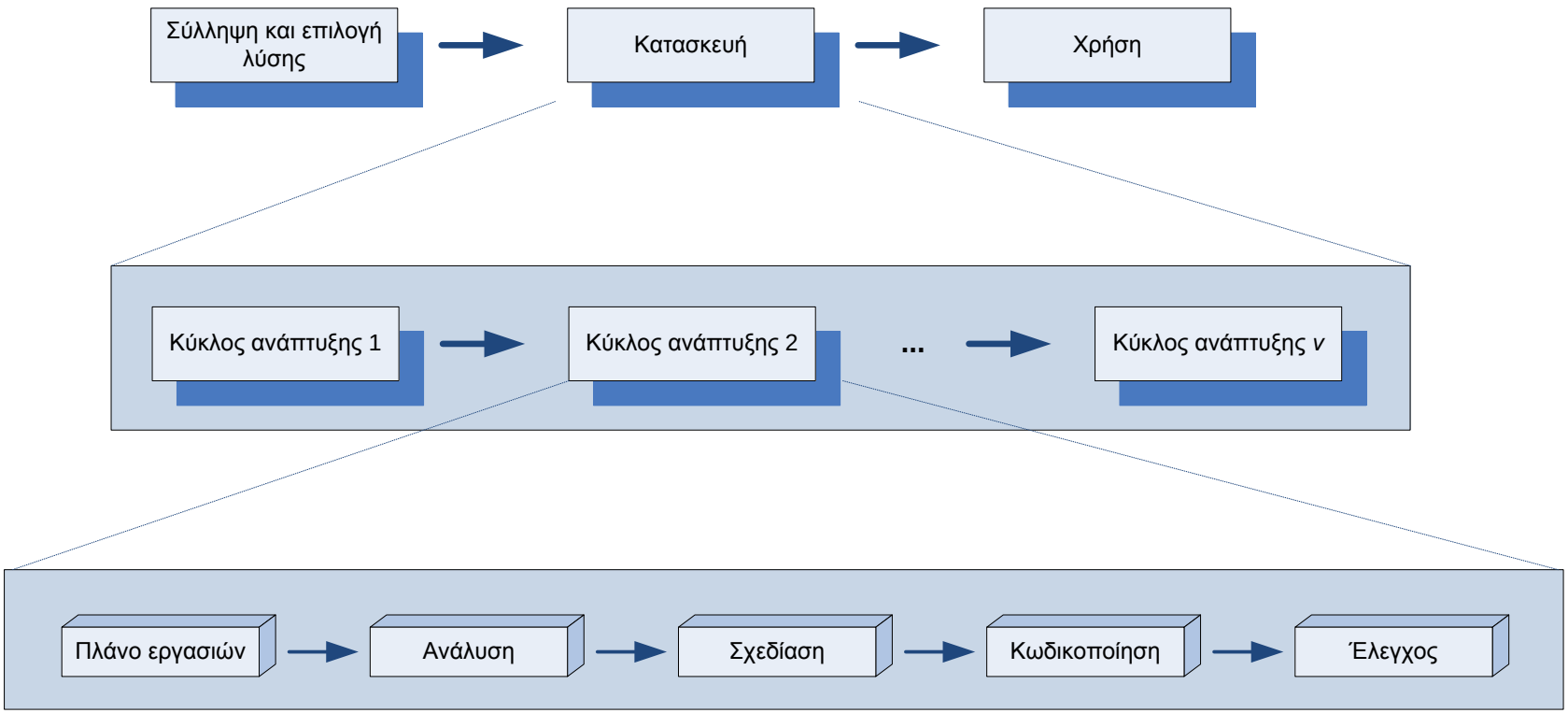
Ελευθερία στο κατασκευαστή λογισμικού

Εξάλειψη ογκωδέστατων παραδοτέων με σχέδια και προδιαγραφές

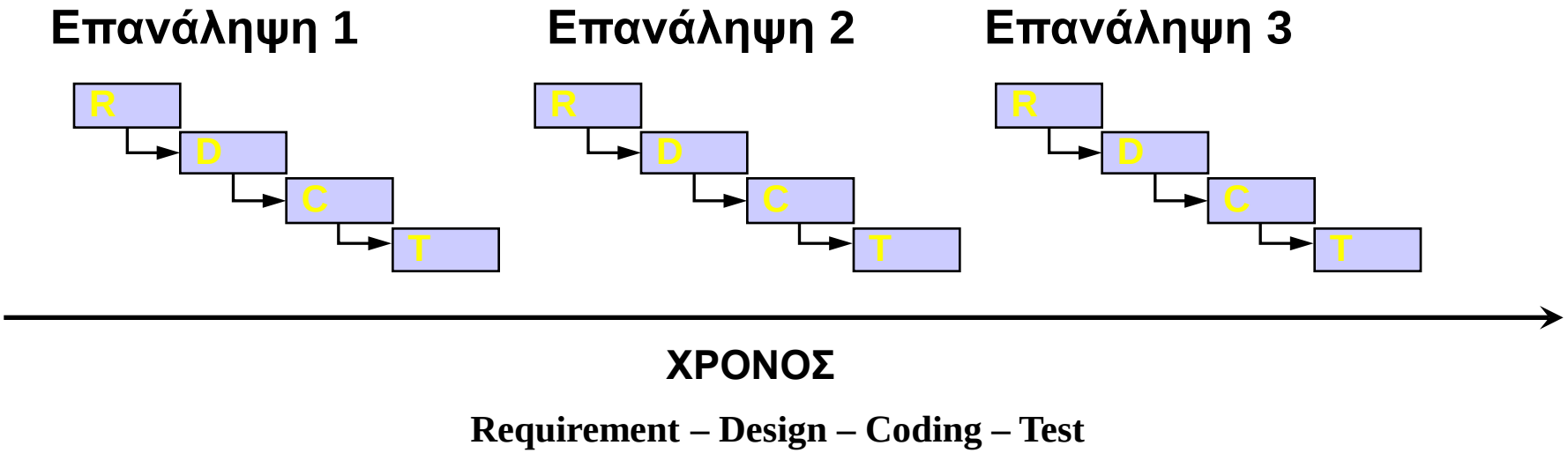
Εξειδίκευση κατευθύνσεων στο εκάστοτε περιβάλλον ανάπτυξης, για το συγκεκριμένο πρόβλημα, για τον συγκεκριμένο κατασκευαστή, για τον συγκεκριμένο πελάτη, κλπ.

Δεν είναι συνδεδεμένα με συγκεκριμένη μεθοδολογία ανάπτυξης πληροφοριακών συστημάτων.

Σύγχρονα μοντέλα κύκλου ζωής



Επαναληπτική Ανάπτυξη Συστημάτων



- Μειώνεται ο κίνδυνος
- Παράγει πρωτότυπα για επαλήθευση αποτελεσμάτων
- Κάθε επανάληψη περιλαμβάνει ολοκλήρωση και έλεγχο

Ενοποιημένη Διαδικασία (ΕΔ) – Unified Process (UP)

Η ΕΔ είναι μια μεθοδολογία

που χρησιμοποιεί τη UML γλώσσα για να περιγράψει τα παραδοτέα

Η ΕΔ προσδιορίζει

Ποιος κάνει τι;

Πότε το κάνει;

Πώς το κάνει;

έχοντας πάντα ως στόχο την ανάπτυξη του ΠΣ

Βασικές Αρχές ΕΔ – Φάσεις ΕΔ

Η ΕΔ βασίζεται

- Στο καλό χειρισμό των απαιτήσεων με το μοντέλο των περιπτώσεων χρήσης
- Στην ανάπτυξη της αρχιτεκτονικής του συστήματος ως βασικό συστατικό επιτυχίας
- Στη γραφική ανάπτυξη με χρήση UML
- Στην επαναληπτική διαδικασία ανάπτυξης (iterative development)

