Ανάπτυξη & Σχεδίαση Λογισμικού (ΗΥ420)

Διάλεξη 3:

Μοντέλα Διεργασιών –

Κύκλος Ζωής Λογισμικού

Χρήστος Δ. Αντωνόπουλος
11/3/2009

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών & Δικτύων
Πανεπιστήμιο Θεσσαλίας

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.

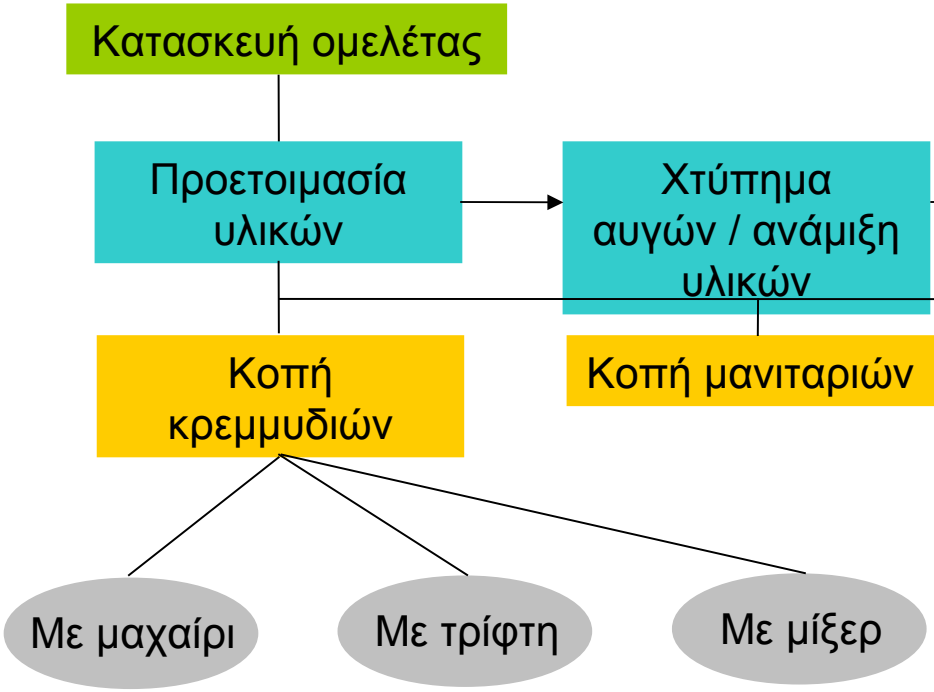
Μοντέλα Διεργασιών Ανάπτυξης Λογισμικού

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

Ορισμός Διεργασίας

- Περιγραφή των κυρίων δραστηριοτήτων.
- Προσδιορισμός των απαιτούμενων πόρων και των περιορισμών τους.
- Υποδιεργασίες.
- Οργάνωση δραστηριοτήτων σε σειρά.
- Περιορισμοί εισόδου / εξόδου κάθε δραστηριότητα.
- Ορισμός στόχων κάθε δραστηριότητας.

Παράδειγμα Διεργασίας



- Πιθανόν να εμπεριέχει υποδιεργασίες.
- Χρονικές ακολουθίες υποδιεργασιών.
- Παράλληλη ή ακολουθιακή εκτέλεση, ανάλογα με διαθεσιμότητα πόρων και εξαρτήσεις μεταξύ υποδιεργασιών.

Χαρακτηριστικά Καλής Διεργασίας Ανάπτυξης Λογισμικού (1/2)

- Κατανόηση
  - Είναι η διεργασία σαφώς καθορισμένη και κατανοητή ?
- Παρακολούθηση
  - Είναι δυνατόν να παρακολουθήσουμε εξωτερικά την πρόοδο της διεργασίας ?
- Υποστήριξη
  - Είναι δυνατόν να υποστηριχθεί η διεργασία με εργαλεία (π.χ. CASE tools) ?
- Αποδοχή
  - Είναι η διεργασία αποδεκτή από αυτούς που συμμετέχουν σε αυτή ?

Χαρακτηριστικά Καλής Διεργασίας Ανάπτυξης Λογισμικού (2/2)

- Αξιοπιστία
  - Ανακαλύπτονται τα λάθη της διαδικασίας πριν μετατραπούν σε λάθη στο προϊόν ?
- Ανθεκτικότητα
  - Είναι δυνατόν να αντεπεξέλθει η διεργασία σε μη αναμενόμενα προβλήματα ?
- Συντήρηση
  - Είναι δυνατόν να εξελιχθεί η διαδικασία για την κάλυψη μεταβαλλόμενων προδιαγραφών και απαιτήσεων ?
- Ταχύτητα
  - Πόσο γρήγορα παράγεται το προϊόν ?

Στάδια Διεργασίας Ανάπτυξης Λογισμικού (Κύκλος Ζωής Λογισμικού)

1. Εξαγωγή & Ανάλυση Απαιτήσεων.
2. Σχεδίαση Συστήματος.
3. Σχεδίαση Προγραμμάτων.
4. Κωδικοποίηση.
5. Δοκιμή Μονάδων και Ενοποίησης.
6. Δοκιμή Συστήματος.
7. Δοκιμή Αποδοχής.
8. Παράδοση, Λειτουργία & Συντήρηση Συστήματος.

Τι;

Πώς;

Έλεγχος

Συντήρηση

Απαιτήσεις

- Τι είναι: Αυτό που θέλουμε να κάνει το σύστημά μας (στόχοι, επιθυμίες, απαιτήσεις)
  - Π.χ. Υπολογισμός και εκτύπωση δεδομένων μισθοδοσίας με βάση την πολιτική της επιχείρησης και το νομικό καθεστώς
- Τι δεν είναι:
  - Αρχιτεκτονική λογισμικού
  - Σχέδιο
  - Διασύνδεση με το χρήστη

Παραδείγματα Απαιτήσεων

- Δηλώσεις:
  - Το σύστημα θα παρέχει δυνατότητα διασύνδεσης με τις βάσεις δεδομένων των ασφαλιστικών ταμείων.
- Σενάρια:
  - Οι εργαζόμενοι που ασθενούν μέχρι 3 ημέρες αποζημιώνονται από το ασφαλιστικό τους ταμείο. Σε περίπτωση που ασθενήσουν παραπάνω και έως 190 μέρες παίρνουν όλο το μισθό τους, ενώ πλέον των 190 ημερών και έως τις 270 παίρνουν το 70% του μισθού. Αναρρωτική άδεια πέραν των 270 ημερών δεν αμείβεται.

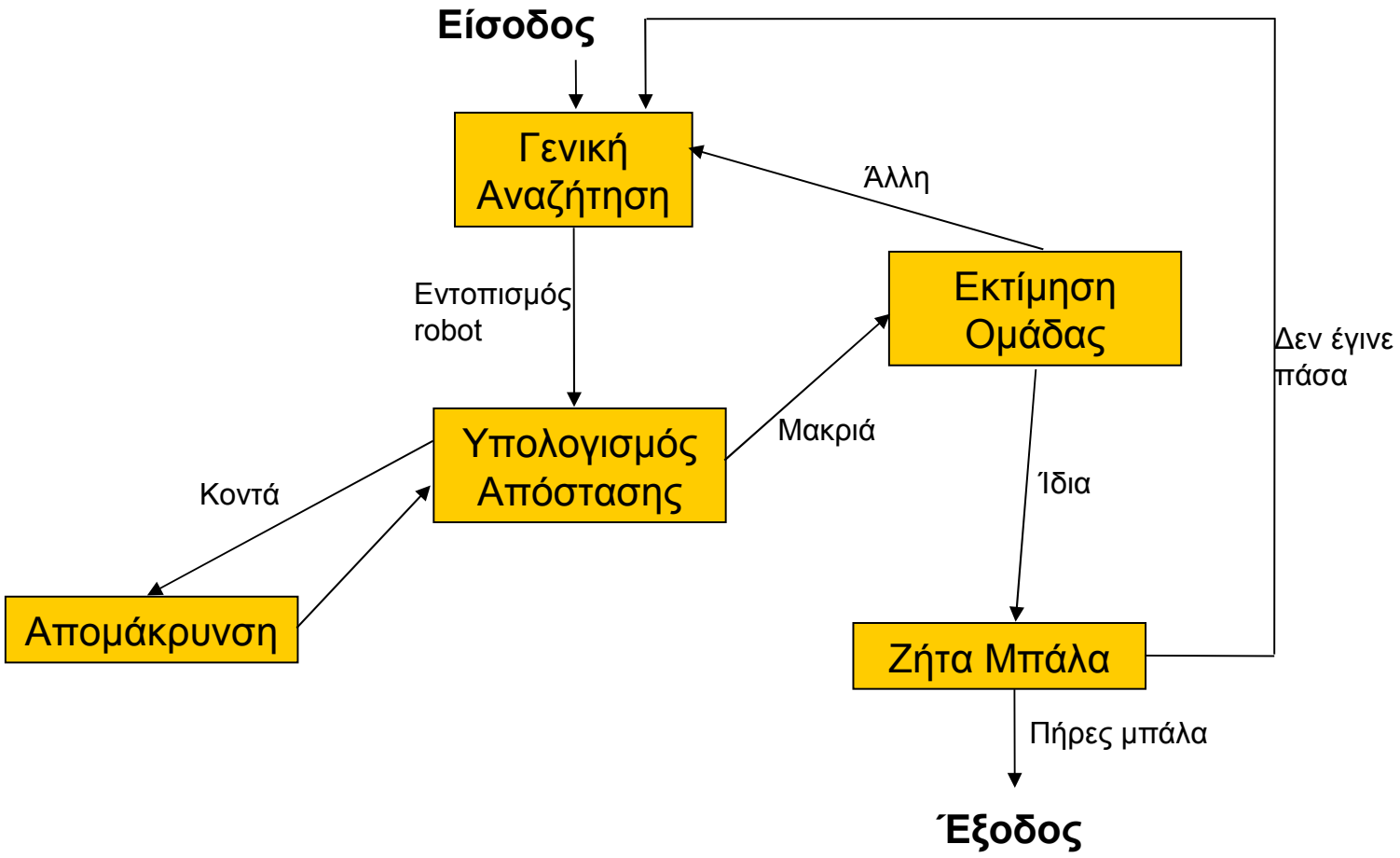
Ανάλυση Απαιτήσεων

- Δόμηση απαιτήσεων.
- Έλεγχος απαιτήσεων.
- Προσδιορισμός εξωτερικών παραγόντων που επηρεάζουν τις απαιτήσεις.
- Προσδιορισμός αλληλεξαρτήσεων μεταξύ απαιτήσεων.
- Γνώση σχετικά με τις απαιτήσεις πριν συνεχίσουμε.

