ΑΝΑΠΤΥΞΗ & ΣΧΕΔΙΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

Διδάσκουσα: Μπίμπη Ματίνα

ΠΜΣ, Πανεπιστήμιο Θεσσαλίας
Τμήμα Μηχανικών Η/Υ Τηλεπικοινωνιών & Δικτύων
sbibi@inf.uth.gr

Αντικείμενο Μελέτης

- Κατανόηση του τρόπου επινόησης και υλοποίησης του λογισμικού
- Αντικειμενοστρεφής ανάλυση και σχεδίαση με τη UML
- Μεθοδολογίες, τεχνικές και εργαλεία ανάπτυξης λογισμικού.
- Έλεγχος ορθής λειτουργίας λογισμικού
- Τεχνικές διαχείρισης έργων λογισμικού

Βιβλιογραφία- Πηγές

- **Βασικά συγγράμματα**
  - Α. Χατζηγεωργίου, "Αντικειμενοστρεφής Σχεδίαση: UML, Αρχές, Πρότυπα και Ευρετικοί Κανόνες" των Εκδόσεων, εκδ. Κλειδάριθμος
  - M. Fowler, "Εισαγωγή στη UML", εκδ. Κλειδάριθμος
  - I. Sommerville, "Βασικές αρχές Τεχνολογίας Λογισμικού", εκδ. Κλειδάριθμος
- **Συμπληρωματικά συγγράμματα**
  - R. Pressman, "Software Engineering: A Practitioner's Approach", εκδ. McGraw- Hill.
  - E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, Boston, MA, 1995
  - C. Horstmann, Object-Oriented Design & Patterns, Wiley, John Wiley & Sons, New York, 2004
- **Διαφάνειες, Σημειώσεις, Διαδίκτυο**

Εκπαιδευτικό υλικό μαθήματος

- **Ιστοσελίδα: http://eclass.uth.gr/**
  - Χρονοδιάγραμμα μαθήματος
  - Ανακοινώσεις
  - Διαφάνειες μαθημάτων
  - Άρθρα
  - Χρήσιμοι σύνδεσμοι
  - Λογισμικό
- **Επικοινωνία**
  - Δευτέρα 13.00 -14.00
  - Email: **sbibi@inf.uth.gr**

Περιεχόμενο Μαθήματος

Εβδομάδα	Περιεχόμενο
1η Εβδομάδα	Εισαγωγή στη Τεχνολογία Λογισμικού- Μοντέλα Κύκλου Ζωής
2η Εβδομάδα	Εισαγωγή στη UML
3η Εβδομάδα	Υλοποίηση της UML - Πολυμορφισμός
4η Εβδομάδα	Αρχές Σχεδίασης
5η Εβδομάδα	Αρχές Σχεδίασης
6η Εβδομάδα	Πρότυπα Σχεδίασης
7η Εβδομάδα	Πρότυπα Σχεδίασης
8η Εβδομάδα	Ευρετικοί Κανόνες
9η Εβδομάδα	Ευρετικοί Κανόνες
10η Εβδομάδα	Μετρικές Κώδικα
11η Εβδομάδα	Τεχνικές Ελέγχου
12η Εβδομάδα	Διαχείριση έργων Λογισμικού
13 Εβδομάδα	Παρουσίασεις εργασιών

Αξιολόγηση/ Εξέταση

- **Εξέταση**
- **Ομαδική Εργασία- Συντήρηση έργου λογισμικού**
  - Υποχρεωτική
  - Ομάδες έως 2 άτομα
- **Ατομική Εργασία- Ενασχόληση και παρουσίαση εργαλείου CASE**
  - Ατομική
- **Βαθμολογία**
  - Τελική γραπτή εξέταση 50%
  - Εργασία 30%
  - Συμμετοχή 10%
  - Παρουσίαση 10%

Διάλεξη 1

Εισαγωγή στην Τεχνολογία Λογισμικού

Περιεχόμενα

- ❑ **Η σημασία της Τεχνολογίας Λογισμικού**
- ❑ **Το ιστορικό της Τεχνολογίας Λογισμικού**
- ❑ **Η έννοια του «καλού λογισμικού»**
- ❑ **Προσέγγιση από πλευράς συστήματος και σχεδίασης**

Το λογισμικό

- 1950, αρχή
- Έδωσε δυνατότητες σε υπάρχουσες τεχνολογίες (πχ. Τηλεπικοινωνίες)
- Δημιούργησε νέες επιστήμες (π.χ. γενετική βιολογία)
- Βελτίωση ποιότητα ζωής
  - Υγεία, τηλεϋπηρεσίες, έρευνα, κ.λπ.

Που βρίσκεται το λογισμικό;

- Άϋλο («εκτελείται», αποθηκεύεται)
- Δεν είναι αντικείμενο των 5 αισθήσεών μας
- Δεν φθείρεται με τον χρόνο
- Δεν υπακούει στους νόμους παραγωγής-ανάπτυξης των υλικών αγαθών
- Διαφοροποιείται από τον τρόπο αξιοποίησης των υλικών αγαθών

Ορισμός λογισμικού

- Σύνολο προγραμμάτων που περιλαμβάνουν
  - δομές δεδομένων (διαχείριση πληροφορίας)
  - Εντολές (παρέχουν λειτουργίες)
  - Τεκμηρίωση (περιγράφει τρόπο λειτουργίας των προγραμμάτων)

Η τεχνολογία λογισμικού

❑ «Είναι κλάδος της πληροφορικής που ασχολείται με την μελέτη και την εφαρμογή των συστηματικών, μεθοδικών και ποσοτικοποιημένων προσεγγίσεων για την ανάπτυξη, λειτουργία και συντήρηση του λογισμικού» (IEEE 90)