Οι **φάσεις του κύκλου ζωής** είναι τέσσερις:

- Σύλληψη (inception)
- Επεξεργασία (elaboration)
- Κατασκευή (construction)
- Μετάβαση (transition)

Φάση Σύλληψης (Inception)

- Στη φάση αυτή ορίζουμε
  - Το **όραμα** που έχουμε για το σύστημα
  - Τη **στρατηγική** που θέλουμε να υλοποιήσουμε με την ανάπτυξη του συστήματος
  - Τις **υψηλού επιπέδου απαιτήσεις** και περιπτώσεις χρήσης του συστήματος
  - Τους **κινδύνους** και τα **κέρδη** από την ανάπτυξη του συστήματος
  - Παραγωγή **αρχικών εκτιμήσεων** για το κόστος και διάρκεια των εργασιών ανάπτυξης
- Η φάση της σύλληψης δεν είναι καταγραφή απαιτήσεων. Είναι περισσότερο κάτι σαν **μελέτη σκοπιμότητας** ανάπτυξης του συστήματος

Φάση Επεξεργασίας (Elaboration)

Στη φάση αυτή:

- Αναπτύσσουμε και συγκεκριμενοποιούμε τους στρατηγικούς στόχους του συστήματος
- Καταγράφουμε τις **απαιτήσεις**
- Αναλύουμε τις απαιτήσεις λεπτομερώς
- Προσδιορίζουμε επακριβώς το **αντικείμενο των εργασιών** του έργου (scope)
- Αναπτύσσουμε τη **βασική αρχιτεκτονική** του συστήματος
- **Αντιμετωπίζουμε τους κινδύνους** που εντοπίσαμε στη φάση της σύλληψης
- **Βελτιώνουμε**, συγκεκριμενοποιούμε και επικαιροποιούμε τις **προβλέψεις** σχετικά με το κόστος, χρόνο, πόρους που απαιτεί το έργο

Φάση Κατασκευής (Construction)