Σχεδίαση

- Πώς θα ικανοποιήσουμε τις προδιαγραφές;
- Εφαρμογή μεθόδων και τεχνικών για τον ορισμό του συστήματος λογισμικού σε ικανοποιητική λεπτομέρεια ώστε να είναι δυνατή η ανάπτυξή του.

Παράδειγμα Σχεδίασης: Πάσα στο Κέντρο σε Ποδόσφαιρο Robots



Υλοποίηση

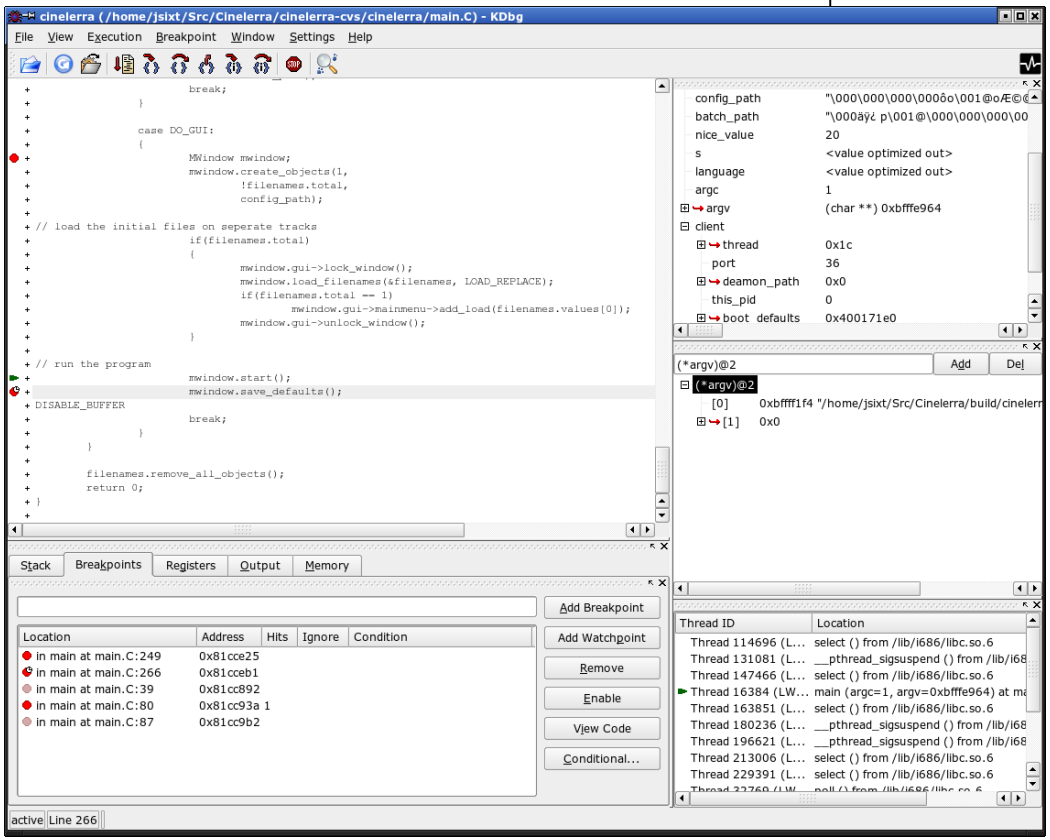
- Μετατροπή της σχεδίασης σε μορφή κατανοητή από τη μηχανή.
  - Λεπτομερής δημιουργία διαγραμμάτων σε CASE tool.
  - Συγγραφή κώδικα σε γλώσσα προγραμματισμού.
  - …

Παράδειγμα Υλοποίησης

```
/*************************************************************************/

static int __clone(unsigned long clone_flags, void (*fn)(struct nth_desc *),
                   struct nth_desc *function_arg, void *child_stack)
{
 struct pt_regs *regs;

 if (!fn)
   {
    printk(KERN_EMERG "Error : __clone called with function pointer =
                       NULL\n");
    return(-1);
   }
 if (!child_stack)
   {
    printk(KERN_EMERG "Error : __clone called with child_stack pointer =
                       NULL\n");
    return(-1);
   }

 regs = ((struct pt_regs *)(current->kernel_stack_page + PAGE_SIZE)) - 1;

 return(do_fork(clone_flags, (unsigned long)child_stack, regs, fn,
                function_arg, 1, 1));
}

/*************************************************************************/
```

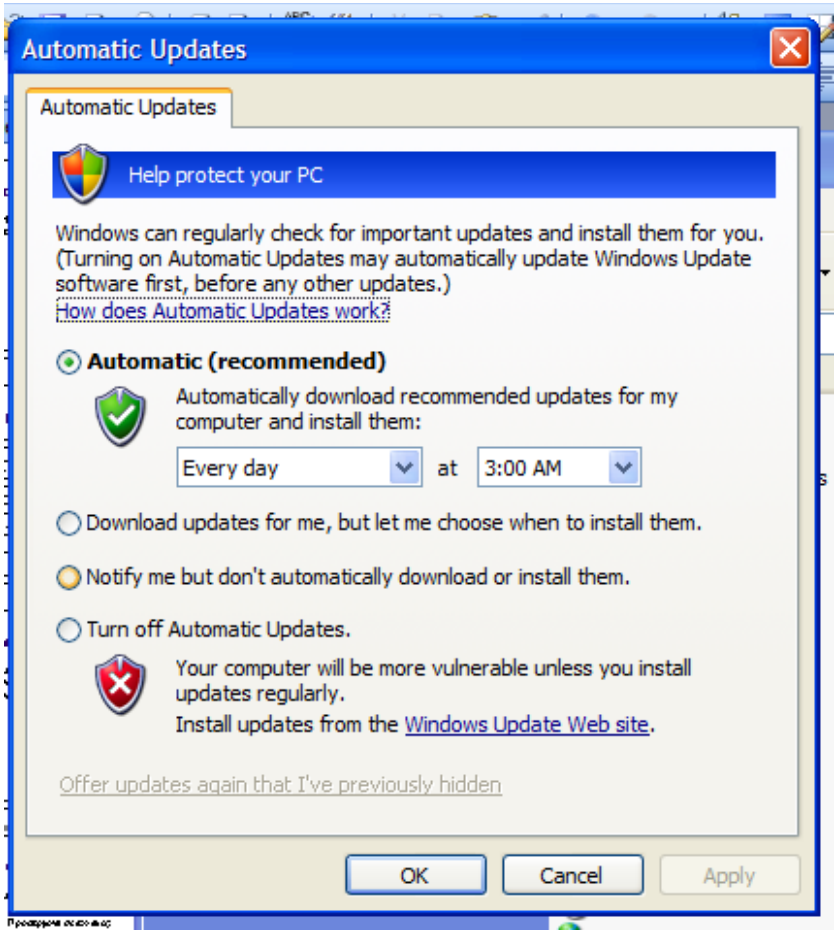
Έλεγχος

- Διαδικασία εκτέλεσης και σχεδιαστικής ανασκόπησης του συστήματος με σκοπό την ανεύρεση λαθών.



Συντήρηση

- Λόγοι:
  - Διόρθωση σφαλμάτων.
  - Προσαρμογή σε αλλαγές:
    - Προδιαγραφών / Απαιτήσεων
    - Περιβάλλοντος
  - Διευκόλυνση μελλοντικής συντήρησης



Δυνατότητα Παρατήρησης - Ελέγχου

- Κάθε φάση στη σχεδίαση λογισμικού καταλήγει στη συγγραφή τεκμηρίωσης.
- Δυνατότητα παρατήρησης της προόδου από τη διοίκηση (σε αντίθεση με άλλα έργα όπου ή πρόοδος είναι πιο εύκολα ορατή)
- Προβλήματα:
  - Η διοίκηση ζητά τεκμηρίωση σε συγκεκριμένες ημερομηνίες (όταν οι σχεδιαστές πνίγονται στη δουλειά !!!).
  - Η προετοιμασία επικύρωση της τεκμηρίωσης απαιτεί επιπρόσθετο χρόνο.
  - Η υλοποίηση του προϊόντος δεν συμβαδίζει, για πρακτικούς λόγους, με το μοντέλο ανάπτυξης.

