Κεφάλαιο 13
ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

Σύνοψη
Σε αυτό το κεφάλαιο γίνεται αναφορά σε μεθοδολογίες και θέματα σχετικά με την ανάπτυξη έργων Πληροφοριακών Συστημάτων στο Διαδίκτυο (ΠΣΔ). Είναι δεδομένο ότι ένα τέτοιο σύστημα θα πρέπει πρώτα να έχει σαφή στόχο τη θεραπεία του προβλήματος ή της ανάγκης που καλείται να επιλύσει. Πρέπει να υιοθετηθεί σαφής μεθοδολογία ανάπτυξης, στην οποία η καταγραφή των απαιτήσεων από το ΠΣΔ να γίνει με ακρίβεια, προκειμένου να καταγραφούν οι λειτουργικές προδιαγραφές. Κατόπιν, εξετάζεται ο σχεδιασμός της εφαρμογής και της βάσης δεδομένων, και, προς τούτο, παρουσιάζεται συνοπτικά η γλώσσα UML. Ακολουθούν η υλοποίηση και ο έλεγχος της ορθότητας του λογισμικού εφαρμογής, με απώτερο σκοπό την παραγωγική λειτουργία της. Το κεφάλαιο κλείνει με αναφορά στις μεθόδους αξιολόγησης της αποτελεσματικότητας της εφαρμογής, που θα πρέπει να συνοδεύουν τη λειτουργία της.

Προαπαιτούμενη γνώση
1) Κεφάλαια 1-12 του παρόντος.

2) G. Hoffer-Valacich (2013), *Ανάλυση και σχεδίαση πληροφοριακών συστημάτων*, Α. Τζιόλας (Κωδικός Βιβλίου στον Εύδοξο: 33155341).

3) Ν. Δημητριάδης (επιμ.) (2003), *Σχεδιασμός – Ανάπτυξη – Λειτουργία Πληροφοριακών Συστημάτων*, Εκδόσεις Νέων Τεχνολογιών (Κωδικός Βιβλίου στον Εύδοξο: 111971).

13.1 Γενικά περί ανάπτυξης Πληροφοριακών Συστημάτων στο Διαδίκτυο

Στη σύγχρονη εποχή, με τη ραγδαία ανάπτυξη της πληροφορικής, έχει αναπτυχθεί πληθώρα εφαρμογών ΠΣΔ, άλλοτε επιτυχώς και άλλοτε ανεπιτυχώς. Κατά τη δημιουργία ενός ΠΣΔ, πρέπει να συνυπολογίζονται όλες εκείνες οι παράμετροι που θα επηρεάσουν τη μετέπειτα λειτουργία του,[1] μεταξύ των οποίων περιλαμβάνονται η αξιολόγησή του σε σχέση με τη σωστή και αποτελεσματική λειτουργία του, η φιλικότητά του προς το χρήστη και η εκπλήρωση των αναγκών της επιχείρησης.[2]

Η χρήση μιας μεθοδολογίας ανάπτυξης ενός ΠΣΔ συντελεί στην καταγραφή των απαιτήσεων και των επιθυμητών λειτουργιών.

Στην ανάπτυξη ενός ΠΣΔ εμπλέκονται, συνήθως, οι εξής:

- *Αναλυτές:* Πραγματοποιούν την ανάλυση απαιτήσεων, σε συνεχή συνεννόηση με τους χρήστες.
- *Σχεδιαστές συστημάτων:* Πραγματοποιούν τη σχεδίαση των συστημάτων, σε συνεχή συνεννόηση με τους χρήστες.
- *Προγραμματιστές:* Προγραμματίζουν και υλοποιούν το ΠΣΔ.
- *Ελεγκτές:* Ελέγχουν την ορθότητα της υλοποίησης, μέσα από δοκιμαστικούς ελέγχους.