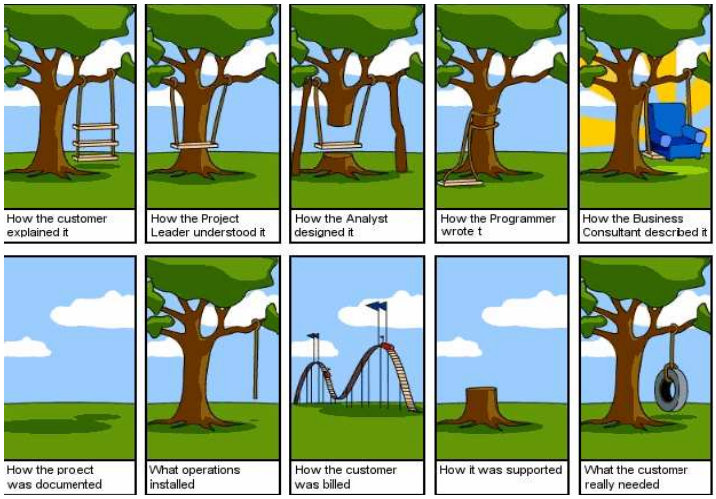
❑ Δεν έχει αποκτήσει ακόμη τα χαρακτηριστικά της επιστήμης, που θα της επιτρέπουν να αποδεικνύει την ύπαρξη βέλτιστης, ορθής, πλήρους και αξιόπιστης λύσης

Software Engineering is

- **David Parnas: "the multi-person construction of multi-version software"**
- **Ghezzi: "the application of engineering to software"**
- **Roberts: "the discipline of writing programs that can be understood and maintained by others"**
  - **in contrast to:**
- **"Programming is the discipline of writing programs that cannot be understood and maintained by anyone else than the author"**

Η Τεχνολογία Λογισμικού

- **Τέλη 1960 «κρίση λογισμικού»**
- **Μεγάλο ποσοστό των έργων παρουσίαζαν:**
  - Αποκλίσεις από
    - Λειτουργικότητα που παρήγγειλαν οι πελάτες
    - Χρονοδιάγραμμα ανάπτυξης
    - Οικονομικό προγραμματισμό
    - Προβλεπόμενο κόστος περιβάλλοντος λειτουργίας
  - Αδυναμία εξέλιξης του λογισμικού ώστε να προσαρμοστεί στις ανάγκες των πελατών



Ιδιαιτερότητες έργων λογισμικού

Διαφέρουν τα έργα λογισμικού από τα άλλα έργα ?

Test 1: Έστω ότι ένα έργο λογισμικού έχει καθυστερήσει σε σχέση με το χρονοδιάγραμμα του. Αν είσαστε ο project manager με τι τρόπους μπορείτε να το επιταχύνετε ?

Αν απαντήσατε με αύξηση των ατόμων ή με χρήση νέων εργαλείων ….. wrong answer ! (Mythical Man-Month)

Test 2: Έστω ότι είστε ο project manager σε ένα έργο λογισμικού (μη έχοντας σχέση με προγραμματισμό). Πώς θα παρακολουθήσετε την πορεία του ?

Προφανώς εξαρτάστε από τις αναφορές των υπολοίπων !!

Test 3: Πουλάτε ένα προϊόν λογισμικού και σας ρωτούν να προσδιορίσετε την ποιότητα του. Τι θα αναφέρατε ?

Προφανώς δεν υπάρχουν σαφή χαρακτηριστικά ποιότητας !!!

Γιατί είναι σημαντικό το μάθημα?

- Τα περισσότερα λάθη (54%) ανακαλύπτονται μετά την κωδικοποίηση και τον έλεγχο.
- Σχεδόν μισά (45%) από τα λάθη εντοπίζονται στις απαιτήσεις και τη σχεδίαση.
- Τα περισσότερα λάθη κατά τη φάση της ανάλυσης απαιτήσεων (77%) δεν οφείλονται στους αναλυτές αλλά προκύπτουν λόγω λανθασμένων δεδομένων, ασυνέπειες, παραλήψεις και ασάφειες.
- Η διόρθωση λαθών στις απαιτήσεις μπορεί να κοστίσει και 100 φορές παραπάνω από τα λάθη υλοποίησης.
- Τα λάθη στην καταγραφή απαιτήσεων μπορούν να εντοπιστούν καθώς οι τεχνικές επιθεώρησης έχουν αποδειχτεί αποτελεσματικές και μπορούν να εφαρμοστούν και στο σχέδιο και στον κώδικα

Τεχνολογία λογισμικού (1/2)

Προϊόν συστήματος λογισμικού

- **Λογισμικό**
  - Προγράμματα υπολογιστή
- **Προϊόν λογισμικού**
  - Λογισμικό που μπορεί να πουληθεί σε κάποιο πελάτη (γενικής χρήσης, κατά παραγγελία). Περιλαμβάνεται και η σχετική τεκμηρίωση όπως απαιτήσεις, σχέδιο, εγχειρίδιο.
- **Σύστημα Λογισμικού**
  - Επιτρέπει τη συνεργασία και την ενσωμάτωση με άλλα συστήματα λογισμικού.

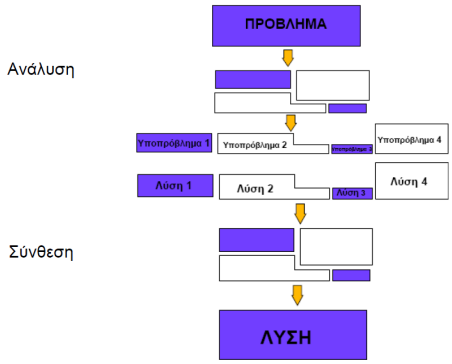
Τεχνολογία Λογισμικού
είναι ο τεχνικός κλάδος που ασχολείται με όλες τις πτυχές ανάπτυξης λογισμικού από τα πρώτα στάδια της εξαγωγής προδιαγραφών μέχρι τη συντήρηση του συστήματος (Somerville)

Τεχνολογία Λογισμικού

- **Επιλύει προβλήματα με τη βοήθεια του λογισμικού**
  - ανάλυση προβλήματος
  - σύνθεση της λύσης
- **Με τη χρήση:**
  - Μεθόδων ή τεχνικών: διαδικασία παραγωγής αποτελέσματος
  - Εργαλείων: βοήθημα ή αυτοματοποιημένο σύστημα
  - Διαδικασιών: συνδυασμό εργαλείων και μεθόδων
  - Υποδειγμάτων: συγκεκριμένη προσέγγιση ή μεθοδολογία