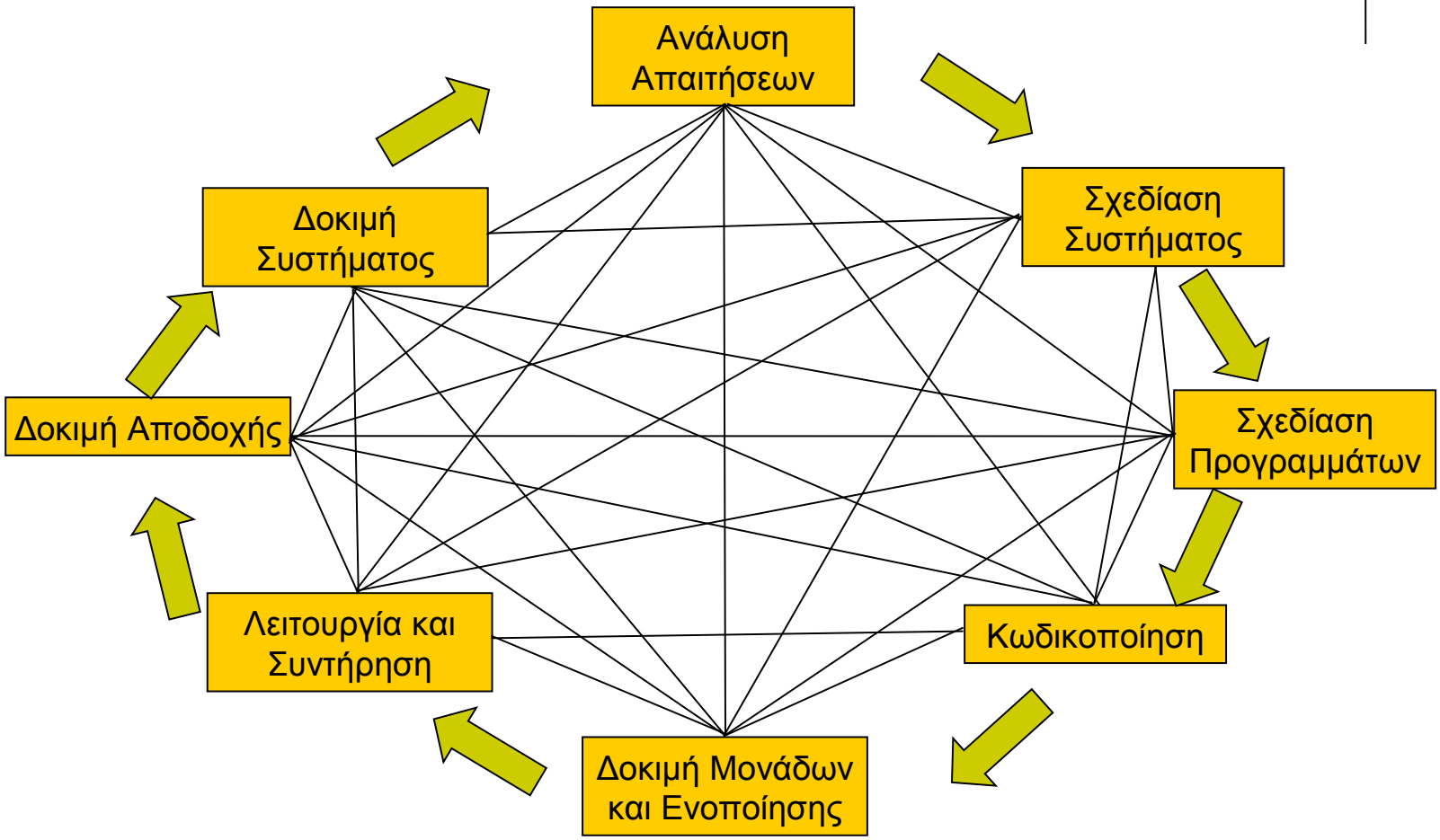
Τεκμηρίωση – Παραγόμενα Έγγραφα

Δραστηριότητα	Παραγόμενα Έγγραφα
Καθορισμός Απαιτήσεων	Requirements Specification
Ανάλυση Απαιτήσεων	Feasibility Study
Σχεδίαση Συστήματος	Functional Specification
Σχεδίαση Λογισμικού	Design Specification
Κωδικοποίηση	Program code
Έλεγχος Μονάδων	Unit test report
Έλεγχος Τμημάτων	Module test report
Έλεγχος Συστήματος	System test report
Παράδοση Προϊόντος	User Guide – Refer. Manual

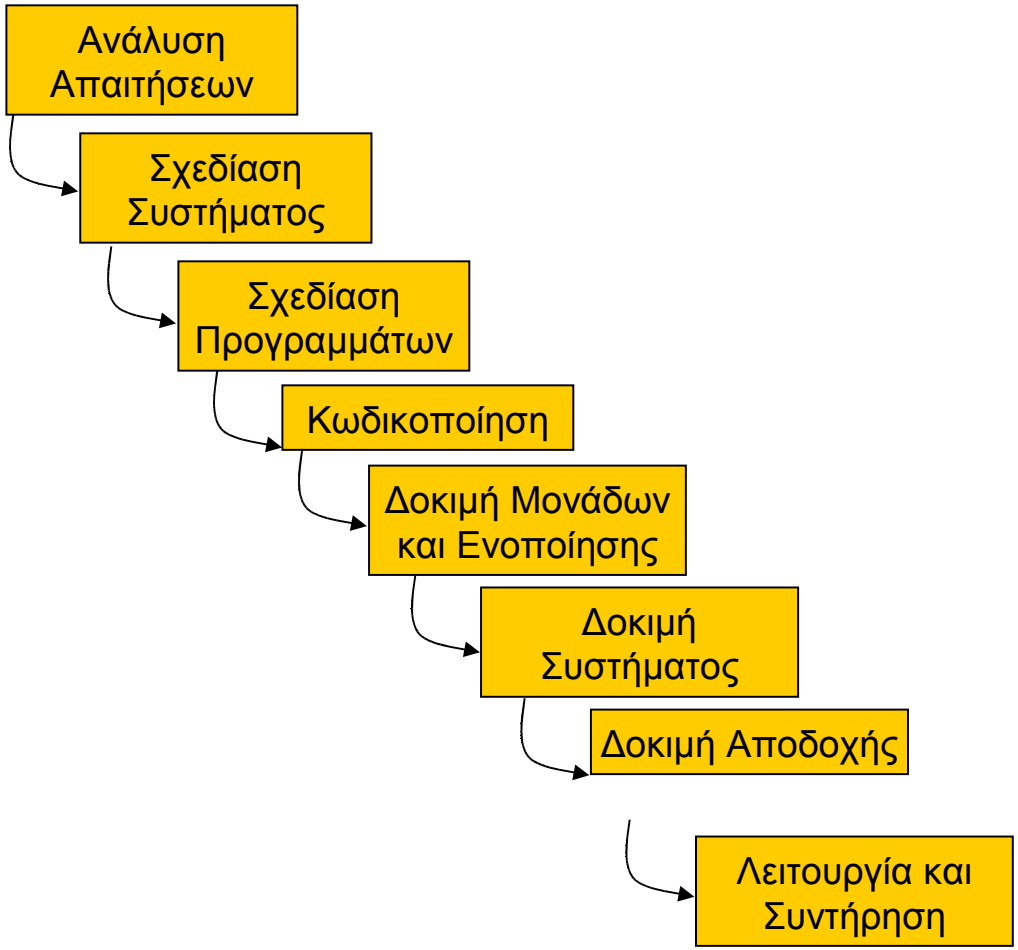
Μοντέλα Διεργασιών Λογισμικού

- Προσδιορίζουν τον τρόπο με τον οποίο πρέπει να προχωρήσει η ανάπτυξη του λογισμικού.
- Περιγράφουν τον τρόπο με τον οποίο γίνεται η ανάπτυξη στην πραγματικότητα.
- Κατανόηση δραστηριοτήτων, πόρων και περιορισμών.
- Εντοπισμός ασυνεπειών, πλεονασμών και παραλείψεων.
- Είσοδος: απαιτήσεις συστήματος,
- Έξοδος: τελικό προϊόν.

Η Διαδικασία Ανάπτυξης Λογισμικού στην Πραγματικότητα



Γραμμικό Μοντέλο (Καταρράκτη)



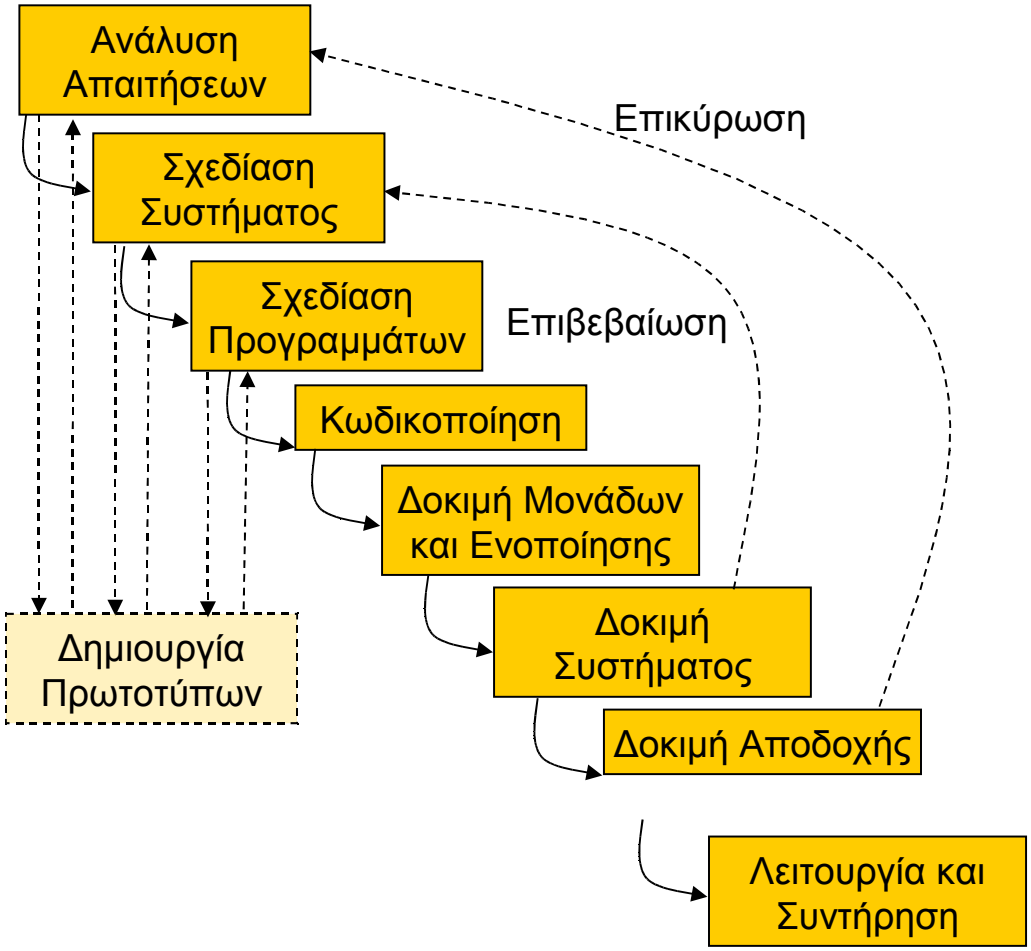
Χαρακτηριστικά Μοντέλου Καταρράκτη

- Το μοντέλο καταρράκτη δεν κάνει υποθέσεις για τη μεθοδολογία και τους συμβολισμούς σε κάθε στάδιο.
- Βασικές αρχές μοντέλου:
  - Ακολουθία βημάτων.
  - Κάθε βήμα είναι σαφώς καθορισμένο.
  - 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.

