Πανεπιστήμιο Αιγαίου

Τεχνολογία Λογισμικού

ΓΡΗΓΟΡΗ ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

Γιάννης Χαραλαμπίδης (yannisx@aegean.gr)

Αναπληρωτής Καθηγητής

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Στόχοι

- Εξήγηση του τρόπου με τον οποίο μια επαναληπτική, βαθμιαία προσέγγιση στην ανάπτυξη λογισμικού οδηγεί σε ταχύτερη παράδοση χρησιμότερων εφαρμογών
- Εξέταση των ουσιαστικών εννοιών των ευέλικτων μεθόδων ανάπτυξης
- Ερμηνεία των αρχών και των πρακτικών του ακραίου προγραμματισμού
- Εξήγηση του ρόλου της κατασκευής πρωτοτύπων στη διαδικασία παραγωγής λογισμικού

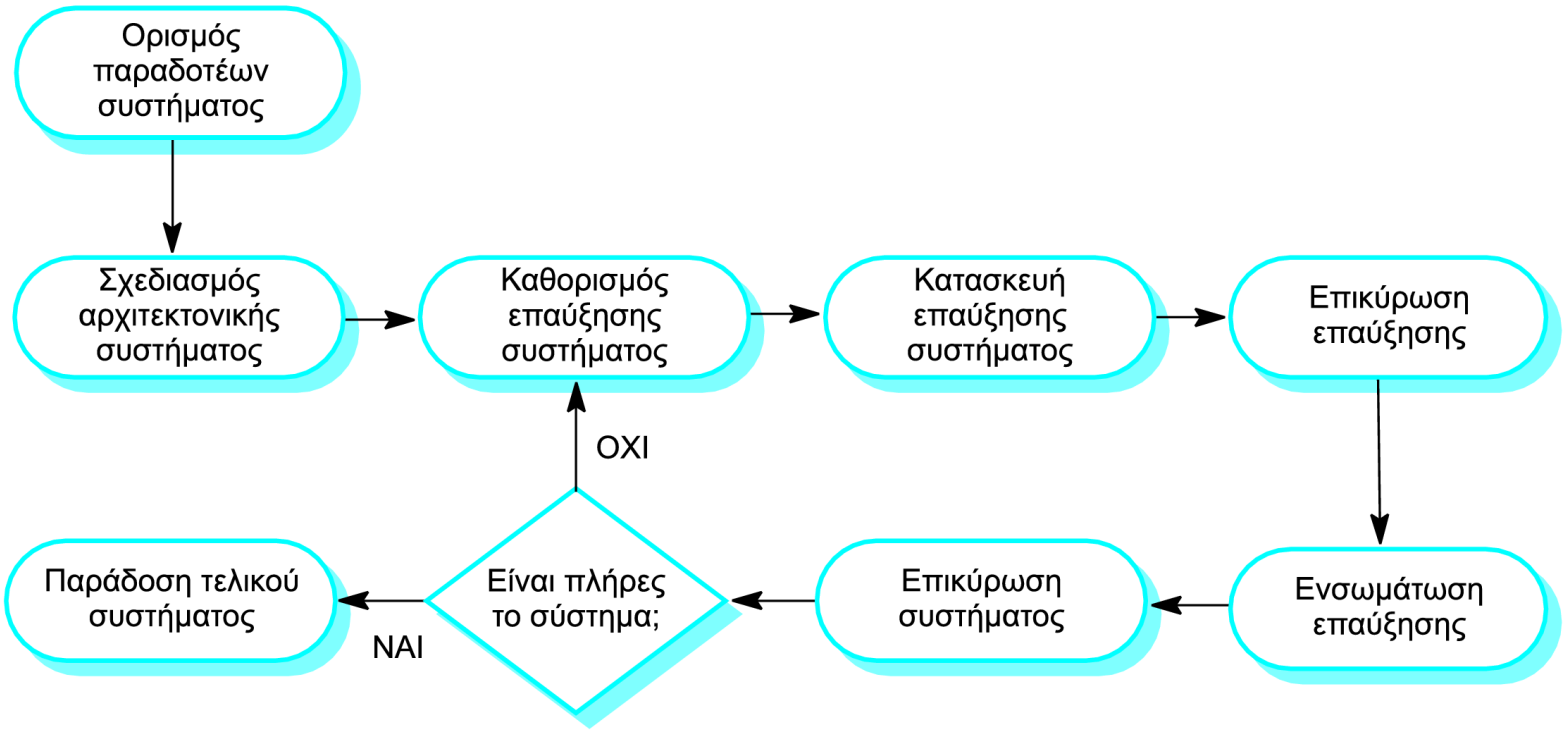
Περιεχόμενα

- Ευέλικτες μέθοδοι
- Ακραίος προγραμματισμός
- Γρήγορη ανάπτυξη εφαρμογών λογισμικού
- Κατασκευή πρωτοτύπων λογισμικού

Χαρακτηριστικά των διαδικασιών γρήγορης ανάπτυξης λογισμικού

- Οι διαδικασίες της κατάρτισης των προδιαγραφών, του σχεδιασμού και της υλοποίησης είναι ταυτόχρονες. Δεν υπάρχουν λεπτομερείς προδιαγραφές του συστήματος, ενώ η τεκμηρίωση του σχεδιασμού είναι η ελάχιστη δυνατή.
- Το σύστημα αναπτύσσεται σε μια σειρά επαυξήσεων. Οι τελικοί χρήστες του συστήματος αξιολογούν κάθε επαύξηση και προτείνουν αλλαγές για μεταγενέστερες επαυξήσεις.
- Οι διασυνδέσεις χρήστη της εφαρμογής συχνά αναπτύσσονται με τη χρήση ενός αλληλεπιδραστικού συστήματος.