Δεδομένης της πολυπλοκότητας και της κατανεμημένης φύσης ΠΣΔ, πολλές φορές προτείνονται νέα μοντέλα, τα οποία είναι πιο αποτελεσματικά στην παρουσίαση με οργανωμένο τρόπο των διαδικασιών ανάπτυξής τους.[2] Με τη χρήση των μοντέλων ανάπτυξης, επιτυγχάνεται η καλύτερη περιγραφή των φάσεων υλοποίησης του έργου, καθώς και των στόχων του. Με την υιοθέτηση ενός μοντέλου, επιτρέπεται η ανάπτυξη ενός αξιόπιστου πλάνου υλοποίησης, το οποίο είναι συνήθως συνυφασμένο με το μοντέλο ανάπτυξης του ΠΣ. Στο πλάνο υλοποίησης καθορίζονται οι φάσεις του έργου, οι στόχοι τους, ο τρόπος υλοποίησής τους, τα παραδοτέα που απαιτούνται και, εντέλει, τα κριτήρια επιτυχίας

τους. Μέσα από κατάλληλα εργαλεία, μπορεί ο υπεύθυνος του έργου να παρακολουθεί την υλοποίηση του έργου ανάπτυξης του ΠΣ, καθώς και τις αποκλίσεις από το αρχικό πλάνο, ώστε να προβεί σε διορθωτικές ενέργειες, εάν αυτές απαιτούνται. Όλα αυτά συνιστούν τη διαχείριση έργου, που δεν εξετάζεται στο παρόν κεφάλαιο.

Σε σχέση με τις συνήθεις μεθόδους που χρησιμοποιούνται για την περιγραφή της ανάπτυξης ενός ΠΣ, τα μοντέλα κύκλου ζωής (Systems Design Life Cycle/SDLC) είναι τα επικρατέστερα, γιατί εμπλέκουν όλες τις φάσεις του έργου, από τον προσδιορισμό των αναγκών μέχρι και την τελική αξιολόγηση επιτυχίας του έργου. Στο συγκεκριμένο κεφάλαιο, παρουσιάζεται το Μοντέλο του Καταρράκτη (Waterfall Model), το οποίο εστιάζεται κατά κύριο λόγο στην ανάλυση, τη σχεδίαση, την υλοποίηση, τον έλεγχο και την ενσωμάτωση.

Σημειώνεται ότι η μεθοδολογία αυτή εφαρμόζεται αποτελεσματικά στα ΠΣΔ, όπως έχει ήδη εφαρμοστεί και στα άλλα ΠΣ, με την παρατήρηση ότι σε κάθε φάση θα πρέπει λαμβάνονται υπόψη οι ιδιαιτερότητες που εισάγει το διαδίκτυο, οι υπηρεσίες νέφους, καθώς και αυτές της κινητής υπολογιστικής. Σε σχέση με τα παλαιότερα ΠΣ, οι ιδιαιτερότητες αυτές εισάγουν μερικές δομικές αλλαγές στον τρόπο με τον οποίο επηρεάζουν το περιβάλλον τους. Για παράδειγμα, κατά τη φάση ανάλυσης απαιτήσεων ή/και σχεδιασμού ενός ΠΣΔ, μπορεί να διαπιστωθεί η δυνατότητα για νέες μορφές επιχειρείν, που μπορεί να δώσει το ίδιο το ΠΣΔ στην επιχείρηση, λόγω των ειδικών χαρακτηριστικών του. Παράδειγμα αποτελεί η διείσδυση της επιχείρησης, παράλληλα με τους λόγους για τους οποίους αναπτύσσεται το ΠΣΔ στην Ελλάδα, σε νέες αγορές, που μπορεί να συνεπάγεται πρόσθετες απαιτήσεις, όπως άλλου τύπου ενσωματωμένο μάρκετινγκ στον ιστότοπο (π.χ. για την Κίνα) ή αποδιαμεσολάβηση από μεσάζοντες μέσω της διαδικτυακής εφαρμογής, οι οποίοι ήταν παλαιότερα απαραίτητοι για το άνοιγμα σε μια τέτοια αγορά. Αυτό ανατροφοδοτεί (feedback) την ίδια τη φάση ανάλυσης ή τη μετέπειτα φάση σχεδίασης της διαδικτυακής εφαρμογής.