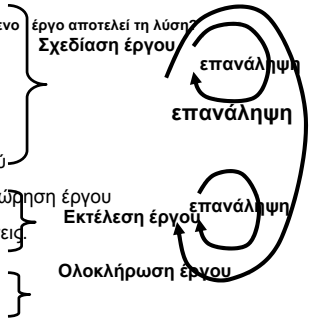
- **Η Τεχνολογία Λογισμικού έχει ως στόχο την σχεδίαση και υλοποίηση λογισμικού υψηλής ποιότητας.**

Επίλυση προβλημάτων



Δραστηριότητες διαχείρισης

- Συγγραφή πρόταση- **Γιατί το προτεινόμενο έργο αποτελεί τη λύση?**
- Κοστολόγηση έργου
- Σχεδίαση + Προγραμματισμός έργου
  - Τι θα γίνει? Πότε θα γίνει?
- Επιλογή και αξιολόγηση προσωπικού
- Έλεγχος, Παρακολούθηση και Επιθεώρηση έργου
- Συγγραφή αναφορών και παρουσιάσεις
- (Ανάλυση έργου)



Συχνότερες αιτίες αποτυχίας

- **Μη ρεαλιστικοί στόχοι/ στόχοι που δεν είναι καλά καθορισμένοι**
- **Λάθος ορισμός απαιτήσεων**
- **Ανικανότητα χειρισμού πολυπλοκότητας έργων**
- **Κακές εκτιμήσεις για τους απαιτούμενους πόρους**
- **Κακή αναφορά της κατάστασης του έργου**
- **Κακή διαχείριση του έργου**
- **Κακή διαχείριση κινδύνων**
- **Ελλιπής επικοινωνία πελάτη, προγραμματιστή, χρήστη**
- **Κακές προγραμματιστικές τεχνικές**
- **Χρήση ανώριμης τεχνολογίας**

Παράγοντες επιτυχίας

Έρευνες έχουν δείξει ότι για την επιτυχή ολοκλήρωση ενός έργου…

- **Επιχειρησιακή υποστήριξη**
  - Στάση διοίκησης
  - Καθορισμός στόχων και εμβέλειας
- **Εμπλοκή χρηστών**
  - Συμμετοχή σε όλη την πορεία του έργου
- **Ικανός διοικητής έργου**
- **Σαφείς επιχειρησιακοί στόχοι**
- **Εστιασμένο πεδίο εφαρμογής του προϊόντος**
  - Κοινή συμφωνία οριοθέτησης της λειτουργικότητας σε συνδυασμό με εφικτότητα χρονοδιαγράμματος και προϋπολογισμένου κόστους
- **Πρότυπες υποδομές**
  - «κώδικας υποδομής»
  - Χρήση έτοιμων, διαδεδομένων ώριμων και δοκιμασμένων υποδομών, τεχνολογιών και εργαλείων

Τι είναι η διεργασία λογισμικού?

- **Ένα σύνολο δραστηριοτήτων που έχουν ως στόχο την ανάπτυξη/ εξέλιξη του λογισμικού**
- **Γενικές δραστηριότητες που εντοπίζονται σε όλες τις διεργασίες ανάπτυξης είναι:**
  - Προδιαγραφή – τι πρέπει να κάνει το σύστημα, περιορισμοί ανάπτυξης
  - Ανάπτυξη- παραγωγή του συστήματος
  - Επικύρωση- έλεγχος αν το λογισμικό είναι αυτό που θέλει ο πελάτης
  - Εξέλιξη- αλλαγή λογισμικού συναρτήσει των μεταβαλλόμενων απαιτήσεων

Τι είναι ένα μοντέλο διεργασίας λογισμικού?

- **Μια απλοποιήμενη αναπαράσταση της διεργασίας ανάπτυξης λογισμικού παρουσιασμένη με μια συγκεκριμένη οπτική.**
- **Παραδείγματα οπτικών διεργασίας αποτελούν:**
  - Οπτική ροής εργασιών: σειρά δραστηριοτήτων
  - Οπτικής ροής δεδομένων: ροή πληροφοριών
  - Οπτική ρόλου/ δράσης: ποιος κάνει τι
- **Γενικά μοντέλα**
  - Καταρράκτης
  - Επαναληπτική ανάπτυξη
  - Βασισμένα σε συνιστώσες

Η διεργασία ανάπτυξης λογισμικού

Συντήρηση λογισμικού

- **Κύκλος Ζωής Λογισμικού (Software Life Cycle):**
  - **Από τη σύλληψη του έργου μέχρι την απόσυρσή του.**
- Ο τρόπος που παράγουμε λογισμικό περιλαμβάνει:
  - **Μοντέλο κύκλου ζωής**
  - **Ανθρώπινο δυναμικό (managers, experts, programmers, …)**
  - **Εργαλεία CASE**
- **Φάσεις κύκλου ζωής:**
  1. *Απαιτήσεων (Requirements):* Εξαγωγή από ανάγκες πελατών.
  2. *Ορισμού Προδιαγραφών (Specification):* Τι πρέπει το προϊόν να κάνει (specification doc, software project management plan).
  3. *Σχεδιασμού (Design):* Πώς θα το κάνει (architectural and detailed design).

Συντήρηση λογισμικού

4. ***Υλοποίησης*** *(Implementation):* Κώδικας και έλεγχος.
5. ***Συνένωσης Κώδικα*** *(Integration):* Συνένωση ανεξάρτητων τμημάτων και συνολικός έλεγχος (δημιουργών και πελάτη – acceptance test).
6. ***Συντήρησης*** *(Maintenance):* Αλλαγές στο προϊόν μετά την αποδοχή του από πελάτη:
   - **Διορθωτική(corrective or repair) – 17.5%**
     - **Ενισχυτική: Τελειοποίησης(perfective) – 60.5%**
     - **Προσαρμογής(adaptive) –18%**
7. ***Απόσυρσης*** *(Retirement)* προϊόντος από λειτουργία.