Μια επαναληπτική διαδικασία ανάπτυξης



Πλεονεκτήματα της βαθμιαίας ανάπτυξης

- Ταχύτερη παράδοση υπηρεσιών στον πελάτη. Με κάθε επαύξηση παραδίδεται στον πελάτη η λειτουργική δυνατότητα με τη μεγαλύτερη προτεραιότητα.
- Συμμετοχή των χρηστών στο σύστημα. Οι χρήστες πρέπει να συμμετέχουν στη διαδικασία της ανάπτυξης, το οποίο συνεπάγεται ότι το σύστημα έχει περισσότερες πιθανότητες να ικανοποιεί τις ανάγκες τους και επίσης ότι και εκείνοι έχουν συνεισφέρει.

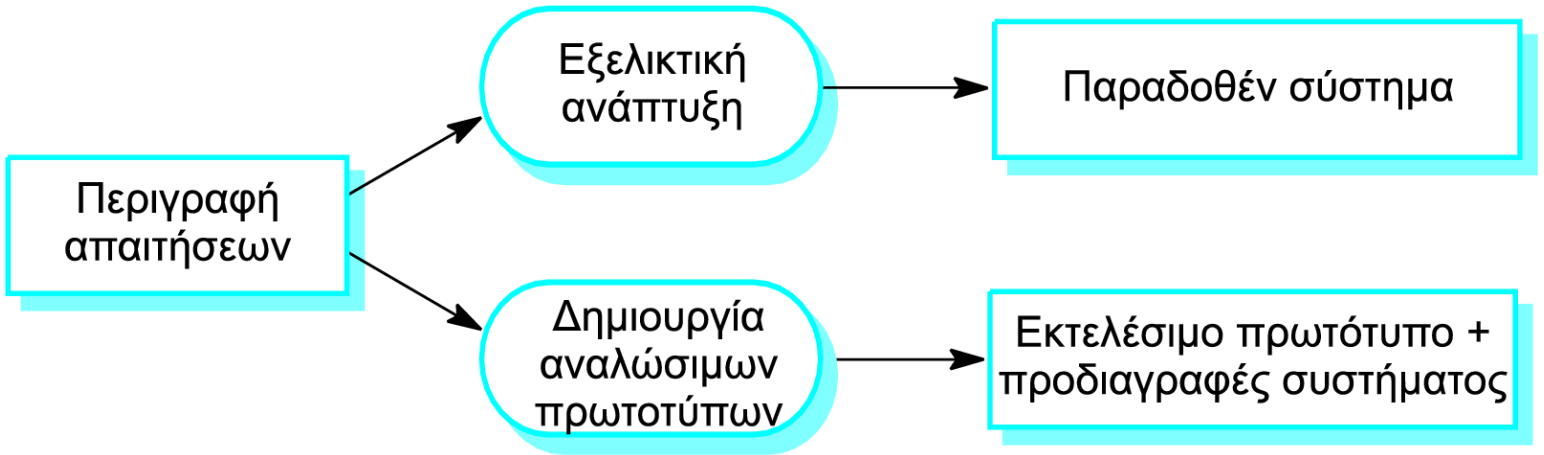
Προβλήματα με τη βαθμιαία ανάπτυξη

- Διαχειριστικά προβλήματα
  - Δύσκολα κρίνεται η πρόοδος του έργου και δύσκολα εντοπίζονται ενδεχόμενα προβλήματα επειδή δεν υπάρχει τεκμηρίωση η οποία παρουσιάζει τι έχει επιτευχθεί.
- Προβλήματα σύμβασης
  - Το σύνηθες μοντέλο σύμβασης βασίζεται σε κάποιο σύνολο προδιαγραφών. Όταν δεν υπάρχουν τέτοιες προδιαγραφές, πρέπει να αναζητηθούν άλλες μορφές συμβάσεων.
- Προβλήματα επικύρωσης
  - Αν δεν υπάρχουν προδιαγραφές, με ποιο μέτρο συγκρίνεται το σύστημα;
- Προβλήματα συντήρησης
  - Οι συνεχείς αλλαγές τείνουν να αλλοιώνουν τη δομή οποιουδήποτε συστήματος λογισμικού, καθιστώντας έτσι πιο ακριβή την πραγματοποίηση τροποποιήσεων σε αυτό και την εξέλιξή του ώστε να ανταποκρίνεται σε νέες απαιτήσεις.