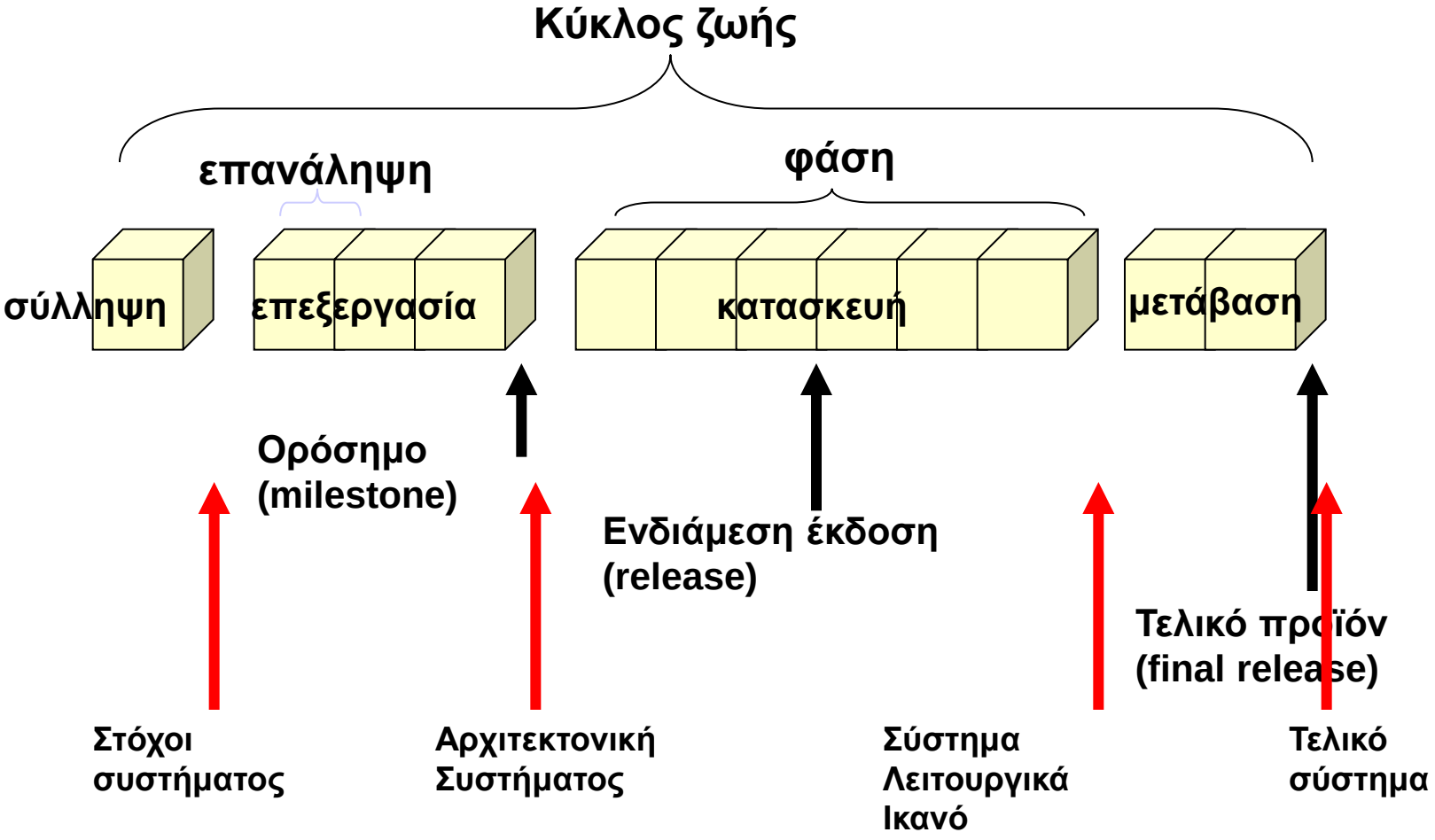
Είναι η φάση κατά την οποία υλοποιούμε και ελέγχουμε τμήματα του συστήματος για τα οποία:

- Έχουν διευκρινιστεί οι απαιτήσεις
- Και έχουν αντιμετωπισθεί οι κίνδυνοι

Φάση Μετάβασης (Transition)

- Είναι η φάση κατά την οποία το **τελικό προϊόν δίνεται** στον χρήστη
- Είναι η φάση η οποία έχει μεγάλο βαθμό **μεταβλητότητας** η οποία εξαρτάται από το συγκεκριμένο έργο και μπορεί να περιλαμβάνει:
  - Εκπαίδευση χρηστών
  - Μεταφορά δεδομένων
  - Δοκιμαστική λειτουργία νέου συστήματος
  - Παράλληλη λειτουργία παλαιού και νέου συστήματος
  - Beta testing

Φάσεις του Κύκλου Ζωής στην Ενοποιημένη Διαδικασία



Σύγχρονες μεθοδολογίες

ICONIX

Ευέλικτες μεθοδολογίες (eXtreme Programming (XP))