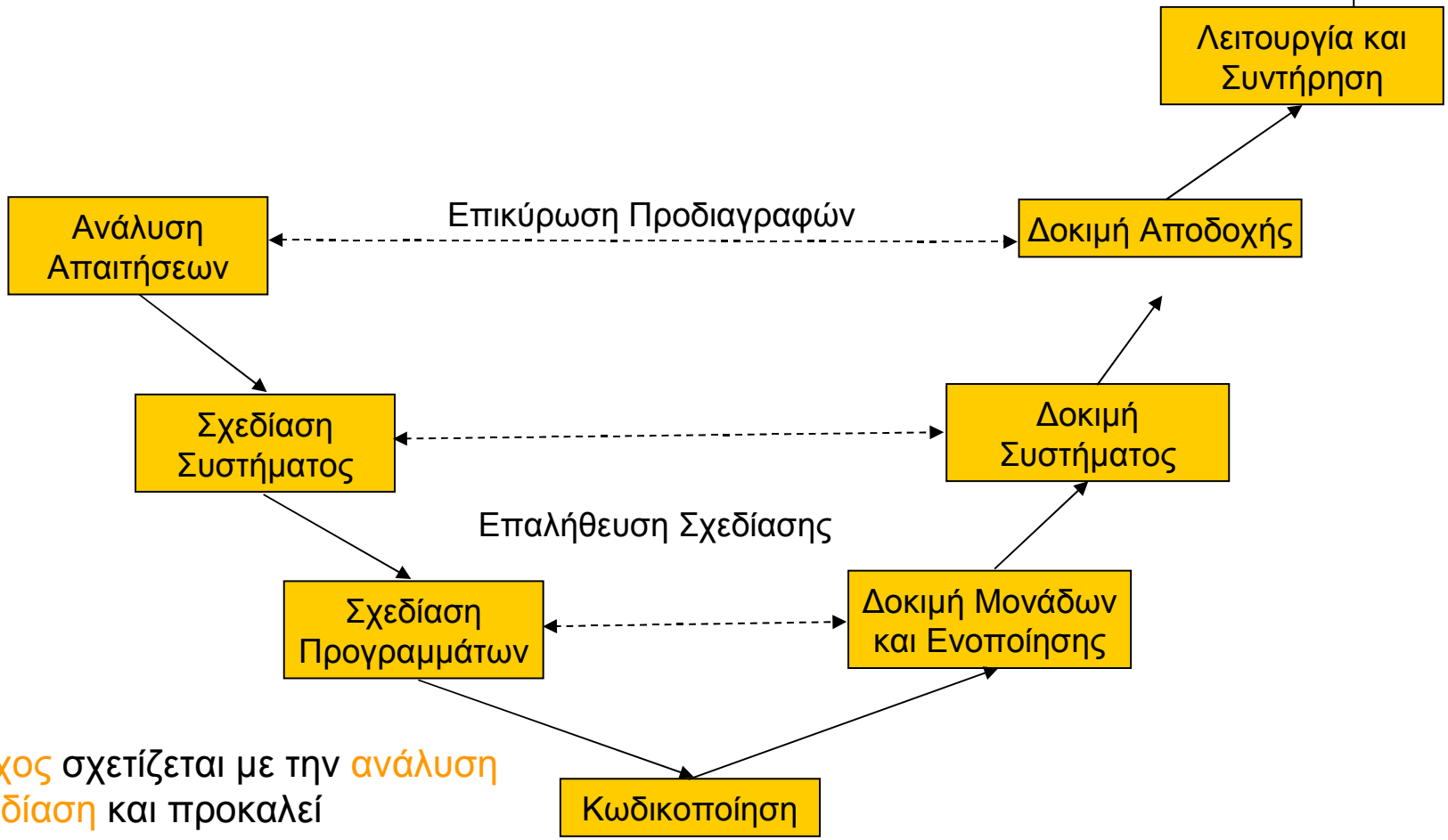
Πλεονεκτήματα / Μειονεκτήματα Μοντέλου Καταρράκτη

- Πλεονεκτήματα
  - Καλός διαχωρισμός του έργου σε απλούστερες φάσεις.
  - Κάθε φάση παράγει ένα σαφώς καθορισμένο παραδοτέο.
  - Manager + Waterfall model =
- 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 !!!

Γραμμικό Μοντέλο με Δημιουργία Πρωτοτύπων

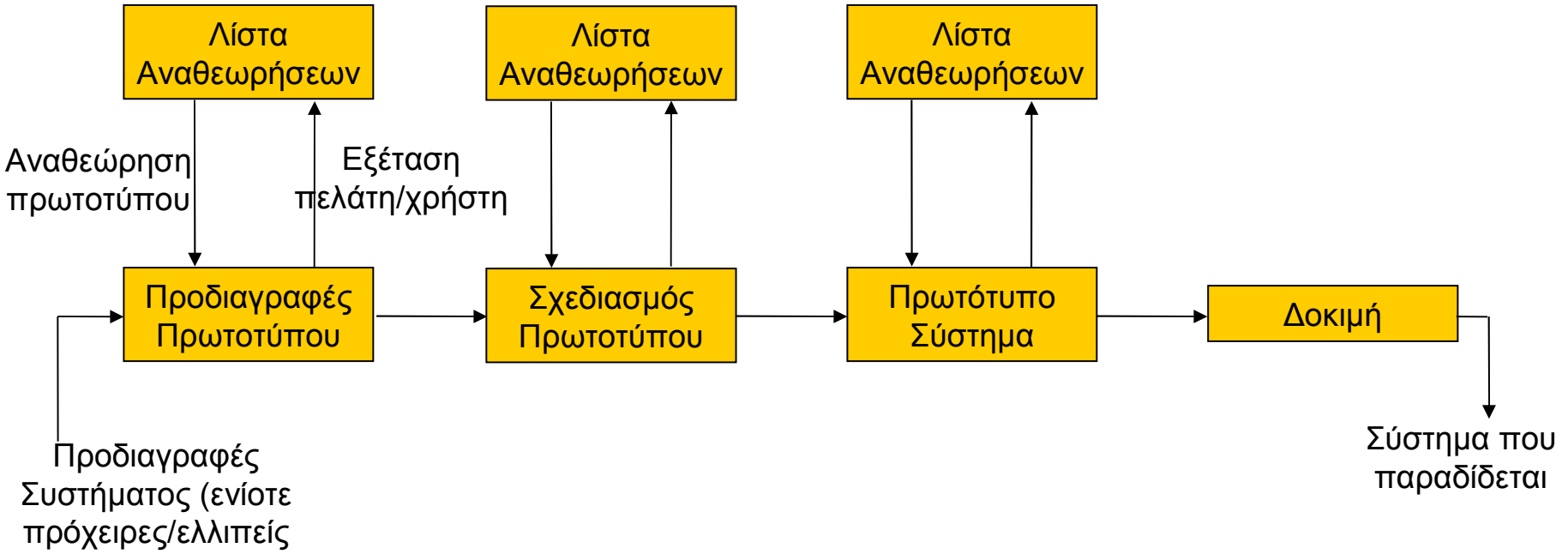


Μοντέλο V



Ο έλεγχος σχετίζεται με την ανάλυση και σχεδίαση και προκαλεί επαναλήψεις.

Μοντέλο Δημιουργίας Πρωτοτύπου



Πλεονεκτήματα Μοντέλου Δημιουργίας Πρωτοτύπου

- 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.

Μειονεκτήματα Μοντέλου Δημιουργίας Πρωτοτύπου

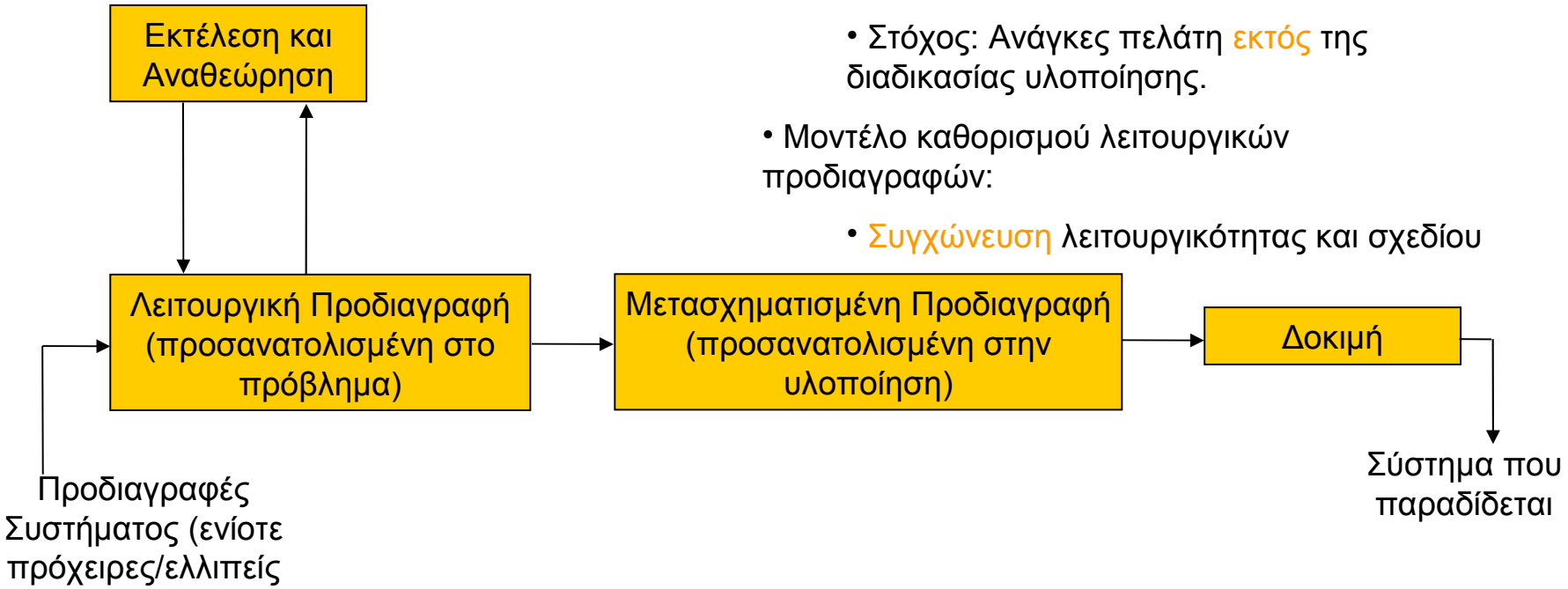
- Έλλειψη παρατήρησης στη διαδικασία (αδυναμία πρόβλεψης επαναλήψεων).
- Συστήματα λιτά δομημένα λόγω συχνών αλλαγών.
- Εστίαση στη λειτουργικότητα – Λιγότερη έμφαση στις μη λειτουργικές απαιτήσεις.
- Ειδικές ικανότητες μπορεί να απαιτηθούν (π.χ. γλώσσες για rapid prototyping).
- Υπερβολικός ενθουσιασμός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