Κατασκευή πρωτοτύπων

- Για πολύ μεγάλα συστήματα, η βαθμιαία ανάπτυξη και παράδοση δεν είναι η πιο πρακτική προσέγγιση, ειδικά όταν στην ανάπτυξη εμπλέκονται πολλές ομάδες οι οποίες εργάζονται σε διαφορετικές τοποθεσίες.
- Στην κατασκευή πρωτοτύπων, αναπτύσσεται ένα πειραματικό σύστημα το οποίο μπορεί να χρησιμοποιηθεί ως βάση για τη διατύπωση των απαιτήσεων. Αυτό το σύστημα απορρίπτεται όταν πλέον έχουν οριστικοποιηθεί οι προδιαγραφές του συστήματος.

Βαθμιαία ανάπτυξη και κατασκευή πρωτοτύπων



Ευέλικτες μέθοδοι

- Η δυσαρέσκεια την οποία προκαλούσαν οι πρόσθετες καθυστερήσεις που συνεπάγονταν οι μέθοδοι σχεδιασμού οδήγησαν στη δημιουργία των ευέλικτων μεθόδων. Αυτές:
  - επικεντρώνονται στον κώδικα και όχι στο σχεδιασμό
  - βασίζονται σε μια επαναληπτική προσέγγιση για την ανάπτυξη του λογισμικού
  - έχουν σκοπό τη γρήγορη παράδοση λογισμικού που λειτουργεί και την ταχεία εξέλιξή του για την ικανοποίηση των μεταβαλλόμενων απαιτήσεων.
- Οι ευέλικτες μέθοδοι πιθανότατα ταιριάζουν καλύτερα σε συστήματα μικρού ή μεσαίου μεγέθους ή σε προϊόντα υπολογιστών.

Αρχές των ευέλικτων μεθόδων

Αρχή	Περιγραφή
Συμμετοχή του πελάτη	Οι πελάτες πρέπει να συμμετέχουν στενά σε όλη τη διαδικασία της ανάπτυξης. Ο ρόλος τους είναι να παρέχουν νέες απαιτήσεις συστήματος και να καθορίζουν την προτεραιότητα τους, καθώς και να αξιολογούν τις επαναλήψεις του συστήματος.
Βαθμιαία παράδοση	Το λογισμικό αναπτύσσεται σε επαυξήσεις, και ο πελάτης καθορίζει τις απαιτήσεις που θα συμπεριληφθούν σε κάθε επαύξηση.
Άνθρωποι, όχι διαδικασίες	Πάντοτε πρέπει να προσδιορίζονται και να αξιοποιούνται οι ικανότητες της ομάδας ανάπτυξης. Τα μέλη της ομάδας πρέπει να έχουν το ελεύθερο να αναπτύξουν τους δικούς τους τρόπους εργασίας, χωρίς προδιαγεγραμμένες διαδικασίες.
Πρόβλεψη για μεταβολές	Να περιμένετε ότι οι απαιτήσεις συστήματος θα αλλάξουν, οπότε σχεδιάστε το σύστημα έτσι ώστε να είναι σε θέση να δεχτεί αυτές τις μεταβολές.
Διατήρηση απλότητας	Επικεντρωθείτε στην απλότητα τόσο του προς ανάπτυξη λογισμικού όσο και της διαδικασίας ανάπτυξης. Όπου είναι δυνατό, αφιερώστε προσπάθεια για να εξαλείψετε την πολυπλοκότητα από το σύστημα.