Η αντικειμενοστραφής μεθοδολογία (object oriented) ICONIX

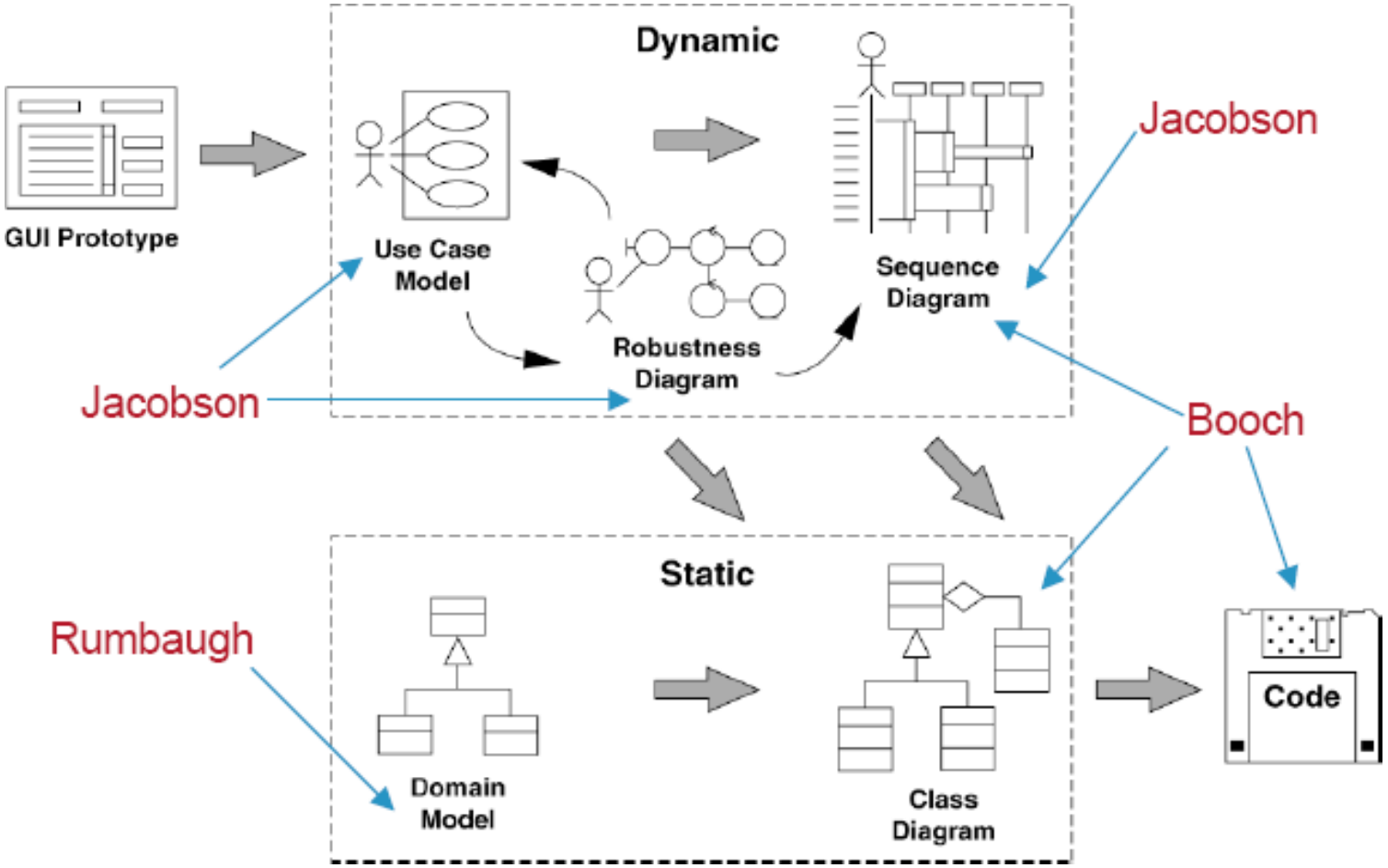
Προέκυψε ως σύνθεση των μεθόδων των Booch/Rumbaugh/Jacobson και υφίσταται πριν καν υπάρξει η UML (1992).

Βελτιώθηκε σε διάστημα πάνω των 10 ετών, εκατοντάδες εκπαιδευτικά workshops καθώς και από την εφαρμογή της σε ανάπτυξη λογισμικού.

Χρησιμοποιεί ένα ελάχιστο αλλά επαρκές υποσύνολο των διαγραμμάτων της UML τα οποία χρησιμοποιούνται από το σύνολο των υπαρχόντων αντικειμενοστραφών μεθόδων παγκοσμίως.