Εφαρμογές Μοντέλου Δημιουργίας Πρωτοτύπου

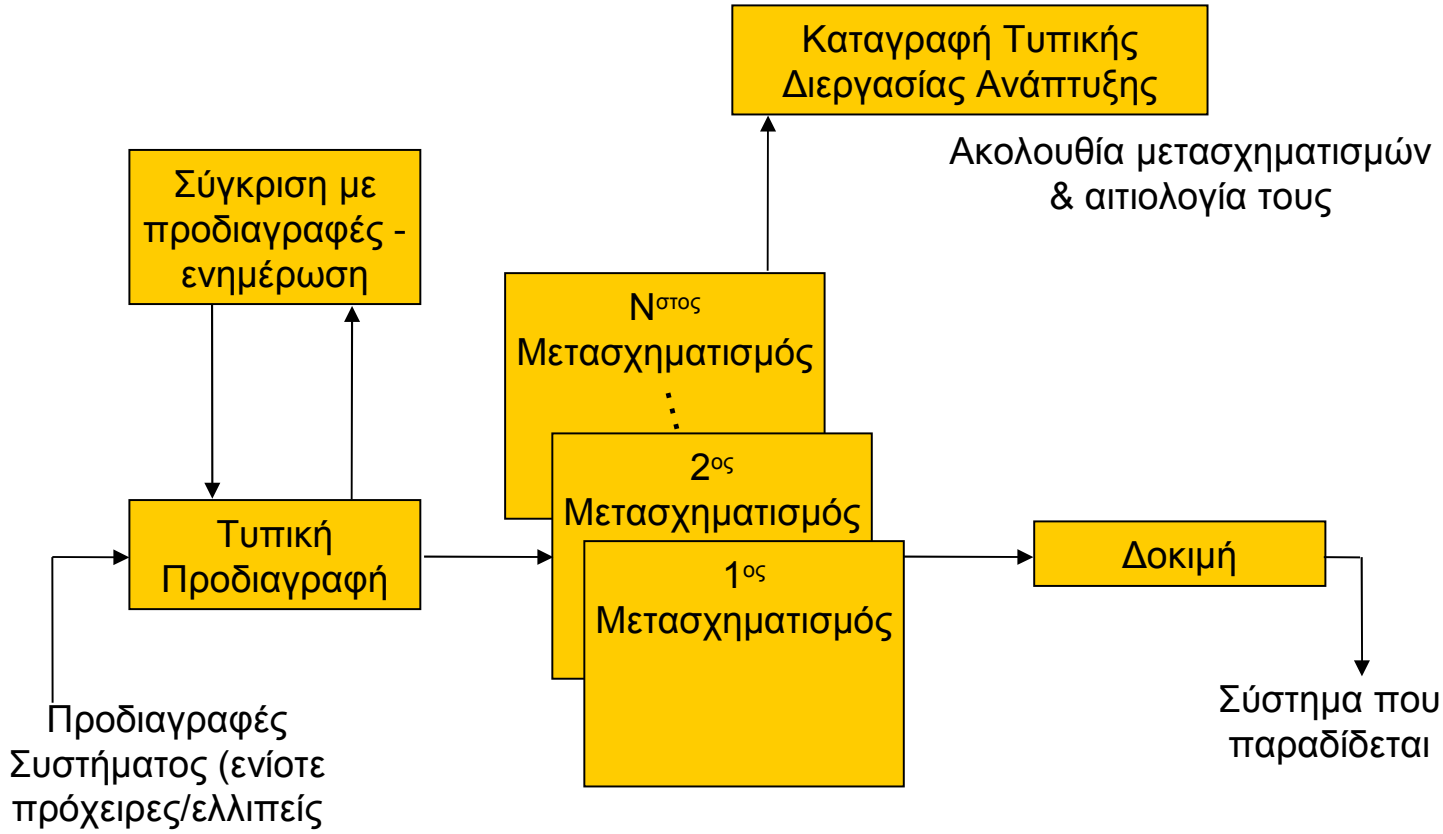
- Εφαρμογή
  - Για μικρά ή μεσαίου μεγέθους συστήματα (το αρχικό πρωτότυπο πλησιάζει το τελικό προϊόν).
  - Για τμήματα μεγάλων συστημάτων (π.χ.. η διεπαφή χρήστη ενός συστήματος εναέριας κυκλοφορίας).
  - Για συστήματα με μικρό χρόνο ζωής .
  - Για συστήματα όπου υπάρχει αδυναμία έκφρασης των απαιτήσεων από πριν.
- Ακατάλληλο για:
  - Λογισμικό ενσωματωμένων συστημάτων.
  - Λογισμικό πραγματικού χρόνου.
  - Επιστημονικό λογισμικό.

Μοντέλο Καθορισμού Λειτουργικών Προδιαγραφών

- Γραμμικό μοντέλο:
  - Λειτουργικότητα διαχωρίζεται από το σχέδιο.
  - Στόχος: Ανάγκες πελάτη εκτός της διαδικασίας υλοποίησης.
- Μοντέλο καθορισμού λειτουργικών προδιαγραφών:
  - Συγχώνευση λειτουργικότητας και σχεδίου



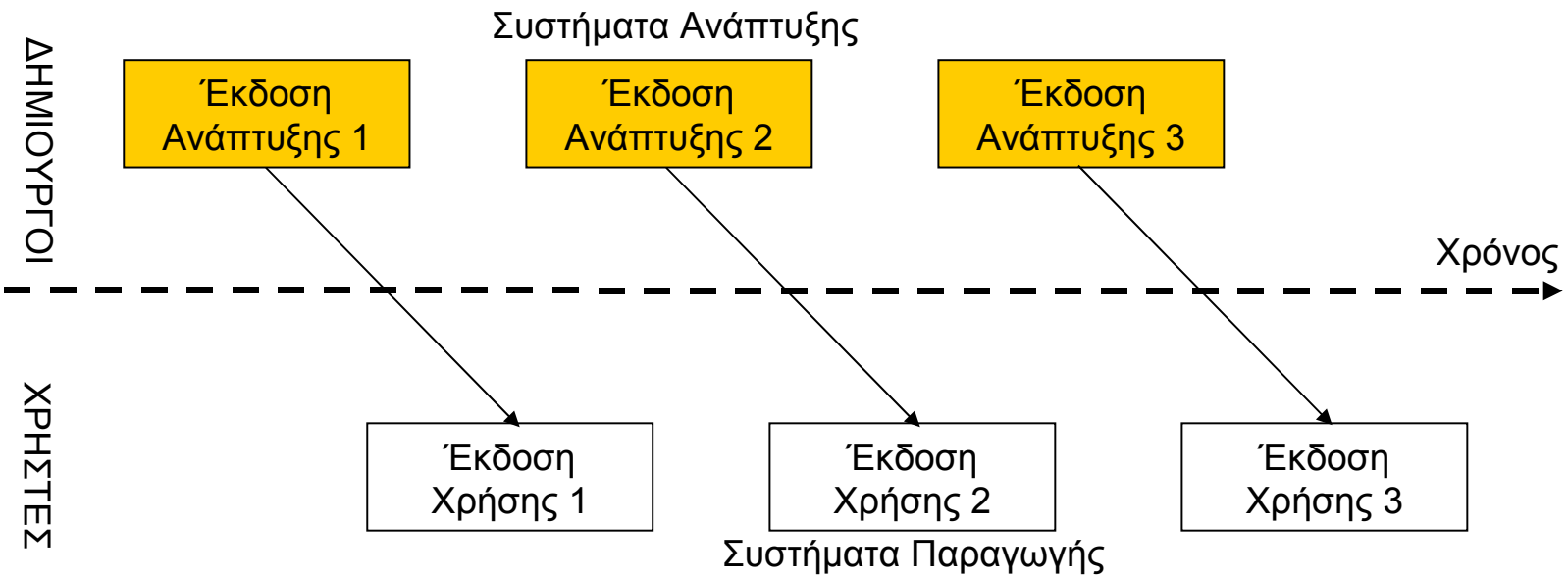
Μοντέλο Τυπικών Μετασχηματισμών



Χαρακτηριστικά Μοντέλων Τυπικών Μετασχηματισμών

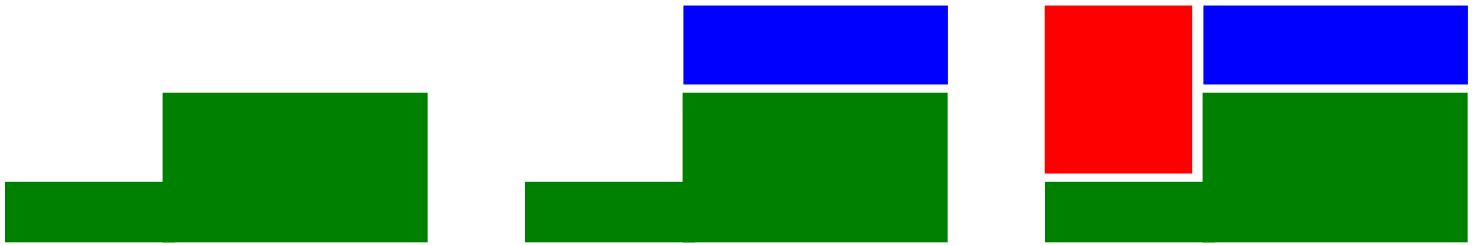
- Οι προδιαγραφές συντάσσονται με αυστηρά μαθηματικό τρόπο
- Η αρχική μαθηματική περιγραφή μετασχηματίζεται σταδιακά στο τελικό προϊόν
- Διατήρηση της ορθότητας με χρήση μαθηματικών τεχνικών απόδειξης
- Πλεονεκτήματα
  - Σαφήνεια – Ακρίβεια
  - Μεγάλη δυνατότητα αυτοματοποίησης
- Μειονεκτήματα
  - Η χρήση μαθηματικών συμβολισμών δυσχεραίνει την επικοινωνία με τον πελάτη
  - Εκπαίδευση του προσωπικού στη χρήση => χρόνος, χρήμα

Μοντέλο Ανάπτυξης σε Φάσεις

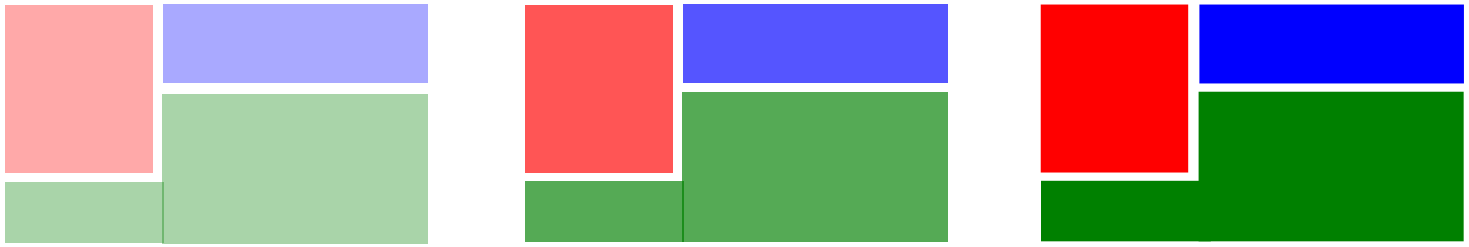


Αυξητική vs. Επαναληπτική Ανάπτυξη

Αυξητική
Ανάπτυξη



Επαναληπτική
Ανάπτυξη



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.