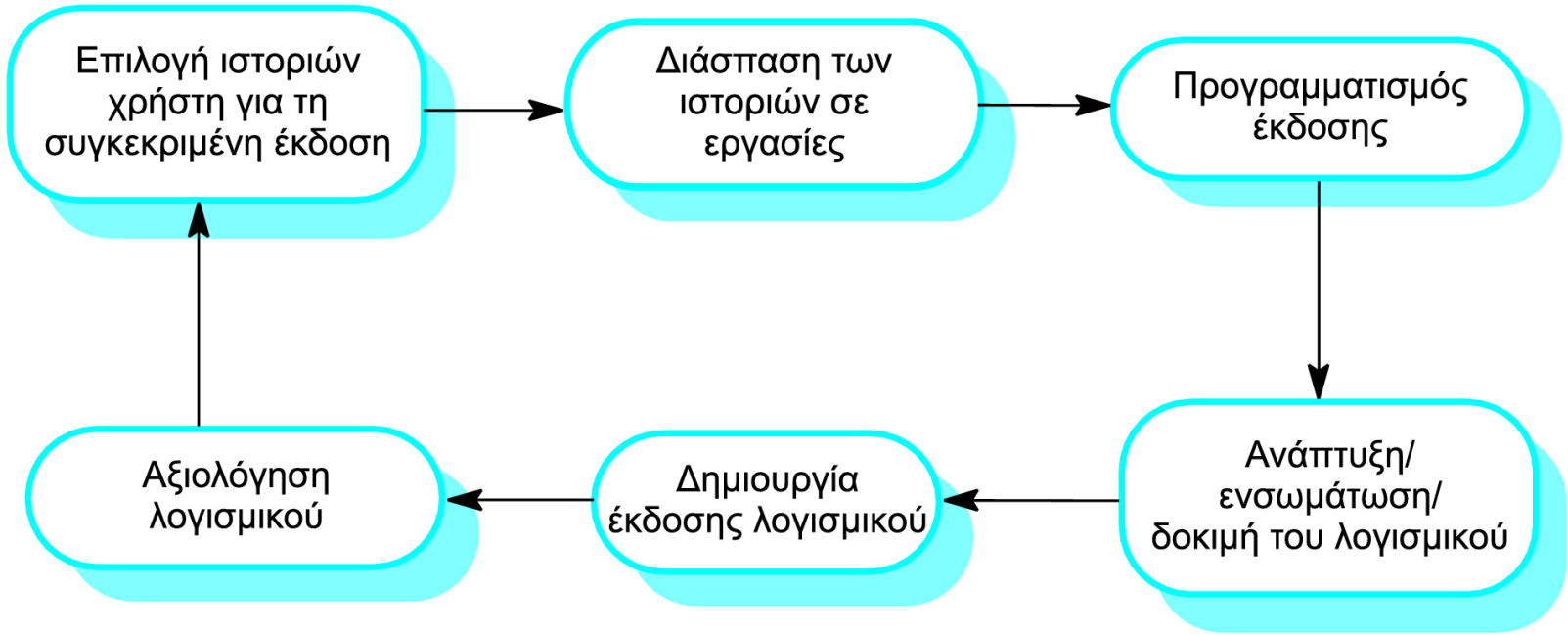
Προβλήματα των ευέλικτων μεθόδων

- Είναι δύσκολη η διατήρηση του ενδιαφέροντος των πελατών που εμπλέκονται στη διαδικασία.
- Κάποια άτομα της ομάδας ανάπτυξης ίσως να μην ταιριάζουν στην έντονη συμμετοχή που συνηθίζεται στις ευέλικτες μεθόδους.
- Η απόδοση προτεραιοτήτων σε μεταβολές μπορεί να αποδειχτεί εξαιρετικά δύσκολη υπόθεση σε συστήματα όπου υπάρχουν πολλοί ενδιαφερόμενοι.
- Η διατήρηση της απλότητας απαιτεί επιπλέον εργασία.
- Πρόβλημα μπορεί να αποτελέσουν και οι συμβάσεις, κάτι που μπορεί να συμβεί και σε άλλες προσεγγίσεις βαθμιαίας ανάπτυξης.

Ακραίος προγραμματισμός

- Ίσως η πιο γνωστή και πιο ευρέως διαδεδομένη ευέλικτη μέθοδος.
- Ο ακραίος προγραμματισμός (eXtreme Programming - XP) υιοθετεί μια "ακραία" προσέγγιση της βαθμιαίας ανάπτυξης.
  - Μπορεί να δομηθούν πολλές νέες εκδόσεις κάθε ημέρα
  - Οι επαυξήσεις παραδίδονται στους πελάτες κάθε 2 εβδομάδες
  - Για κάθε δόμηση πρέπει να εκτελεστούν όλες οι δοκιμές, ενώ η δόμηση αυτή γίνεται αποδεκτή μόνο αν όλες οι δοκιμές εκτελεστούν σωστά.

Ο κύκλος εκδόσεων του ακραίου προγραμματισμού



Πρακτικές του ακραίου προγραμματισμού (1)

Βαθμιαίος χρονοπρογραμματισμός	Οι απαιτήσεις καταγράφονται σε Κάρτες Ιστοριών και οι Ιστορίες που θα συμπεριληφθούν σε μια έκδοση καθορίζονται από το διαθέσιμο χρόνο και τη σχετική τους προτεραιότητα. Οι προγραμματιστές χωρίζουν αυτές τις Ιστορίες σε «Εργασίες» ανάπτυξης. Δείτε τις Εικόνες 17.6 και 17.7.
Μικρές εκδόσεις	Πρώτα αναπτύσσεται το ελάχιστο σύνολο λειτουργικότητας που είναι χρήσιμο στην επιχείρηση. Οι εκδόσεις του συστήματος είναι συχνές και προσθέτουν λειτουργικότητα στην πρώτη έκδοση με βαθμιαίο τρόπο.
Απλός σχεδιασμός	Πραγματοποιείται όσος σχεδιασμός είναι απαραίτητος για την ικανοποίηση των τρεχουσών απαιτήσεων, και όχι περισσότερος.
Ανάπτυξη βάσει δοκιμών	Χρησιμοποιείται ένα αυτοματοποιημένο πλαίσιο εργασίας για τη συγγραφή δοκιμών κάθε νέου τμήματος λειτουργικότητας, πριν υλοποιηθεί η ίδια η λειτουργικότητα.
Ανασύνθεση (refactoring)	Κάθε προγραμματιστής πρέπει συνεχώς να ανασυνθέτει τον κώδικα αμέσως μόλις εντοπίζονται πιθανές βελτιώσεις. Με αυτόν τον τρόπο ο κώδικας διατηρείται απλός και συντηρήσιμος.