13.2 Ο χώρος του προβλήματος

Πριν γίνει αναφορά στις φάσεις ανάπτυξης ενός έργου, πρέπει να γίνει αναφορά στην ανάγκη η επιχείρηση ή ο οργανισμός να έχει καταλάβει καλά το χώρο του προβλήματος (problem space). Ο όρος «χώρος του προβλήματος» αναφέρεται στο σύνολο των συνιστωσών που διαμορφώνουν το περιβάλλον λειτουργίας μιας επιχείρησης ή ενός οργανισμού και γεννούν την απαίτηση για ανάπτυξη νέων ή συμπληρωματικών ΠΣ.

Κατανοώντας το χώρο του προβλήματος, μια επιχείρηση αποκτά τακτικό πλεονέκτημα έναντι του ανταγωνισμού, επειδή έτσι μπορεί καλύτερα να διαστασιολογήσει τις ανάγκες της και να προβλέψει τις ενέργειες που πρέπει να κάνει, για να τοποθετηθεί στρατηγικά στο χώρο. Το τελευταίο βέβαια προϋποθέτει να γνωρίζει τι θέλει να πετύχει και πώς βλέπει τον εαυτό της μέσα στην αγορά. Αυτό το τελευταίο αφορά στην παρούσα θεματολογία κυρίως *τις ηλεκτρονικές αγορές και το διαδίκτυο*. Με άλλα λόγια, μια επιχείρηση θα πρέπει να αναπτύξει αξιόπιστη *ψηφιακή επιχειρηματική στρατηγική* (digital business strategy) και, συνακόλουθα, *στρατηγική των πληροφοριακών της συστημάτων* (Information Technology/IT strategy), με στόχο να ικανοποιήσει τη στρατηγική της κατ' ελάχιστον ή/και να δημιουργήσει ευκαιρίες για επαναδιαμόρφωση της στρατηγικής αυτής.

Η διαμόρφωση της στρατηγικής απαιτεί μια σειρά από ενέργειες, όπως είναι η ανάπτυξη ενός στρατηγικού χάρτη των στοιχείων που διαμορφώνουν το επιθυμητό αποτέλεσμα, η ανάπτυξη αξιόπιστου επιχειρηματικού πλάνου, ο εντοπισμός των ευκαιριών και των αδυναμιών της επιχείρησης στην αγορά, καθώς και η αξιολόγηση του κέρδους από την επένδυση στα νέα ΠΣΔ. Επίσης, η υλοποίηση ενός έργου ανάπτυξης και εισαγωγής ενός νέου ΠΣ στην επιχείρηση απαιτεί και την αξιολόγησή του. Προς τούτο, εφαρμόζονται διάφορες μέθοδοι, οι οποίες έχουν επί της ουσίας κύρια κριτήρια τη βελτίωση των εσωτερικών διαδικασιών της επιχείρησης, με στόχο καλύτερα προϊόντα ή καλύτερες υπηρεσίες, που θα οδηγήσουν, με τη σειρά τους, σε περισσότερο ευχαριστημένους πελάτες και, εντέλει, σε καλύτερα οικονομικά αποτελέσματα.

Οι σωστές επιλογές για την επιτυχή ανάπτυξη ενός ΠΣΔ είναι ένα πολυδιάστατο θέμα. Η κατανόηση του επιχειρησιακού περιβάλλοντος και του χώρου του προβλήματος, η

σωστή στρατηγική τοποθέτηση στην αγορά, η ευθυγράμμιση της στρατηγικής της πληροφορικής με αυτήν της επιχειρηματικής και η συνεχής αποτίμηση και αξιολόγηση των επιλογών της είναι καίριοι παράγοντες για την επιτυχία της ανάπτυξης των έργων πληροφορική και, με την ευρύτερη έννοια, της ίδιας της επιχείρησης.