"Καλό"είναι το λογισμικό που συντηρείται—το"κακό"πετιέται

Επιπτώσεις από την εφαρμογή της MEK2 στη φάση της συντήρησης του λογισμικού

Θέματα ποιότητας (1/5)

Καλό λογισμικό → ποιότητα

Ποιότητα:

- **Προϊόντος**
- **Διεργασίας**
- **Σε σχέση με το επιχειρηματικό περιβάλλον**

Θέματα ποιότητας (2/5)

- Άποψη χρηστών με βάση τη λειτουργικότητα, τις αστοχίες, την ευκολία χρήσης.
- Άποψη δημιουργών με βάση τα εσωτερικά χαρακτηριστικά.
- Μοντέλα συσχέτισης της άποψης των χρηστών με αυτή των δημιουργών

Θέματα ποιότητας (3/5)

Ποιότητα προϊόντος

1. **Θα συναντήσουμε σφάλματα ?**
   - Που και πότε?
   - Πως θα τα εντοπίσουμε?
   - Πως θα αποφύγουμε την εξέλιξη του σφάλματος σε αστοχία?

Ποιότητα διεργασίας

2. **Πως θα γίνει αποτελεσματικότερη η διεργασία?**
   - Capability Maturity Model (CMM)
   - ISO 9000
   - Software Process Improvement and Capability dEtermination (SPICE)

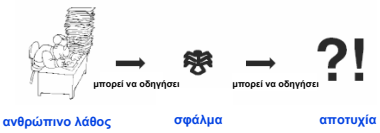
Θέματα ποιότητας (4/5)

Σε σχέση με το επιχειρηματικό περιβάλλον

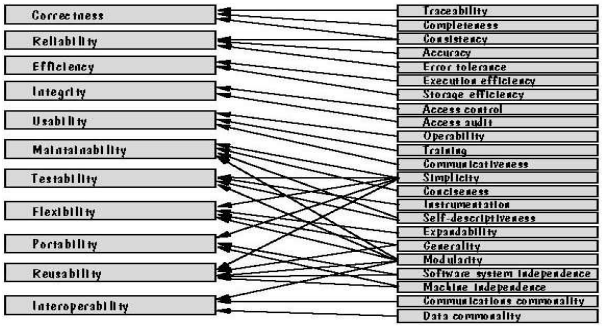
- **Η ποιότητα εκτιμάται βάσει των υπηρεσιών και των προϊόντων που παρέχει η εταιρεία στην οποία εγκαθίσταται το λογισμικό.**
- **Απόδοση επένδυσης (Return On Investment, ROI)**
  - Το χρηματικό κόστος (κυβερνήσεις)
  - Την απαιτούμενη προσπάθεια (εταιρείες)
    - **Εκπαίδευση**
    - **Χρονοδιάγραμμα**
    - **Κίνδυνος**
    - **Παραγωγικότητα**
    - **Ποιότητα**

Θέματα ποιότητας (5/5)

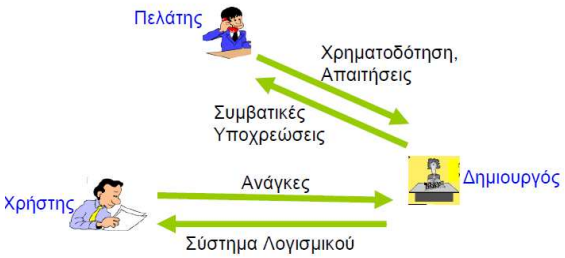
- **Ελάττωμα/ σφάλμα**
  - Αφορά την εσωτερική συμπεριφορά του συστήματος
- **Αστοχία/ δυσλειτουργία**
  - Αφορούν την εξωτερική συμπεριφορά του συστήματος



Ποιότητα προϊόντος (μοντέλο McCall)



Συμμετέχοντες σε ένα έργο λογισμικού



Πελάτης

- Καθορίζει τι θα κατασκευαστεί.
- Παρέχει προδιαγραφές των απαιτήσεων.
- Χρηματοδοτεί την ανάπτυξη.
- Παραλαμβάνει και αξιολογεί το τελικό προϊόν.

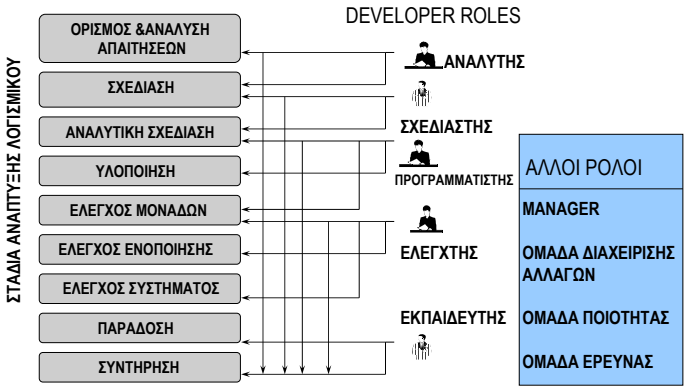
Χρήστης- Διαχειριστής

- Σύνδεσμος μεταξύ πελάτη – δημιουργού.
- Διαπραγματεύεται χρόνο παράδοσης και κόστος.
- Χρονοπρογραμματίζει και επιβλέπει το έργο.
- Θέτει περιορισμούς στο χρόνο και στην προσπάθεια στο δημιουργό.

Δημιουργός

- Καθορίζει πώς θα κατασκευαστεί το προϊόν.
- Δημιουργεί το προϊόν (λογισμικό).
- Προσπαθεί για την ικανοποίηση του πελάτη.