Πρακτικές του ακραίου προγραμματισμού (2)

Προγραμματισμός ανά ζεύγη	Οι προγραμματιστές εργάζονται ανά ζεύγη, ελέγχοντας ο ένας τη δουλειά του άλλου και παρέχοντας υποστήριξη ώστε να γίνεται πάντοτε καλή δουλειά.
Συλλογική ιδιοκτησία	Τα ζεύγη των προγραμματιστών εργάζονται σε όλους τους τομείς του συστήματος, έτσι ώστε να μη δημιουργούνται απομονωμένες νησίδες εξειδίκευσης και όλος ο κώδικας να ανήκει σε όλους. Ο καθένας μπορεί να αλλάξει οτιδήποτε.
Συνεχής ενσωμάτωση	Μόλις ολοκληρωθεί μια εργασία, ενσωματώνεται στο συνολικό σύστημα. Μετά από κάθε τέτοια ενσωμάτωση, πρέπει να εκτελεστούν με επιτυχία όλες οι δοκιμές μονάδας (unit tests) του συστήματος.
Μη εξοντωτικοί ρυθμοί εργασίας	Οι πολλές επιπλέον ώρες εργασίας δε θεωρούνται αποδεκτή πρακτική, καθώς το τελικό αποτέλεσμα είναι να ελαττώνεται η ποιότητα του κώδικα και η παραγωγικότητα να κυμαίνεται σε μέτρια επίπεδα.
Πάντα διαθέσιμος πελάτης	Η ομάδα του ακραίου προγραμματισμού πρέπει να έχει κάθε στιγμή στη διάθεσή της έναν αντιπρόσωπο του τελικού χρήστη του συστήματος (τον Πελάτη). Σε μια διαδικασία ακραίου προγραμματισμού, ο πελάτης αποτελεί μέλος της ομάδας ανάπτυξης και είναι υπεύθυνος για να τροφοδοτεί την ομάδα με απαιτήσεις συστήματος προς υλοποίηση.

Σενάρια απαιτήσεων

- Στον ακραίο προγραμματισμό, όλες οι απαιτήσεις εκφράζονται με τη μορφή σεναρίων ή ιστορίες χρήστη.
- Γράφονται σε κάρτες και μετά η ομάδα ανάπτυξης τις διασπά σε εργασίες υλοποίησης. Αυτές οι εργασίες αποτελούν τη βάση των χρονικών και οικονομικών εκτιμήσεων.
- Ο πελάτης επιλέγει τις ιστορίες που θα συμπεριληφθούν στην επόμενη έκδοση με βάση τις προτεραιότητες και τις εκτιμήσεις που αφορούν το χρονοδιάγραμμα.

Κάρτα ιστορίας για το κατέβασμα εγγράφων

Κατέβασμα και εκτύπωση άρθρου

Καταρχήν επιλέγετε το άρθρο που θέλετε από τον κατάλογο που εμφανίζεται. Μετά πρέπει να ορίσετε στο σύστημα πώς θα πληρώσετε: μέσω συνδρομής, μέσω εταιρικού λογαριασμού, ή μέσω πιστωτικής κάρτας.

Κατόπιν, το σύστημα εμφανίζει μια φόρμα πνευματικών δικαιωμάτων για να τη συμπληρώσετε. Αφού την υποβάλετε, το άρθρο που ζητήσατε κατεβαίνει στον υπολογιστή σας.

Στη συνέχεια επιλέγετε έναν εκτυπωτή, στον οποίο τυπώνεται ένα αντίγραφο του άρθρου. Ενημερώνετε το σύστημα ότι η εκτύπωση ήταν επιτυχής.

Αν το άρθρο είναι μόνο για εκτύπωση, δεν μπορείτε να κρατήσετε την έκδοση PDF, οπότε διαγράφεται αυτόματα από τον υπολογιστή σας.

Ακραίος προγραμματισμός και πραγματοποίηση τροποποιήσεων

- Μια γενική αρχή της τεχνολογίας λογισμικού είναι ότι ο σχεδιασμός πρέπει να γίνεται λαμβάνοντας υπόψη το ενδεχόμενο πραγματοποίησης αλλαγών. Είναι σκόπιμη η επένδυση χρόνου και προσπάθειας για την πρόβλεψη αλλαγών καθώς αυτό θα μειώσει τα έξοδα σε μεταγενέστερα στάδια του κύκλου ζωής του συστήματος.
- Όμως, σύμφωνα με τον ακραίο προγραμματισμό, αυτό δεν αξίζει τον κόπο αφού οι διάφορες αλλαγές δεν μπορούν να προβλεφθούν με αξιόπιστο τρόπο.
- Αντ' αυτού, προτείνεται η συνεχής βελτίωση του κώδικα (ανασύνθεση) προκειμένου να διευκολυνθεί η διαδικασία αλλαγών όταν πρόκειται αυτές να υλοποιηθούν.