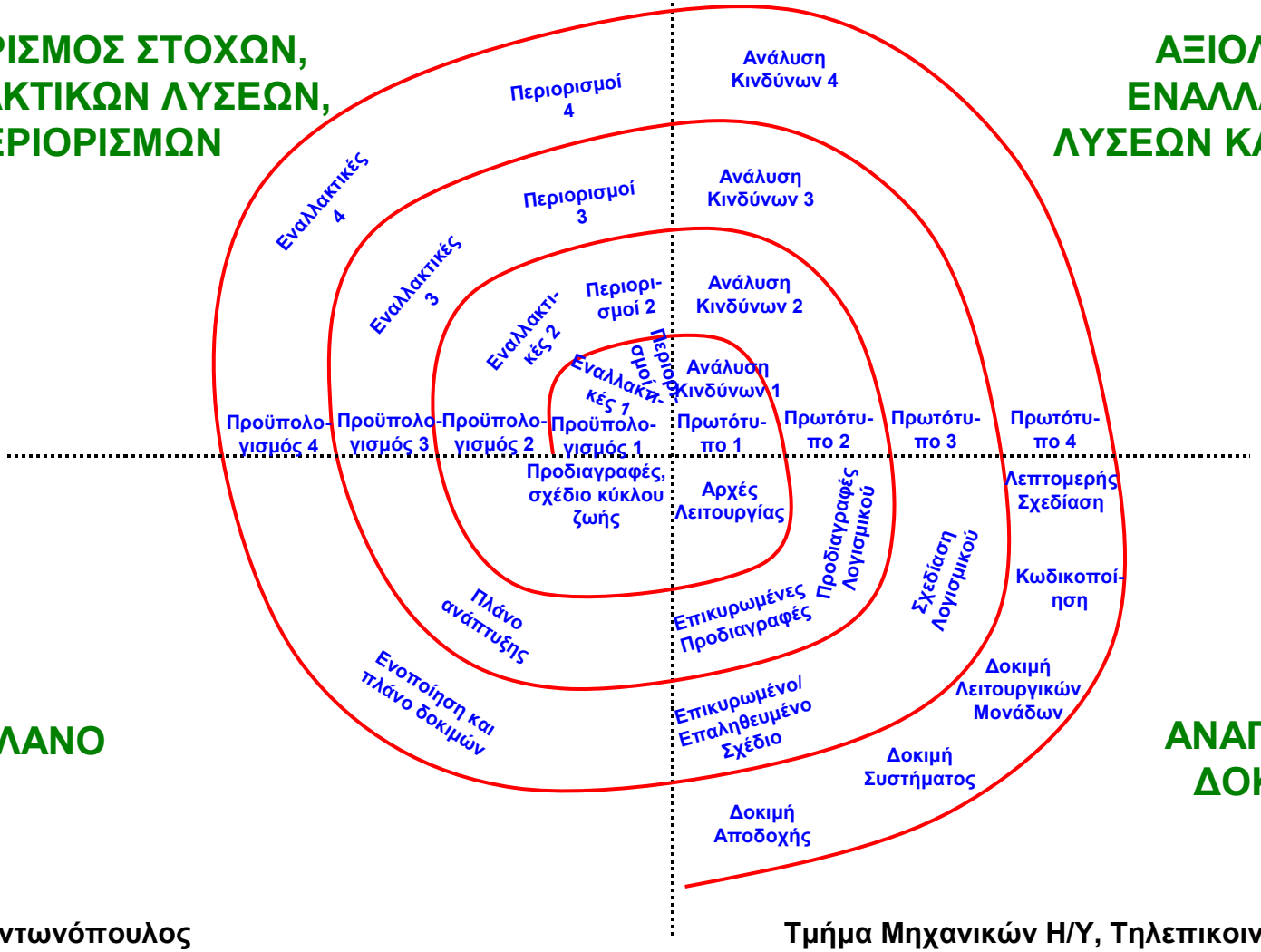
Παραδείγματα Κινδύνου

- 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.

Διαχείριση Κινδύνων – Μοντέλα Ανάπτυξης

- Καταρράκτη
  - Υψηλό ρίσκο για νέα συστήματα λόγω προβλημάτων στις προδιαγραφές και στη σχεδίαση.
  - Χαμηλό ρίσκο για πλήρως κατανοητά συστήματα που αναπτύσσονται με γνωστή τεχνολογία.
- Δημιουργία πρωτοτύπων
  - Χαμηλό ρίσκο για νέες εφαρμογές λόγω του γεγονότος ότι οι προδιαγραφές και η υλοποίηση εξετάζονται ανά βήμα.
  - Υψηλό ρίσκο λόγω έλλειψης παρατήρησης.
- Τυπικοί Μετασχηματισμοί
  - Υψηλό ρίσκο λόγω απαιτήσεων για προηγμένη τεχνολογία και εξειδίκευση του προσωπικού.

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.

Όταν λέμε Agile;

- Agile
  - Ευκίνητο
  - Ταχύ
  - Εύκαμπτο
  - Εύστροφο
  - Ευέλικτο
    - Επιδέξιο στο χειρισμό μεταβολών / αλλαγών

Βασικά Σημεία

- Γιατί εμφανίστηκε το Agile Development;
- Τι σημαίνει να είναι κάτι Agile?
- Ποιες οι βασικές αρχές του Agile Development?

- Βιβλιογραφία:
  - An Introduction to Agile Methods (Hayes, Andrews)
  - Agile Software Development Ecosystems (Highsmith)
  - Agile Alliance website: http://www.agilealliance.org
  - The New Methodology (Fowler)

Από το Τίποτα στο Μνημειώδες

- Συνήθως η ανάπτυξη λογισμικού γίνεται χαοτικά
  - Γράψε κώδικα σαν τρελός (δηλαδή γράψε και φτιάξε)
  - Κανένα σχέδιο
  - Δουλεύει καλά αρχικά, αλλά δεν μπορεί να ακολουθήσει όταν μεγαλώνει το μέγεθος των συστημάτων
    - Δύσκολο να προστεθούν νέα χαρακτηριστικά
    - Δύσκολο να βρεθούν και να διορθωθούν σφάλματα
    - Μεγάλες φάσεις ελέγχου (αν υπάρχουν...)
- Η μέθοδοι τεχνολογίας λογισμικού εισήγαγαν πειθαρχία
  - Βελτίωση προβλεψιμότητας και αποδοτικότητας
  - Όμως θεωρούνται γραφειοκρατικές, δεν είναι δημοφιλείς
  - Μειώνουν το ρυθμό ανάπτυξης

Από το Μνημειώδες στο Agile

- Αναπτύχθηκε ως αντίδραση στις μεθόδους τεχνολογίας λογισμικού
  - Συμβιβασμός στο “βάρος” της διαδικασίας
  - Προσαρμογή αντί πρόβλεψης
  - Προσανατολισμένη στους ανθρώπους αντί στη διαδικασία
- Γιατί το “μνημειώδες” & “ηρωϊκό” δε δουλεύει για το λογισμικό;
  - Κόστος σχεδίασης / κόστος υλοποίησης ενός έργου
  - Πιθανότητα αλλαγής και επιπτώσεις της
  - Οι μέθοδοι αυστηρά δομημένης ανάπτυξης λογισμικού δεν έχουν ορισμάσει πλήρως
  - Το μνημειώδες και ηρωϊκό συχνά δε δουλεύει για τους μηχανικούς!

Το “μανιφέστο” του Agile Development

“We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over Processes and Tools.
- Working software over Comprehensive documentation.
- Customer collaboration over Contract negotiation.
- Responding to change over Following a plan.

That is, while there is value in the items on the right, we value the items on the left more.”

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert Martin, Stephen J. Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas **(2001)**

Βασικές Αξίες

- Επανάληψη!
- Αυξητικές προσθήκες!
  - Φτιάξε το απλούστερο δυνατό που δουλεύει, και άσε τα επόμενα επαναληπτικά βήματα να χτίσουν επάνω του
- Καθοδήγηση από τον πελάτη!
  - Φρόντισε να αναμειχθεί ο πελάτης και δείξε του την πρόοδο σε κάθε επανάληψη
- Γενναία, αποδοτική επικοινωνία!
  - Η ανοιχτή και ειλικρινής επικοινωνία μπορεί να είναι δύσκολη, ειδικά μεταξύ διαφορετικών ιεραρχικών επιπέδων
- Υψηλή ποιότητα!
  - Μέσω της συνεχούς βελτίωσης, όχι του υπερ-σχεδιασμού
- Ενωμένη ομάδα!

Χαρακτηριστικά

- Περιορισμός του αριθμού των συμμετεχόντων σε κάθε ομάδα ανάπτυξης
- Μικροί και διακριτοί κύκλοι ανάπτυξης με σαφώς καθορισμένα και απόλυτα λειτουργικά αποτελέσματα
- Διαρκής έλεγχος του λογισμικού
- Μεγαλύτερη δυνατή τμηματοποίηση του έργου σε επίπεδο διαδικασίας ανάπτυξης.
- Εξάλειψη όλων των περιττών ενεργειών της διαδικασίας ανάπτυξης

Χαρακτηριστικά

- Ανάπτυξη με βάση τη συνεργασία και την επικοινωνία
- Ανάπτυξη μιας πρώτης έκδοσης σε διάστημα μερικών εβδομάδων για τη λήψη άμεσης και γρήγορης ανατροφοδότησης
- Εύρεση και χρήση των απλούστερων λύσεων
- Μείωση της απαραίτητης τεκμηρίωσης του έργου
- Συνεχής επαφή, διαθεσιμότητα και συμμετοχή του πελάτη

Έμφαση σε:

- Ταλέντο & ικανότητες (λιγότεροι, καλύτεροι στην ομάδα ανάπτυξης)
- Προσιτότητα (ομάδα ανάπτυξης – ομάδα ανάπτυξης / χρήστες)
- Επικοινωνία (ηθικό, καθημερινή αναμέτρηση)
- Just-in-time ανάλυση απαιτήσεων & σχεδίαση
- Συχνές παραδόσεις (αυξητική ανάπτυξη)
- Λιγότερο “χαρτί”, περισσότερη προσωπική, προφορική επικοινωνία
- Εργαλεία
- Ποιότητα δουλειάς
- Διαφορετικές στρατηγικές για διαφορετικά έργα

Πότε μια Μεθοδολογία Θεωρείται Agile?

- Προϋποτίθεται:
  - η σταδιακή ανάπτυξη
  - η συνεργασία
  - η ευκολία στην κατανόηση και εφαρμογή της (καλά τεκμηριωμένη)
  - η προσαρμοστικότητα

Υφιστάμενες Agile Μεθοδολογίες

- Extreme Programming (Beck 1999)
- Scrum (Schwaber 1995, Schwaber and Beedle 2002)
- Crystal family of methodologies (Cockburn 2002)
- Feature Driven Development (Palmer and Felsing 2002)
- Rational Unified Process (Kruchten 1996, Kruchten 2000)
- Dynamic Systems Development Method (Stapleton 1997)
- Adaptive Software Development (Highsmith 2000)
- Open Source Software Development

Μύθοι του Agile Development

- Είναι νέο και μη δοκιμασμένο
- Δεν καταγράφονται οι απαιτήσεις
- Δεν υπάρχει αρχιτέκτονας / αρχιτεκτονική => chaos
- Δεν υπάρχει φάση σχεδίασης
- Τα ρίσκα αγνοούνται
- Η ομάδα ανάπτυξης και οι πελάτες θα το μισήσουν

Αλήθειες και Ψέματα...