Κύρια συστατικά (διαγράμματα)

Κύρια συστατικά της Μεθοδολογίας ICONIX και από πού αυτά προέκυψαν:



Τα σημεία κλειδιά της ICONIX Μεθόδου

Αποφυγή της παράλυσης της ομάδας ανάπτυξης λογισμικού λόγω υπερβολικής ανάλυσης του προβλήματος

Απλοποιημένος τρόπος χρήσης της UML (μόνο 4 διαγράμματα)

Μινιμαλιστική αλλά επαρκέστατη

Υψηλός βαθμός ιχνηλασιμότητας

Βασισμένη στα βασικά ερωτήματα της αντικειμενοστρεφούς ανάλυσης και σχεδιασμού

Δουλεύει από τα έξω προς τα μέσα (από το χρήστη στην εφαρμογή)

Agile Methods and eXtreme Programming (XP)

Ευέλικτες Μέθοδοι και Ακραίος Προγραμματισμός

Ευέλικτες Μέθοδοι (Agile Methods)

Ευελιξία στον προγραμματισμό (Agility) είναι η ικανότητα της προσαρμογής και επαναπροσδιορισμού ενός αναπτυσσόμενου και συνεχώς εξελισσόμενου συστήματος στην περίπτωση που εμφανίζονται αλλαγές στις αρχικές θεωρήσεις και παραδοχές.

Οι Ευέλικτες μέθοδοι είναι:
Επαναληπτικές *(Iterative)*
Επαυξητικές *(Incremental)*
Αυτό-διοργανούμενες *(Self-Organizing)*
Προκύπτουσες *(Emergent)*

Agile Manifesto (Αρχές Ευέλικτων Μεθόδων)

1. Ικανοποίηση του πελάτη
2. Συχνή παράδοση λογισμικού
3. Η αλλαγή είναι ευπρόσδεκτη
4. Καθημερινή συνεργασία με πελάτες
5. Ικανό προσωπικό και περιβάλλον εμπιστοσύνης στην ομάδα
6. Διαπροσωπική συζήτηση για την ανταλλαγή πληροφοριών
7. Σωστή λειτουργία του λογισμικού που κατασκευάζεται
8. Εξασφάλιση σταθερού ρυθμού ανάπτυξης
9. Τεχνική αρτιότητα και καλός σχεδιασμός
10. Υλοποίηση στόχων με σύντομο και αποτελεσματικό τρόπο
11. Αυτό-διοργανονώμενες ομάδες
12. Επαναπροσδιορισμός της συμπεριφοράς της ομάδας

Διαθέσιμες Ευέλικτες Μέθοδοι

Agile Modeling

Adaptive Software Development (ASD)

Crystal methods

Dynamic System Development Methodology (DSDM)

eXtreme Programming (XP)

Feature Driven Development (FDD)

Lean Development

Scrum

Ακραίος Προγραμματισμός (XP)

Ένας ελαφρύς, αποτελεσματικός, χαμηλού-κινδύνου, ευέλικτος, προβλέψιμος, επιστημονικός και ευχάριστος τρόπος για την ανάπτυξη λογισμικού

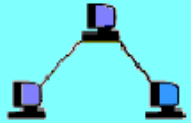
Βασίζεται σε τέσσερις αξίες στην απλότητα, επικοινωνία, ανατροφοδότηση και κουράγιο. Η αποτελεσματικότητα της οφείλεται στη στενή συνεργασία της ομάδας κάτω από απλές πρακτικές με συχνή ανατροφοδότηση που τους επιτρέπει να αξιολογούν την πρόοδο τους και να προσαρμόζουν τις πρακτικές στις τρέχουσες ανάγκες.