Εκτέλεση δοκιμών στον ακραίο προγραμματισμό

- Ανάπτυξη βάσει δοκιμών
- Βαθμιαία ανάπτυξη δοκιμών από σενάρια
- Συμμετοχή των χρηστών στην ανάπτυξη και επικύρωση των δοκιμών
- Κάθε φορά που δομείται μια νέα έκδοση, χρησιμοποιούνται αυτοματοποιημένα δοκιμαστικά προγράμματα για την εκτέλεση των δοκιμών όλων των συστατικών στοιχείων.

Περιγραφή περίπτωσης δοκιμής

Δοκιμή 4: Δοκιμή εγκυρότητας πιστωτικής κάρτας

Είσοδος:
Ένα αλφαριθμητικό που αντιπροσωπεύει τον αριθμό της πιστωτικής κάρτας, και δύο ακέραιοι που αντιπροσωπεύουν το μήνα και το έτος λήξης της κάρτας.
Δοκιμές:
Έλεγχος ότι όλα τα byte στο αλφαριθμητικό είναι αριθμητικά ψηφία
Έλεγχος ότι ο μήνας έχει τιμή από 1 μέχρι 12 και ότι το έτος είναι μεγαλύτερο ή ίσο από το τρέχον έτος.
Με βάση τα 4 πρώτα ψηφία του αριθμού της πιστωτικής κάρτας, έλεγχος για την εγκυρότητα του εκδότη της κάρτας με εξέταση του πίνακα εκδοτών καρτών
Έλεγχος εγκυρότητας της πιστωτικής κάρτας με υποβολή του αριθμού της και της ημερομηνίας λήξης της στον εκδότη
Έξοδος:
OK ή μήνυμα σφάλματος που υποδεικνύει ότι η κάρτα είναι άκυρη