- Το Agile SD (Software Development) είναι ζαβολιά...
- Το Agile SD απαιτεί τους καλύτερους στην ομάδα ανάπτυξης
- Το Agile SD είναι hacking
- Το Agile SD δε “δουλεύει” για όλα τα έργα

Το Agile SD είναι “ζαβολιά”

- Προσέλαβε τους καλύτερους
- Βάλε τους μαζί για να αλληλοβοηθούνται
- Φέρε τους κοντά στους πελάτες και τους χρήστες
  - Φρόντισε να υπάρχει ταχεία ανάδραση στις αποφάσεις
  - Επέτρεψέ τους να βρίσκουν εύκολους και γρήγορους τρόπους να τεκμηριώνουν τη δουλειά τους
- Αφαίρεσε γραφειοκρατεία

- Στο τέλος όλα αυτά ακούγονται σαν καλή ιδέα…
  - Είναι η “καρδιά” του agile development

Το Agile “δουλεύει” μόνο αν έχεις τους καλύτερους...

- Κάθε έργο απαιτεί τουλάχιστον έναν ικανό και έμπειρο ηγέτη (κρίσιμος παράγοντας επιτυχίας).
- Για κάθε έμπειρο & ικανό μέλος, μπορούν να μπουν στην ομάδα 4-5 μέτριοι / εκπαιδευόμενοι.

- Με αυτό το μείγμα έχει αποδειχθεί ότι το agile development “δουλεύει” καλά.

Το Agile SD είναι hacking

- Hackers: “...αφιερώνουν όλον τους τον χρόνο στη συγγραφή κώδικα”
- Agilists: ...ελέγχουν σύμφωνα με τις προτεραιότητες του έργου, επανελέγχουν συχνά τα αποτελέσματα με τους χρήστες

- Hackers: “...μιλούν μεταξύ τους όταν κολλήσουν”
- Agilists: ...μιλούν μεταξύ τους και με τους χρήστες στα πλαίσια της διαδικασίας ανάπτυξης

- Hackers: “...αποφεύγουν τον προγραμματισμό (planning)”
- Agilists: ...προγραμματίζουν τακτικά

- Hackers: “...ο manager είναι το θηρίο...”
- Agilists: ...περιμένουν ο manager να δώσει τις προτεραιότητες & να συμμετέχει στις αναπροσαρμογές του έργου

Είναι κατάλληλο για το έργο μου;

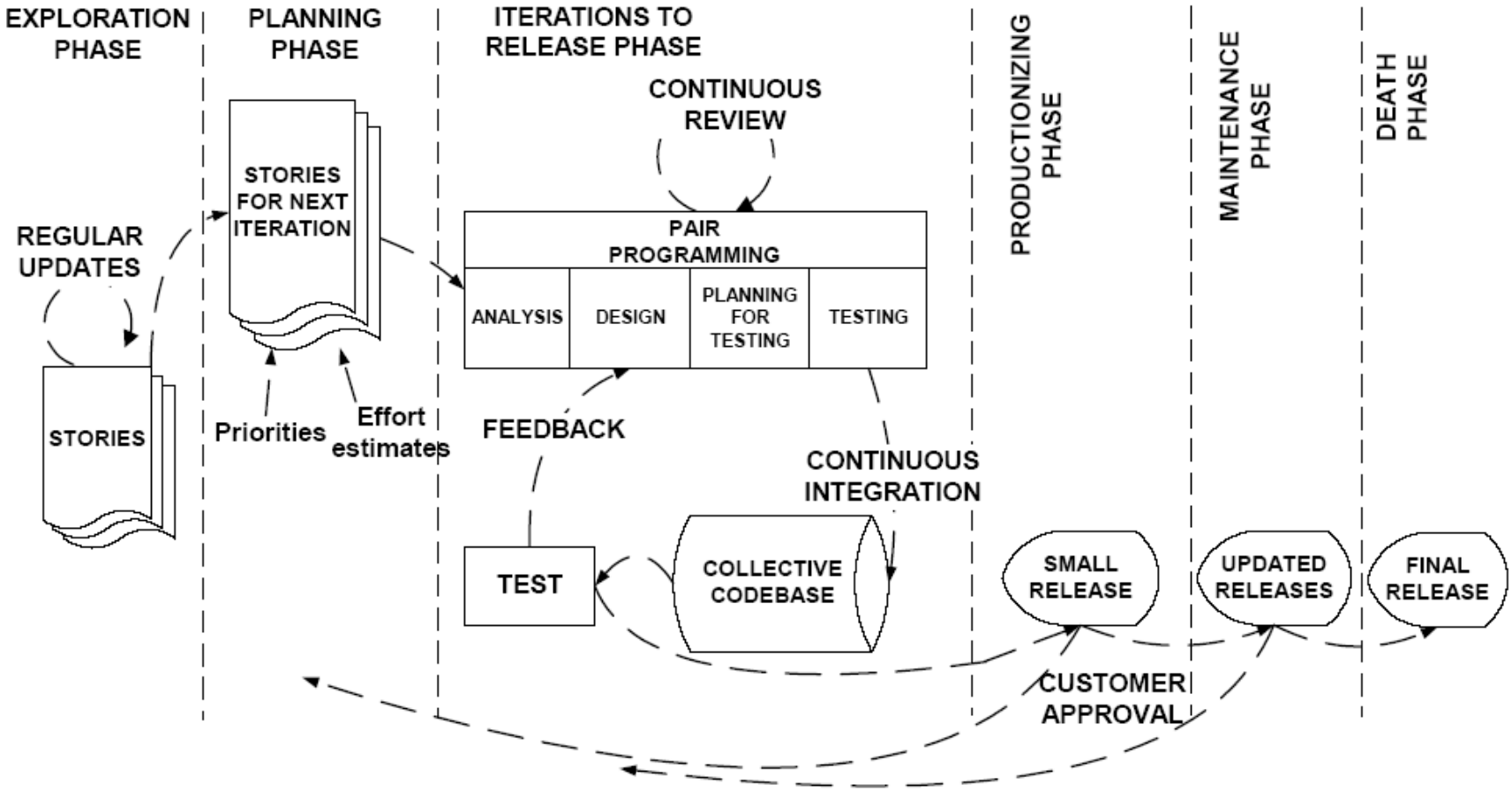
- Όχι πάντα...
  - Δεν εγγυάται επιτυχία
- Πρέπει να είναι προσαρμοστικό!
  - Βρες τα αδύνατα σημεία και διόρθωσε τη διαδικασία
- Μπορεί να είναι δύσκολο να εφαρμοστεί σε γεωγραφικά διασκορπισμένες ομάδες
- Δεν μπορούν όλες οι ομάδες να μιλήσουν ανοιχτά

- *Πολλά έργα οφελήθηκαν από το agile development!*

Extreme Programming (XP)

- Αναπτύχθηκε για να επιλύσει προβλήματα τα οποία προκλήθηκαν από τους μεγάλους κύκλους ανάπτυξης των παραδοσιακών μοντέλων ανάπτυξης λογισμικού (Beck 1999)
- Ξεκίνησε ως ένας πρακτικός τρόπος υλοποίησης ενός έργου λογισμικού
- Χρησιμοποιεί πρακτικές οι οποίες είχαν προκύψει μέσα από την πράξη (Beck 1999)
- Οι πρακτικές οι οποίες χρησιμοποιούνται δεν αποτελούν καινοτομία

XP: Κύκλος ζωής



XP: Αξιολόγηση

- Πλεονεκτήματα
  - Χρήση των user stories τα οποία δημιουργούνται από τον πελάτη
  - Καθορισμός προτεραιοτήτων στα user stories
  - Εκτίμηση της προσπάθειας
  - Κυρίαρχος ρόλος των ελέγχων του συστήματος
  - Συμμετοχή του πελάτη στην καταγραφή των απαιτήσεων και τη συγγραφή των λειτουργικών ελέγχων
  - Ο ρόλος του Tracker
  - Απλές πλην απαραίτητες πρακτικές (testing, simple design, refactoring, pair programming)
  - Σαφήνεια της διαδικασίας και των πρακτικών, τεχνικών

XP: Αξιολόγηση

- Μειονεκτήματα
  - Έλλειψη κατευθύνσεων διαχείρισης
  - Μεγάλος αριθμός τεχνικών