Το πρώτο XP-πρόγραμμα ήταν το πρόγραμμα μισθοδοσίας στην Chrysler Comprehensive Compensation (C3), (Beck – Highsmith, 1998). (Kent Beck)

Οι 12 πρακτικές (XP – practices)

Το παιχνίδι του σχεδιασμού	-	The Planning Game
Μικρές εκδόσεις	-	Small releases
Αρχιτεκτονική εικόνα	-	Metaphor
Απλή Σχεδίαση	-	Simple design
Έλεγχοι πριν την κωδικοποίηση	-	Testing
Ανακατασκευή κώδικα	-	Refactoring
Προγραμματισμός ανά ζεύγη	-	Pair Programming
Συλλογική ιδιοκτησία κώδικα	-	Collective Ownership
Διαρκείς ενοποιήσεις του κώδικα	-	Continuous Integration
Υποφερτός ρυθμός εργασίας	-	Sustainable pace
Διαρκής παρουσία πελάτη	-	On-site customer
Πρότυπα κωδικοποίησης	-	Coding standards

Οι πρακτικές σε εικόνες...

Το παιχνίδι του σχεδιασμού  =συμβολή του πελάτη	Προγραμματισμός ανά ζεύγη  =λιγότερα λάθη στον κώδικα	Έλεγχοι πριν την κωδικοποίηση  =ποιοτικό λογισμικό	Ανακατασκευή κώδικα  =καθαρός κώδικας
Σταθερές κωδικοποίησης  =καλύτερη επικοινωνία	Απλή Σχεδίαση  =ευελιξία	Μικρές εκδόσεις  =προσαρμοστικότητα	Διαρκείς ενοποιήσεις του κώδικα  =διαρκής ανάδραση
Συλλογική ιδιοκτησία κώδικα  =διαμοιρασμός της γνώσης	Διαρκή παρουσία Πελάτη  =ξεκαθάρισμα των απαιτήσεων	Αρχιτεκτονική εικόνα  =κατανόηση του συστήματος	Υποφερτός ρυθμός εργασίας  =αποδοτικότητα

Γιατί στα άκρα (extreme);

Αν η **ανατροφοδότηση** είναι καλή, τότε να την λαμβάνεις συνεχώς **(pair programming, planning game, on-site customer)**

Αν η **επιθεώρηση κώδικα** είναι ωφέλιμη, τότε να κάνετε συνεχώς **επιθεωρήσεις (pair programming)**

Αν ο **έλεγχος κώδικα** είναι ωφέλιμος, τότε ελέγχετε συνεχώς **(unit tests - acceptance tests)**

Αν ο **ανασχεδιασμός** είναι καλός, τότε ανασχεδιάζετε συνεχώς **(refactoring)**

Αν η **απλότητα** είναι καλή, τότε κάνε το απλούστερο που μπορεί να δουλέψει **(simple design)**

Βιβλιογραφία

Τεχνολογία Λογισμικού (2000), Βασίλειος Βεσκούκης, Τόμος Α, ISBN: 960–538–097–8, Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο.

Ανάπτυξη Προηγμένων Πληροφοριακών Συστημάτων, (2007), David Avison, Guy Fitzgerald, Κωδικός Βιβλίου στον Εύδοξο: 1177, ISBN 960-8105-96-X, ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ ΜΟΝ. ΕΠΕ.

Online Εξ Αποστάσεως Εκπαίδευση – Από τη Θεωρία στην Πράξη, (2015), Σοφός (Λοΐζος) Αλιβίζος, Κώστας Απόστολος, Παράσχου Βασίλειος, Εκδόσεις ΣΕΑΒ, Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα (www. kallipos .gr).

Σχεδιασμός Εκπαιδευτικού Λογισμικού, (2005), Παναγιωτακόπουλος Χ., Πιερρακέας Χ., Πιντέλας Π., ISBN 978-960-375-579-1, Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο.

Πληροφοριακά Συστήματα, Σύγχρονη Ανάλυση & Σχεδίαση, (2016, 6η εκδ.), Hoffer J., George J., Valacich J., Κωδικός Βιβλίου στον Εύδοξο: 18548910, ISBN: 978-960-418-331-9, Εκδόσεις Τζιόλα.

Σχεδιασμός και αξιολόγηση εκπαιδευτικού λογισμικού, (2014), (Ενότ.3: Μοντέλα Σχεδιασμού, Μοντέλο ADDIE), Νικόλαος Τσέλιος, Πανεπιστήμιο Πατρών.