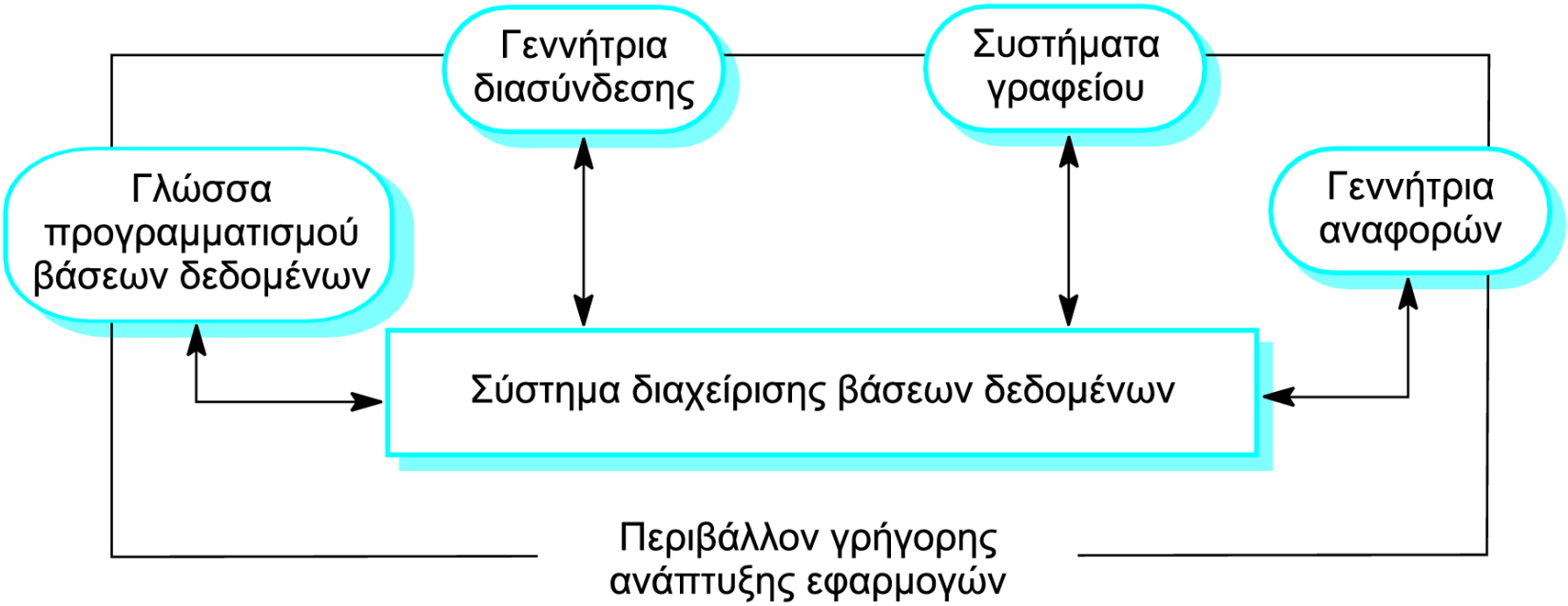
Προγραμματισμός ανά ζεύγη

- Στον ακραίο προγραμματισμό, οι προγραμματιστές εργάζονται ανά ζεύγη και μάλιστα βρίσκονται στον ίδιο σταθμό εργασίας για να αναπτύσσουν κώδικα μαζί.
- Έτσι αναπτύσσεται η κοινή ιδιοκτησία του κώδικα και οι γνώσεις κατανέμονται σε ολόκληρη την ομάδα.
- Δρα ως μια άτυπη διαδικασία επισκόπησης επειδή κάθε γραμμή κώδικα εξετάζεται τουλάχιστον από δύο άτομα.
- Ενθαρρύνει την ανασύνθεση αφού από αυτήν μπορεί να αποκομίσει οφέλη ολόκληρη η ομάδα.
- Σύμφωνα με τις μετρήσεις, η παραγωγικότητα στην ανάπτυξη με τον προγραμματισμό ανά ζεύγη είναι παρόμοια με εκείνη δύο ατόμων που εργάζονται ανεξάρτητα μεταξύ τους.

Εργαλεία ενός περιβάλλοντος γρήγορης ανάπτυξης λογισμικού

- Μια γλώσσα προγραμματισμού βάσεων δεδομένων
- Μια γεννήτρια διασύνδεσης
- Σύνδεσμοι σε εφαρμογές γραφείου
- Μια γεννήτρια αναφορών

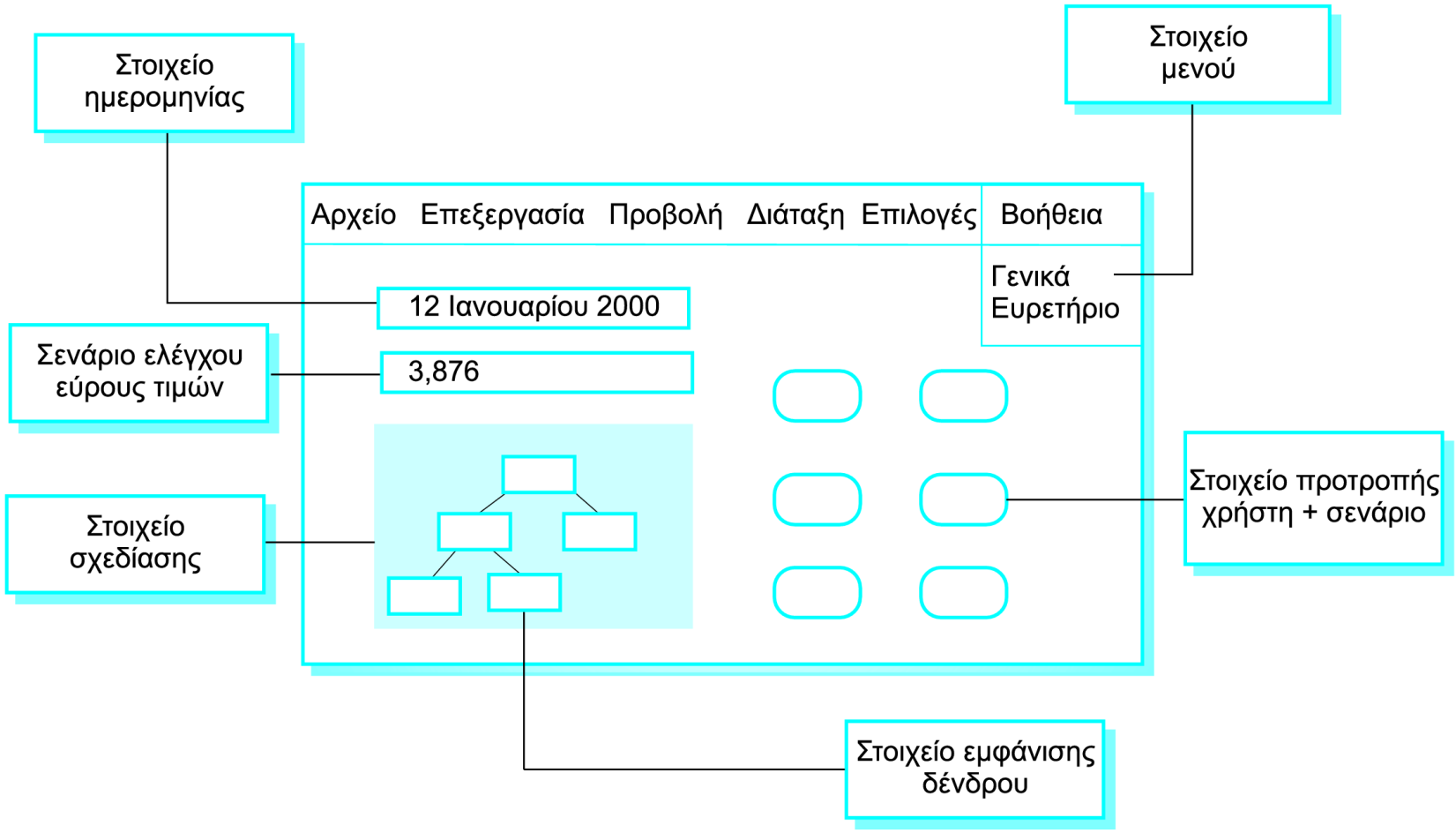
Ένα περιβάλλον γρήγορης ανάπτυξης λογισμικού