13.3 Ανάπτυξη έργων πληροφοριακής σε επιχειρήσεις και οργανισμούς

Σε αυτή την ενότητα παρουσιάζεται μια τυπική προσέγγιση του τρόπου με τον οποίο αναπτύσσονται συνήθως τα έργα πληροφορικής σε οργανισμούς ή επιχειρήσεις, αφότου έχουν αποφασίσει τι θα κάνουν και πώς. Με βάση την προσέγγιση αυτή, η επιχείρηση δεν αναπτύσσει η ίδια τις εφαρμογές της, αλλά τις αγοράζει, πρακτική η οποία υιοθετείται από τις περισσότερες επιχειρήσεις. Προφανώς, αυτό δεν σημαίνει ότι αγοράζει ένα μαύρο κουτί (black box). Θέλει να έχει τον έλεγχο και την εποπτεία αυτού που αγοράζει. Η επιχείρηση ανάπτυξης λογισμικού, δηλαδή η εταιρεία πληροφορικής, πρέπει να εφαρμόσει όλα τα στάδια ανάπτυξης ενός ΠΣ, την ανάλυση, το σχεδιασμό, την ανάπτυξη και τον έλεγχο του κώδικα, την εγκατάστασή του και, εντέλει, τη συντήρησή του.

Στην εταιρεία-πελάτη πρέπει κάποιοι να διαμορφώσουν την πρότασή τους για την αγορά του συστήματος. Αυτό συνήθως γίνεται από το τμήμα επιχειρηματικής ανάπτυξης ή το τμήμα μάρκετινγκ, προφανώς με γνώμονα την αύξηση της πελατειακής βάσης και του κέρδους. Το υπό ανάπτυξη/παραγγελία ΠΣ μπορεί να έχει άμεση σχέση με τους πελάτες της επιχείρησης, όπως είναι ένα σύστημα Διαχείρισης Σχέσεων Πελατών (Customer Relationship Management/CRM), ή έμμεση, όπως είναι το λογισμικό αρωγής (helpdesk) των υπαλλήλων.

Για να υλοποιηθεί το ΠΣ, πρέπει, καταρχάς, η υπεύθυνη ομάδα για την υπηρεσία (π.χ. ο διαχειριστής προϊόντων ή πελατολογίων) να καταγράψει όλες τις ανάγκες και απαιτήσεις που πρέπει να ικανοποιεί το νέο σύστημα, καθώς και το σύνολο των επιχειρησιακών κανόνων και διαδικασιών που θα πρέπει να ακολουθεί. Προφανώς, ο διαμορφωτής των προδιαγραφών από τη μεριά της επιχείρησης θα πρέπει να συνυπολογίσει και να ρωτήσει με αμεσότητα τους χρήστες που θα χειρίζονται τα νέα συστήματα και τις νέες εφαρμογές, επειδή αυτοί θα πρέπει να γνωρίζουν τις λεπτομέρειες της επιχειρησιακής λογικής (business logic) ή θα πρέπει να είναι φιλικά προσκείμενοι στο νέο ΠΣ.

Βέβαια, η προδιαγραφή του συστήματος ή της υπηρεσίας δεν αρκεί, εάν δεν υπάρχουν ο κατάλληλος προϋπολογισμός, καθώς και η κατάλληλη προετοιμασία, για την εισαγωγή του συστήματος στο επιχειρησιακό περιβάλλον. Γι' αυτόν το σκοπό, γίνονται μελέτες κυρίως από τα τμήματα οικονομικών και πληροφορικής της επιχείρησης. Οι αρμόδιοι υπάλληλοι αυτών των τμημάτων επιβλέπουν την υλοποίηση του έργου. Αφού αυτό υλοποιηθεί, το τμήμα μάρκετινγκ αναλαμβάνει την προώθησή του στην αγορά. Τα παραπάνω τμήματα θα αναλάβουν να μελετήσουν την πρόταση του τμήματος μάρκετινγκ, όσον αφορά την εφικτότητα και τις τεχνοοικονομικές λεπτομέρειές του. Τις σχετικές μελέτες θα τις δρομολογήσουν οι αναλυτές συστημάτων/επιχειρηματικότητας (business/system analysts) από τη μεριά της πληροφορικής.