Η ομάδα ανάπτυξης



Προαπαιτούμενα για την Ανάπτυξη Έργου

- Ποιες οι προϋποθέσεις για την ανάπτυξη έργου;
  - **Σαφής διατύπωση απαιτήσεων.**
  - **Σύνολο τεχνικών και εργαλείων.**
- Πλάνο:
  - **Εργασίες που θα εκτελεστούν.**
  - **Σειρά εκτέλεσης.**
  - **Κριτήρια μετάβασης.**
  - **Ενδιάμεσοι σημαντικοί σταθμοί - παραδοτέα.**
  - **Αξιολόγηση πορείας.**

Μοντέλα Διεργασιών Ανάπτυξης Λογισμικού

- Πλάνο: Μοντέλο διεργασίας.
- Μοντέλο: Αναπαράσταση της πραγματικότητας.
- Διεργασία: Σύνολο διατεταγμένων, διακριτών βημάτων.
  - **Αξιοποιεί πόρους.**
  - **Πληροί περιορισμούς.**
  - **Καταλήγει σε αποτέλεσμα.**

Ορισμός Διεργασίας

Διεργασία: **ένα σύνολο διατεταγμένων εργασιών που περιλαμβάνει:**

- **Όλες τις κύριες δραστηριότητές της**
- **Τους πόρους της οι οποίοι υπόκεινται σε περιορισμούς**
- **Υποδιεργασίες που ακολουθούν μια ιεραρχία**
- **Κριτήρια εισόδου και εξόδου κάθε δραστηριότητας**
- **Ακολουθία δραστηριοτήτων**
- **Κατευθυντήριες αρχές της διεργασίας, στόχοι**
- **Περιορισμοί σε πόρους ή προϊόντα**

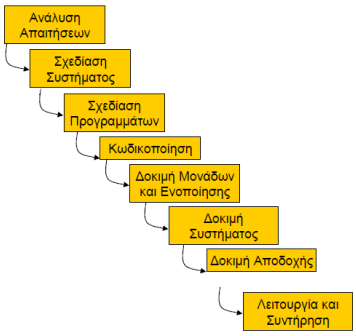
Βασικές κατηγορίες μοντέλων διεργασίας ανάπτυξης λογισμικού

- **Μοντέλο καταρράκτη**
  - Ξεχωριστές και διακριτές φάσεις προδιαγραφών και υλοποίησης
- **Εξελικτικό μοντέλο**
  - Οι φάσεις των προδιαγραφών, της υλοποίησης και της επικύρωσης παρεμβάλλονται
- **Μοντέλο βασισμένο σε συνιστώσες**
  - Το σύστημα προκύπτει από υπάρχουσες συνιστώσες
- **Παραλλαγές, συνδυασμοί των παραπάνω μοντέλων**

Μοντέλο καταρράκτη

- **Γραμμικό μοντέλο ('70)**
- **Ολοκλήρωση κάθε δραστηριότητας**
- **Ορισμός οροσήμων – προτύπων παραδοτέων (Πρότυπο 2167-Α)**
- **Επιμερισμός εργασιών (κατάρτηση καταλόγου)**
- **Απλότητα**
- **Εύκολη ενημέρωση πελατών (ενδιάμ. προϊόντα)**

Μοντέλο καταρράκτη



Μειονεκτήματα- Πλεονεκτήματα μοντέλου καταρράκτη

- Πλεονεκτήματα
  - **Καλός 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.**
  - **Ο πελάτης βλέπει τι τελικά αγοράζει πολύ αργά !!!**
  - **Έλλειψη καθοδήγησης για το χειρισμό δραστηριοτήτων (μετασχηματισμού λογ/κού)**

Γραμμικό μοντέλο με δημιουργία πρωτοτύπων

Πρωτότυπο: **μερικώς ανεπτυγμένο προϊόν**

- Μέσο επικοινωνίας Πελάτη – Δημιουργού, αναγκαίο για την κατανόηση
- 1/10 κόστους αλλαγές στην ανάλυση
- Δομική εναλλακτικών στρατηγικών σχεδίασης

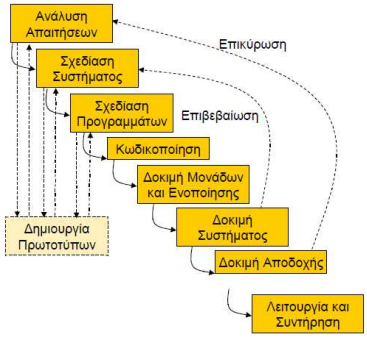
- **Αναθεωρήσεις, διορθώσεις και εμπλουτισμός σε κάθε φάση, μέχρι την συμφωνία του πελάτη**

- **Στόχος: μείωση κινδύνων και αβεβαιότητας**

Επικύρωση: **υλοποίηση προδιαγραφών**

Επαλήθευση: **έλεγχος ορθότητας λειτουργιών**

Γραμμικό μοντέλο με δημιουργία πρωτοτύπων



Μοντέλο Πρωτοτύπων- Πλεονεκτήματα

- Αντιμετωπίζει την ασάφεια στις απαιτήσεις.
- Παρέχει τη δυνατότητα στον πελάτη να αλλάξει γνώμη πριν υπογράψει.
- Μείωση χρόνου ανάπτυξης.
- 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 / ομάδας ανάπτυξης.

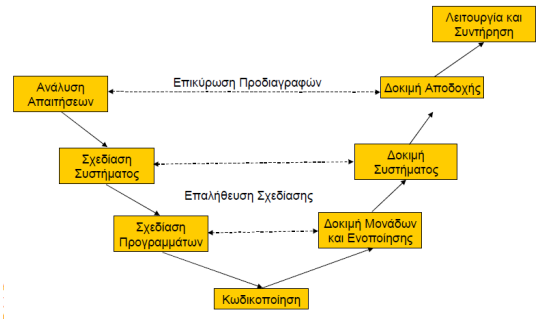
Μοντέλο Πρωτοτύπων- Μειονεκτήματα

- Έλλειψη παρατήρησης στη διαδικασία (αδυναμία πρόβλεψης 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.