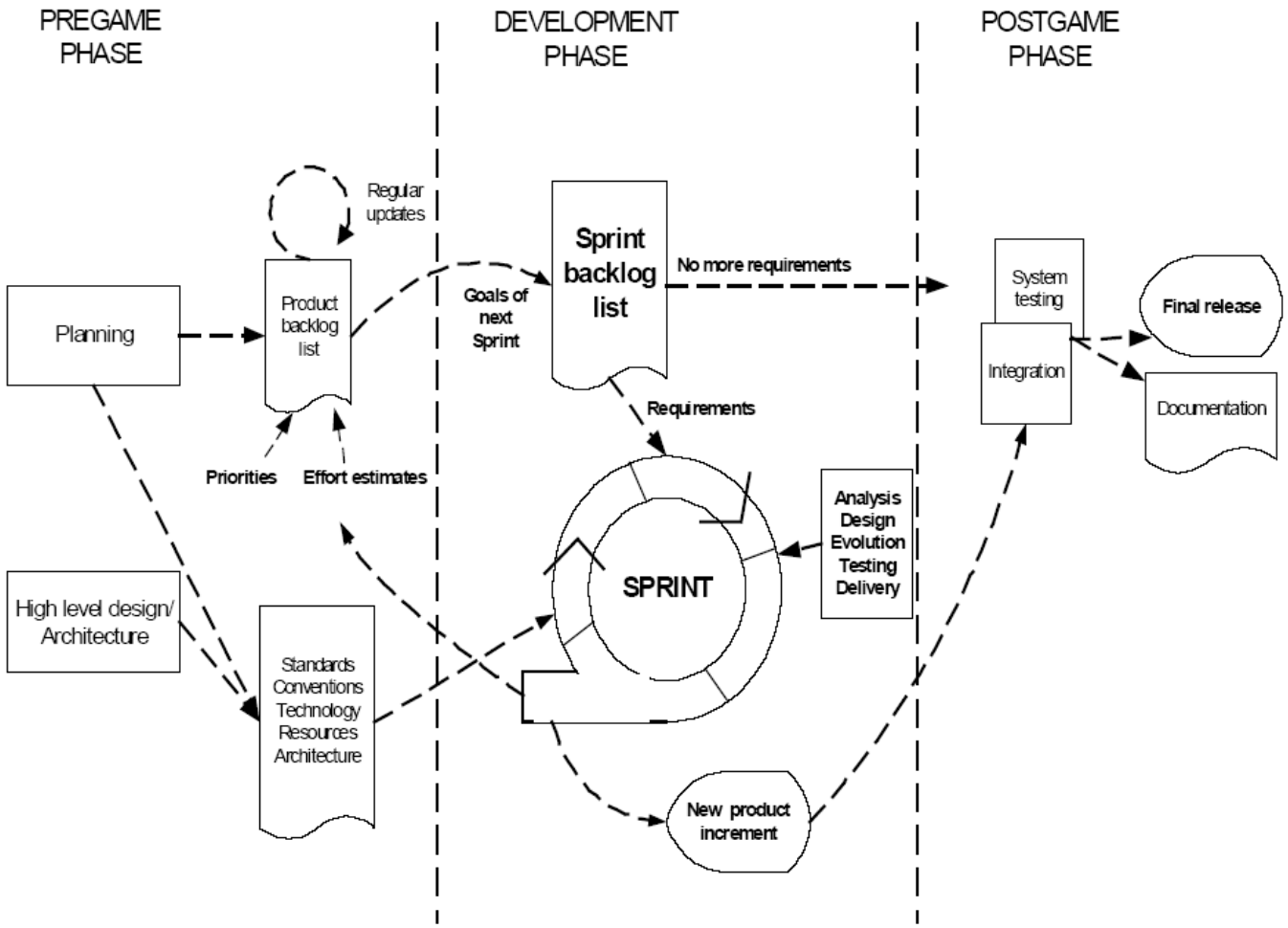
Scrum: Χαρακτηριστικά

- Αναπτύχθηκε για την καλύτερη δυνατή διαχείριση της διαδικασίας ανάπτυξης
- Εμπειρική προσέγγιση
- Εφαρμόζει τις ιδέες της θεωρίας ελέγχου βιομηχανικών διαδικασιών. Αποτέλεσμα μία διαδικασία
  - Ευέλικτη,
  - Ευπροσάρμοστη και
  - Παραγωγική (Schwaber and Beedle 2002)

Scrum: Χαρακτηριστικά

- Επικεντρώνεται στο διαχειριστικό κομμάτι
- Δεν ορίζει συγκεκριμένες τεχνικές ανάπτυξης
- Κεντρική ιδέα: Η ανάπτυξη ενός συστήματος ενέχει αρκετές περιβαλλοντικές και τεχνικές μεταβλητές ...
  - Απαιτήσεις,
  - Χρόνος,
  - Διαθέσιμοι πόροι,
  - Τεχνολογία
- ... οι οποίες είναι δυνατό να μεταβληθούν κατά τη διάρκεια της διαδικασίας

Scrum: Κύκλος Ζωής



Scrum: Αξιολόγηση

- Πλεονεκτήματα
  - Χρήση της λίστας Product Backlog
  - Συνεχής καθορισμός της προτεραιότητας και εκτίμηση της απαιτούμενης προσπάθειας
  - Συμμετοχή του πελάτη στον καθορισμό της Product Backlog λίστας
  - Ρόλος του Scrum Master
  - Ύπαρξη μιας αρκετά συγκροτημένης δομής συναντήσεων (Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review Meeting).
  - Ημερήσια Scrum συνάντηση (Daily Scrum Meeting)
- Μειονεκτήματα
  - Έλλειψη ορισμού συγκεκριμένων τεχνικών

Crystal Family

- Περιλαμβάνει:
  - 'Εναν αριθμό διαφορετικών μεθοδολογιών για την επιλογή της καταλληλότερης ανάλογα με το είδος του κάθε έργου
  - Αρχές για την αντιστοίχηση των μεθοδολογιών αυτών στις μεταβλητές συνθήκες του εκάστοτε έργου
- Δίνεται έμφαση στην επικοινωνία και συνεργασία μεταξύ των ατόμων
- Δεν περιορίζονται σε συγκεκριμένες πρακτικές ανάπτυξης, εργαλεία ή προϊόντα εργασίας
- Επιτρέπει την υιοθέτηση πρακτικών της XP ή της Scrum

Crystal Clear

- Για μικρά έργα (6 το πολύ προγραμματιστές)
- Ομάδα ανάπτυξης σε κοινό χώρο εργασίας
  - Λόγω των περιορισμών στη δομή της επικοινωνίας
- Βασικές αρχές
  - Αυξητική ανάπτυξη και έκδοση του λογισμικού σε τακτή βάση
  - Έλεγχος της προόδου με ορόσημα βασισμένα σε παραδοτέα λογισμικού και λήψη αποφάσεων και όχι σε έγγραφα
  - Άμεση συμμετοχή του τελικού χρήστη
  - Αυτοματοποιημένος έλεγχος λειτουργικότητας του λογισμικού
  - Συναντήσεις για ρύθμιση και έλεγχο προόδου του παραγόμενου και της μεθοδολογίας στην αρχή και στο τέλος κάθε επανάληψης

Crystal Clear: Αξιολόγηση

- Πλεονεκτήματα
  - Δυνατότητα επιλογής και προσαρμογής της κατάλληλης μεθόδου ανάλογα με το μέγεθος και την κρισιμότητα του κάθε έργου
  - Δυνατότητα εφαρμογής επιλεκτικά τεχνικών από άλλες agile μεθοδολογίες (π.χ. XP, Scrum)
- Μειονεκτήματα
  - Περιορισμένος αριθμός μεθόδων οι οποίες έχουν οριστεί
  - Περιορισμένη τεκμηρίωση και χρήση
    - Τα οφέλη από τη χρήση δεν έχουν καταγραφεί πλήρως

Μοντελοποίηση Διεργασιών

- Στατικό Μοντέλο:
  - Αναπαριστά τη διεργασία δείχνοντας τις εισόδους να μετασχηματίζονται σε εξόδους.
- Δυναμικό Μοντέλο:
  - Εκτελεί τη διεργασία ώστε να γίνει σαφές πώς τα ενδιάμεσα και τα τελικά προϊόντα μετασχηματίζονται με την πάροδο του χρόνου.

Στατική Μοντελοποίηση – Σημειολογία Lai

- Αντικείμενα:
  - Ρόλοι
  - Δραστηριότητες
  - Πόροι
  - Σχέσεις
- Φόρμες ορισμού:
  - Πίνακες με πληροφορία για την κατάσταση / πληρότητα κάθε αντικειμένου σε δεδομένη στιγμή.
- Πολλαπλά επίπεδα γενίκευσης / εξειδίκευσης.

Σημειολογία Lai – Τύποι Στοιχείων Διεργασίας

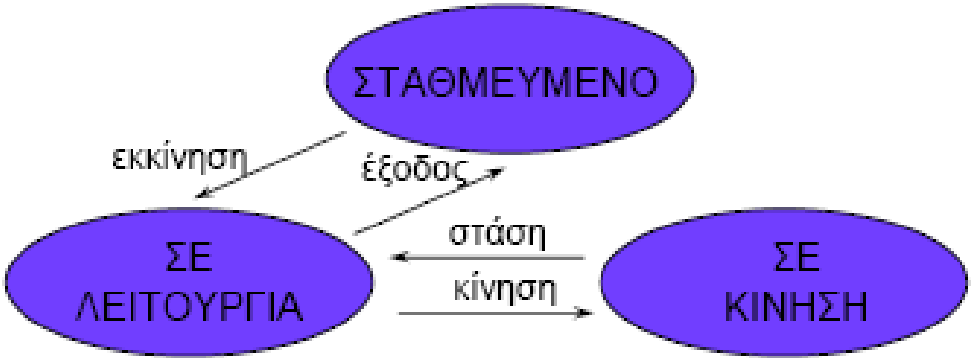
- Δραστηριότητα.
- Ακολουθία.
- Μοντέλο διεργασίας.
- Πόρος.
- Έλεγχος.
- Πολιτική.
- Οργανισμός.

Σημειολογία Lai – Φόρμα Ορισμού

Όνομα	***Car***	
Σύνοψη	***Το αντικείμενο που παριστάνει μια κλάση αυτοκινήτων***	
Πολυπλοκότητα	***Σύνθετο***	
Τύπος Δεδομένων	***(car c, ορισμένος από το χρήστη)***	
Καταστάσεις αντικειμένου		
Σταθμευμένο	***((state_of(car.engine) = off) (state_of(car.gear) = park) (state_of(car.speed) = stand))***	***Το αυτοκίνητο δεν κινείται και η μηχανή του δεν λειτουργεί***
Σε λειτουργία	***((state_of(car.engine) = on) (state_of(car.key_hole) = key_in) (state_of(car-driver(car.)) = in-car) (state_of(car.gear) = drive) (state_of(car.speed) = stand))***	***Το αυτοκίνητο δεν κινείται αλλά η μηχανή του είναι σε λειτουργία***
Σε κίνηση	***((state_of(car.engine) = on) (state_of(car.key_hole) = key_in) (state_of(car-driver(car.)) = driving) ((state_of(car.gear) = drive) or (state_of car.gear) = reverse)) (state_of(car.speed) = stand) or (state_of(car.speed) = slow) or (state_of(car.speed) = medium) or (state_of(car.speed) = high)))***	***Το αυτοκίνητο κινείται προς τα εμπρός ή προς τα πίσω***
Λίστα υποαντικειμένων		
	Doors	***Οι πόρτες του αυτοκινήτου***
	Engine	***Η μηχανή του αυτοκινήτου***
	Keyhole	***Η κλειδαριά της μίζας του αυτοκινήτου***
	Gear	***Η σχέση κιβωτίου του αυτοκινήτου***
	Speed	***Η ταχύτητα του αυτοκινήτου***
Λίστα σχέσεων		
Car-key	***Η σχέση μεταξύ αυτοκινήτου και κλειδιού***	
Car-driver	***Η σχέση μεταξύ αυτοκινήτου και οδηγού***	

Σημειολογία Lai – Διάγραμμα Μετάβασης

Σταθμευμένο:
((state_of(car.engine) = off)
(state_of(car.gear) = park)
(state_of(car.speed) = stand))



Δυναμική Μοντελοποίηση

- Εκτέλεση διεργασίας.
  - Διερεύνηση του τι συμβαίνει στους πόρους και τις οντότητες του συστήματος καθώς εκτελούνται διάφορες δραστηριότητες.
- Προσομοίωση εναλλακτικών επιλογών και υλοποίηση αλλαγών για τη βελτίωση της διεργασίας.
  - Οποιεσδήποτε αλλαγές συμβαίνουν πριν διατεθούν οι απαιτούμενοι πραγματικοί πόροι.

Παράδειγμα Δυναμικού Μοντέλου – Μοντέλο Παραγωγικότητας