Γλώσσες οπτικού προγραμματισμού

- Ο οπτικός προγραμματισμός υποστηρίζεται από γλώσσες σεναρίων, όπως είναι η Visual Basic, στις οποίες αναπτύσσεται το πρωτότυπο μέσω της δημιουργίας μιας διασύνδεσης χρήστη από τυποποιημένα στοιχεία και τη συσχέτιση συστατικών στοιχείων με τα στοιχεία της διασύνδεσης
- Υπάρχει μεγάλη βιβλιοθήκη συστατικών στοιχείων για την υποστήριξη αυτού του τύπου ανάπτυξης
- Τα στοιχεία αυτά μπορούν να προσαρμοστούν ώστε να ταιριάζουν σε συγκεκριμένες απαιτήσεις των εφαρμογών

Οπτικός προγραμματισμός με επαναχρησιμοποίηση κώδικα



Επαναχρησιμοποίηση εμπορικών εφαρμογών

- Μια αποτελεσματική προσέγγιση για γρήγορη ανάπτυξη λογισμικού είναι η διευθέτηση και η σύνδεση υπάρχοντων εμπορικών συστημάτων.
- Για παράδειγμα, ένα σύστημα διαχείρισης απαιτήσεων θα μπορούσε να δημιουργηθεί με χρήση:
  - Μιας βάσης δεδομένων για την αποθήκευση των απαιτήσεων
  - Ενός επεξεργαστή κειμένου για την αποτύπωση των απαιτήσεων και της μορφοποίησης αναφορών
  - Μιας εφαρμογής λογιστικών φύλλων για διαχείριση ανιχνευσιμότητας

Σύνθετα έγγραφα

- Σε κάποιες εφαρμογές μπορούμε να δημιουργήσουμε ένα πρωτότυπο μέσω της ανάπτυξης ενός σύνθετου εγγράφου.
- Αυτό είναι ένα έγγραφο με ενεργά στοιχεία (για παράδειγμα, ένα λογιστικό φύλλο) τα οποία επιτρέπουν στο χρήστη να πραγματοποιεί υπολογισμούς.
- Με κάθε ενεργό στοιχείο συσχετίζεται μια συγκεκριμένη εφαρμογή, η οποία καλείται όταν επιλεγεί το στοιχείο.
- Το ίδιο το έγγραφο είναι το αντικείμενο που ενοποιεί τις διάφορες εφαρμογές.

Δημιουργία αναλώσιμων πρωτοτύπων

- Μετά την ανάπτυξη τα πρωτότυπα πρέπει να απορρίπτονται αφού δεν αποτελούν καλή βάση για ένα σύστημα μαζικής παραγωγής:
  - Η ρύθμιση του συστήματος με σκοπό να ικανοποιεί μη λειτουργικές απαιτήσεις μπορεί να μην είναι εφικτή
  - Κανονικά, δεν υπάρχει τεκμηρίωση για τα πρωτότυπα
  - Η δομή των πρωτοτύπων συνήθως υποβαθμίζεται μέσω των πραγματοποίησης γρήγορων τροποποιήσεων
  - Κατά πάσα πιθανότητα, το πρωτότυπο δεν θα πληροί τα συνήθη εταιρικά ποιοτικά πρότυπα.

Κύρια σημεία

- Μπορεί να επιτευχθεί ταχύτερη παράδοση του λογισμικού με μια επαναληπτική προσέγγιση της διαδικασίας ανάπτυξης.
- Οι ευέλικτες μέθοδοι είναι επαναληπτικές μέθοδοι ανάπτυξης που αποσκοπούν στη μείωση των επιβαρύνσεων στην ανάπτυξη και επομένως την ταχύτερη ανάπτυξη του λογισμικού.
- Ο ακραίος προγραμματισμός εμπεριέχει προγραμματιστικές πρακτικές όπως οι συστηματικές δοκιμές, η συνεχής βελτίωση του λογισμικού και η συμμετοχή του πελάτη στην ομάδα ανάπτυξης.
- Ένα ιδιαίτερα ισχυρό σημείο του ακραίου προγραμματισμού είναι η ανάπτυξη εκτελέσιμων δοκιμών πριν από τη δημιουργία του κώδικα του προγράμματος.

Κύρια σημεία

- Τα περιβάλλοντα γρήγορης ανάπτυξης εφαρμογών περιλαμβάνουν γλώσσες προγραμματισμού βάσεων δεδομένων, γεννήτριες φορμών και συνδέσμους προς εφαρμογές γραφείου.
- Για την εξερεύνηση των απαιτήσεων και των επιλογών σχεδιασμού χρησιμοποιούνται αναλώσιμα πρωτότυπα.
- Κατά την υλοποίηση ενός αναλώσιμου πρωτοτύπου, πρώτα ξεκινάμε με την ανάπτυξη των λιγότερο κατανοητών απαιτήσεων. Σε μια βαθμιαία προσέγγιση ανάπτυξης, η ανάπτυξη ξεκινά από τις απαιτήσεις που είναι περισσότερο κατανοητές.