Το μοντέλο V

- **Παραλλαγή γραμμικού μοντέλου**
  - Γερμανικό Υπουργείο Αμυνας ('92)
- **Αριστερό σκέλος: ανάλυση, σχεδίαση**
- **Δεξιό σκέλος: δοκιμές, συντήρηση**
- **Σύνδεση αριστερού με δεξιού σκέλους για την επαλήθευση και επικύρωση.**
- **Έλεγχοι αποδοχής από πελάτη**
- **Εισάγει επαναληπτικές και αναθεωρητικές εργασίες**
- **Εστίαση στις δραστηριότητες και ορθότητα**

Το μοντέλο V



Μοντέλο ανάπτυξης σε φάσεις

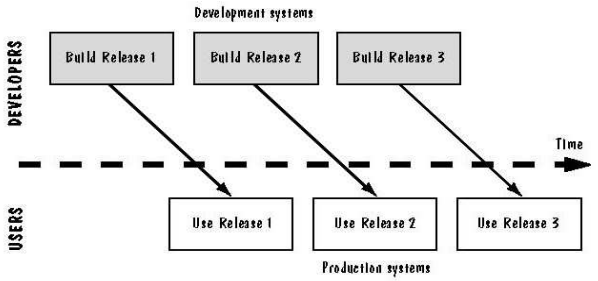
2 συστήματα λειτουργούν παράλληλα

- Σύστημα ανάπτυξης
- Σύστημα λειτουργίας/παραγωγής

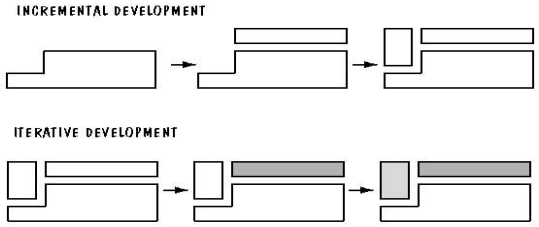
2 μέθοδοι ανάπτυξης

- Αυξητική
- Επαναληπτική

Μοντέλο ανάπτυξης σε φάσεις



Μοντέλο ανάπτυξης σε φάσεις



Μοντέλο ανάπτυξης σε φάσεις

Παράδειγμα: Πακέτο Επεξεργασίας κειμένου

Πλεονεκτήματα

- **Εκπαίδευση χρηστών**
- **Προώθηση στην αγορά νέων λειτουργιών**
- **Ευελιξία στην αντιμετώπιση προβλημάτων**
- **Επικέντρωση σε διαφορετικά προβλήματα**

Διαχείριση Κινδύνου

- Διαχειριστής έργου: Επιδιώκει ελαχιστοποίηση του ρίσκου.
- Ρίσκο: μέτρο της αβεβαιότητας ως προς το αποτέλεσμα.
- Δραστηριότητες υψηλού ρίσκου αυξάνουν το κόστος και προκαλούν καθυστερήσεις.
- Το ρίσκο σχετίζεται με την ποσότητα και την ποιότητα της διαθέσιμης πληροφορίας. Όσο λιγότερη πληροφορία τόσο μεγαλύτερο το ρίσκο.

Διαχείριση Κινδύνου

- Κατά τη διάρκεια ανάπτυξης εισαγωγή λογισμικού από ανταγωνιστή.
- Αποχώρηση ατόμων από την ομάδα ανάπτυξης.
- Ανικανότητα εμπρόθεσμης υλοποίησης ορισμένων σταδίων.
- Το λογισμικό πληροί τις προδιαγραφές αλλά έχει πολύ μεγάλο χρόνο εκτέλεσης.
- Μεγάλη κατανάλωση μνήμης.
- Εμφάνιση νέων εργαλείων ανάπτυξης.
- Λάθος κατανόηση μιας απαίτησης.
- Αλλαγή ορισμένων από τις απαιτήσεις.
- Αλλαγή του υλικού.
- Μείωση προϋπολογισμού.
- Βλάβες στο υλικό της ομάδας ανάπτυξης.

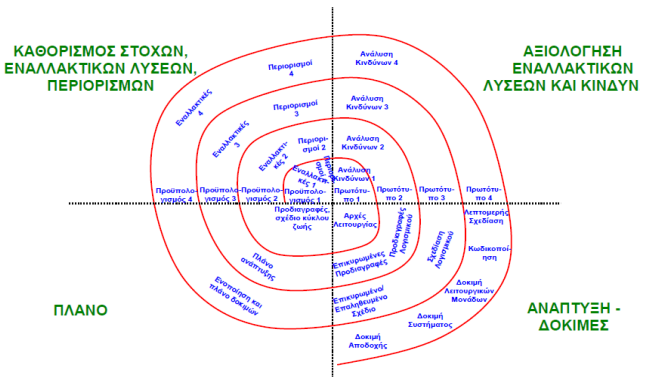
Σπειροειδές μοντέλο ('88)

- **Συνδυασμός ανάπτυξης και διαχείρισης κινδύνων**
  - Νέο βήμα: η αξιολόγηση κινδύνων
- **4 φάσεις επανάληψης**

Σε κάθε φάση γίνονται τα εξής:

- Καθορισμός στόχων → αξιολόγηση κινδύνων → ανάπτυξη και δοκιμές

Σπειροειδές μοντέλο



Βαθμός Παρατηρησιμότητας vs. Μοντέλο Ανάπτυξης

- Καταρράκτη
  - **Καλή. Έγγραφα ανά φάση.**
- Εξελικτικά
  - **Μειωμένη. Συχνές επαναλήψεις.**
- Τυπικοί μετασχηματισμοί
  - **Καλή. Παραγωγή εγγράφων δεδομένη.**
- Σπειροειδές
  - **Καλή. Έγγραφα ανά φάση.**

Η Διεργασία Συγχρονισμού & Σταθεροποίησης της Microsoft (1/2)

- Φιλοσοφία: Αποσύνθεση έργων λογισμικού σε πολλές μικρές ομάδες που εργάζονται παράλληλα, ωστόσο συμπεριφέρονται ως μία μεγάλη ομάδα ανάπτυξης.
- Στόχος: Πλεονεκτήματα γρήγορης και ελεγχόμενης ανάπτυξης γνωστών μοντέλων, και ταυτοχρόνως ελευθερία και αυτονομία στους προγραμματιστές εντός των ομάδων.
- Cusumano and Selby 1995: Synchronize and Stabilize.
- **Φάση Σχεδιασμού**
  - **Vision Statement (στόχοι, προτεραιότητες)**
  - **Specification document (αρχιτεκτονική)**
  - **Πρόγραμμα, προθεσμίες, ομάδες**
- Πάνω από 30% των αρχικών απαιτήσεων τροποποιούνται στην πορεία.

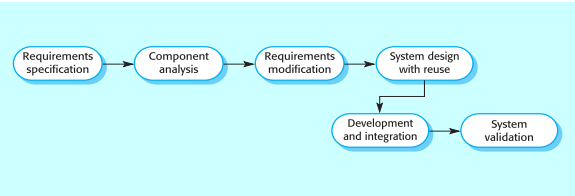
Η Διεργασία Συγχρονισμού & Σταθεροποίησης της Microsoft (2/2)

- Developers και Testers σε κάθε ομάδα.
- Συνεχής έλεγχος, παράλληλη ανάπτυξη εντός ομάδος και μεταξύ
- ομάδων.
- Συγχρονισμός:
  - **Daily synchronization. Σε κάθε ομάδα τα ημερήσια αποτελέσματα εισάγονται σε βάση, ο πηγαίος κώδικας μεταγλωττίζεται και στη συνέχεια ελέγχεται.**
  - **Product synchronization. Καθορισμός οροσήμων (milestones) όπου ελέγχεται αν το προϊόν έχει φθάσει σε σταθερή κατάσταση, μέχρι το στάδιο διανομής του προϊόντος.**
- Συνδυασμός hacker's culture (code and fix) με πειθαρχία και δομή στην ανάπτυξη λογισμικού.

Τεχνολογία Λογισμικού βασισμένη σε συνιστώσες

- **Βασίζεται σε συστηματική επαναχρησιμοποίηση όπου τα συστήματα προκύπτουν από υπάρχουσες συνιστώσες ή από εμπορικά συστήματα COTS (Commercial-off-the-shelf).**
- **Στάδια διεργασίας**
  - Ανάλυση συνιστωσών
  - Τροποποίηση απαιτήσεων
  - Σχεδίαση συστήματος με επαναχρησιμοποίηση
  - Ανάπτυξη και ενοποίηση
- **Αυτή η προσέγγιση έγινε διαδεδομένη από την εξάπλωση των προτύπων συνιστωσών.**

Ανάπτυξη βασισμένη στην επαναχρησιμοποίηση



Extreme programming

- **Μια προσέγγιση η οποία βασίζεται στην ανάπτυξη και παράδοση πολλή μικρών αυξανόμενων τμημάτων λειτουργικότητας.**
- **Βασίζεται στη διαρκή βελτίωση του κώδικα, στη συμμετοχή του χρήστη και στο προγραμματισμό σε ζεύγη.**

(Rational) Unified Process

- Είναι από τις πιο διαδεδομένες μεθόδους ανάπτυξης
- Αναπτύχθηκε από τους Jacobson, Booch και Rumbaugh
- Πλεονεκτήματα
  - **Ώριμη πλέον μέθοδος με καλή υποστήριξη εργαλείων**
  - **Έμφαση στην επανάληψη με καθαρή περιγραφή για το πως μετασχηματίζεται το αντικείμενο μιας δραστηριότητας σε αυτό της επόμενης**
- Μειονεκτήματα
  - **Για μικρότερα έργα προβάλει ένα δυσανάλογα μεγάλο μοντέλο**

(Rational) Unified Process

- Η UP θεωρεί ότι ένα έργο χωρίζεται σε 4 μεγάλες φάσεις:
  - **Την εναρκτήρια φάση**
  - **Τη φάση επεξεργασίας**
  - **Την κατασκευαστική φάση**
  - **Τη φάση μετάβασης**

Η εναρκτήρια φάση

- Σκοπός της είναι να εξετάσει κατά πόσο το έργο μπορεί να γίνει και να καθορίσει κάποιο κοινό στόχο από όλα τα ενδιαφερόμενα μέλη
- Αν έχει προ-αποφασιστεί ότι το έργο θα γίνει τότε η εναρκτήρια φάση είναι πολύ σύντομη, το πολύ μία εβδομάδα ή λιγότερο
- Συνήθως η εναρκτήρια φάση για τα περισσότερα έργα δεν κρατά παρά λίγες εβδομάδες (2-3)
- Τι γίνεται στην εναρκτήρια φάση
  - **Ορίζονται οι στόχοι του έργου (Vision Document). Αυτό είναι ένα έγγραφο που μπορεί να είναι από μια παράγραφο ως μερικές σελίδες και περιέχει μια γενική σύνοψη του έργου**
  - **Παράγεται ένα μοντέλο περιπτώσεων χρήσης**
  - **Γίνεται ανάλυση κόστους/κέρδους και ρίσκου**
  - **Ορίζεται ένα 10%-20% των περιπτώσεων χρήσης**
  - **Μπορεί να κατασκευαστούν και κάποια δείγματα**

Η φάση επεξεργασίας

- Ορίζεται η πλειοψηφία των απαιτήσεων του έργου
- Αντιμετωπίζονται οι πιθανοί κίνδυνοι
- Υλοποιούνται τα βασικά κομμάτια της αρχιτεκτονικής του έργου
- Η φάση επεξεργασίας ολοκληρώνεται συνήθως σε 2 ως 4 επαναλήψεις.
- Κάθε επανάληψη έχει προ-καθορισμένη διάρκεια (συνήθως από 2 ως 4 εβδομάδες)
- Περίπου το 80% των περιπτώσεων χρήσης πρέπει να έχει καταγραφεί σε αυτήν την φάση
- Πρέπει να έχει οριστεί το μοντέλο περιπτώσεων χρήσης
- Πρέπει να έχει οριστεί το μοντέλο σχεδίασης
- Πρέπει να έχει οριστεί το τι θα ελεγχθεί στο τέλος
- Πρέπει να έχει οριστεί η δομή της βάσης δεδομένων
- Πρέπει στο τέλος να έχει παράγει εκτελέσιμο κώδικα με υλοποιημένη τη βασική αρχιτεκτονική

Η κατασκευαστική φάση

- Αφορά την υλοποίηση του έργου
- Έχει διάρκεια συνήθως >= 4 επαναλήψεις
- Κάθε επανάληψη έχει προ-καθορισμένη διάρκεια (συνήθως από 2 ως 4 εβδομάδες)
- Εστιάζει στην υλοποίηση
- Υλοποιεί τις περιπτώσεις χρήσης της προηγούμενης φάσης και αυξάνει τη λειτουργικότητα
- Υλοποιούνται και λεπτομέρειες και όχι μόνο τα βασικά του αρχιτεκτονικού σχεδίου
- Η ανάλυση συνεχίζεται αλλά η σχεδίαση και η υλοποίηση είναι τα βασικά

Η φάση μετάβασης

- Αρχίζει όταν ολοκληρωθεί η υλοποίηση και το λογισμικό παραδοθεί στον πελάτη
- Έχει διάρκεια συνήθως μέχρι 2 επαναλήψεις
- Κάθε επανάληψη έχει προ-καθορισμένη διάρκεια (συνήθως από 2 ως 4 εβδομάδες)
- Σηματοδοτεί την έναρξη χρήσης του συστήματος
- Ο έλεγχος μεταφέρεται περισσότερο στην ομάδα συντήρησης και ελέγχου
- Αφορά κυρίως την ενοποίηση με τα υπάρχοντα συστήματα

Οι επαναλήψεις και η UP

- Η UP μια επαναληπτική διαδικασία ανάπτυξης.
- Η ανάπτυξη του προϊόντος οργανώνεται γύρω από επαναλήψεις συγκεκριμένης διάρκειας (π.χ. 4 εβδομάδων).
- Κάθε επανάληψη είναι ένα mini-project που μοιάζει με τον καταράκτη.
- Κάθε επανάληψη καταλήγει σε ένα εκτελέσιμο, δοκιμασμένο σύστημα που απλά μπορεί να μην υλοποιεί όλες τις απαιτήσεις

Ροές εργασιών

- Υπάρχουν 9 διαφορετικές ροές
  - **Αναπαράσταση του συστήματος**
  - **Καταγραφή απαιτήσεων**
  - **Ανάλυση και ανάπτυξη**
  - **Υλοποίηση**
  - **Έλεγχος**
  - **Παράδοση**
  - **ιαχείριση έργου**
  - **ιαχείριση κώδικα**
  - **Περιβάλλον**

Η σχέση ανάμεσα σε ροές και φάσεις

- Οι ροές εκτελούνται επαναληπτικά κάθε φορά που μπαίνουμε σε μια καινούργια φάση
- Το πόσος χρόνος δαπανάται σε κάθε ροή εξαρτάται από τη φάση που βρισκόμαστε
- Δεν είναι απαραίτητο να εκτελεστούν όλες οι ροές σε κάθε φάση

Computer-aided software engineering (CASE)

- **Computer-aided software engineering (CASE)**
- **Αποτελεί λογισμικό το οποίο υποστηρίζει την ανάπτυξη λογισμικού και την εξέλιξη της διεργασίας**
- **Αυτοματοποίηση δραστηριοτήτων**
  - Γραφικοί συντάκτες και την ανάπτυξη μοντέλων συστημάτων
  - Λεξικά δεδομένων για τη διαχείριση οντοτήτων σχεδίασης
  - Γραφικοί συντάκτες για την δημιουργία διασύνδεσης
  - Debuggers για την εύρεση λαθών
  - Αυτοματοποιημένοι μεταφραστές για τη δημιουργία νέων εκδόσεων προγραμμάτων

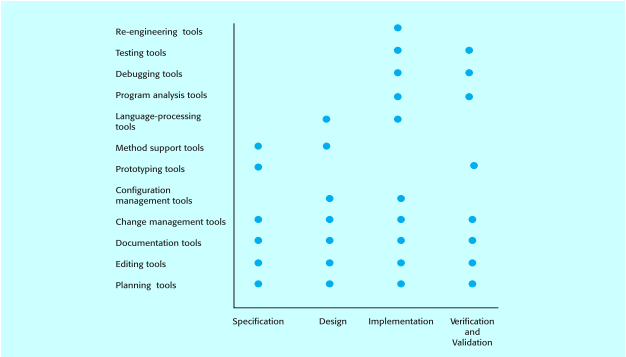
Κατηγοριοποίηση εργαλείων CASE

- **Βασισμένη στις λειτουργίες**
  - Τα εργαλεία κατηγοριοποιούνται ανάλογα με τη λειτουργία τους.
- **Βασισμένη στη διεργασία**
  - Τα εργαλεία κατηγοριοποιούνται ανάλογα με την διεργασία που υποστηρίζουν .
- **Βασισμένη στην ενοποίηση**
  - Τα εργαλεία κατηγοριοποιούνται ανάλογα με την οργάνωση τους σε ενοποιημένες μονάδες.

Λειτουργική Κατηγοριοποίηση εργαλείων CASE

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Κατηγοριοποίηση εργαλείων CASE με βάση τη διεργασία



Ενοποίηση CASE

- **Εργαλεία**
  - Υποστήριξη μεμονωμένων εργασιών διεργασίας, π.χ έλεγχος συνέπεια σχεδίου, μορφοποίηση κειμένου, κ.τ.λ
- **Workbenches**
  - Υποστήριξη ολόκληρης φάσης διεργασίας
  - Support a process phase such as specification or design, Normally include a number of integrated tools.
- **Περιβάλλοντα**
  - Υποστήριξη ολόκληρης διεργασίας.

Εργαλεία, workbenches, περιβάλλοντα