Ο προμηθευτής του ΠΣ θα πρέπει, από την πλευρά του, να εξετάσει ενδελεχώς εάν αυτά που ζητά η επιχείρηση μπορούν να υλοποιηθούν ή όχι. Επίσης, θα πρέπει να εξετάσει τη μετάβαση από την παλαιά κατάσταση στη νέα, ανεξάρτητα εάν πρόκειται για αναβαθμίσεις των υπαρχόντων συστημάτων ή για αντικαταστάσεις τους. Αφού αποκτήσει εικόνα ή, ακόμα καλύτερα, πλήρη αντίληψη αυτού που θέλει ο πελάτης του, θα πρέπει να εκτιμήσει σε πραγματική βάση τους χρόνους υλοποίησης, καθώς και το κόστος. Εάν αυτά είναι ανελαστικά από την πλευρά του πελάτη, ενώ ταυτόχρονα δεν είναι ρεαλιστικά, τότε πιθανότατα να δημιουργηθεί αδιέξοδο στην υλοποίηση του ΠΣ, αλλιώς θα πρέπει τα δύο μέρη να προχωρήσουν σε διαπραγματεύσεις, προκειμένου να έρθουν σε συμφωνία. Εντέλει, ο προμηθευτής ή, αλλιώς, ο κατασκευαστής του ΠΣ θα πρέπει να δώσει ένα αξιόπιστο πλάνο υλοποίησης, που να αφορά τα συμφωνηθέντα και να συμβαδίζει με τους κανόνες της τέχνης και της επιστήμης της πληροφορικής.

Στην περίπτωση της διαπραγμάτευσης, ενδέχεται να υπάρξουν αρκετοί κύκλοι μέχρι τα δύο μέρη να φτάσουν σε πλήρη συμφωνία και, επιπρόσθετα, να έχουν καλή κοινή αντίληψη του έργου. Οι κύκλοι διαπραγμάτευσης μπορεί να έχουν σχέση με την εκτίμηση του κόστους, του χρόνου, της ερμηνείας μιας απαίτησης και της δυνατότητας συνεννόησης των μερών μεταξύ τους. Εύλογα, οι κύκλοι διαπραγμάτευσης δεν μπορεί να είναι αέναοι, ενώ θα πρέπει να κλείνουν σε πραγματικό χρόνο, σύμφωνα με τις ανάγκες της επιχείρησης-πελάτη και τις αναπροσαρμογές που θα γίνουν σε κόστος και χρόνο, και σύμφωνα με τις απαιτήσεις μετά την ολοκλήρωσή τους.

Αφού ολοκληρωθούν οι διαπραγματεύσεις και αρχίσει η υλοποίηση του έργου, η κατασκευάστρια εταιρεία πρέπει να προχωρήσει στο σχεδιασμό του νέου ΠΣ, αφού έχει ήδη αναλύσει τις απαιτήσεις σε βάθος. Ενδεχομένως, και η σχεδίαση του νέου ΠΣ θα πρέπει να αξιολογηθεί από τον πελάτη που το αγοράζει, καθώς ο καλός σχεδιασμός μπορεί να είναι ζωτικής σημασίας γι' αυτόν, όπως στην περίπτωση ιστότοπων που προωθούν πολυτελείς ξενοδοχειακές υπηρεσίες. Σε αυτή την περίπτωση, θα πρέπει να υπάρξει έγκριση από τον πελάτη, η οποία μπορεί να οδηγήσει σε νέο κύκλο διαπραγματεύσεων μεταξύ των δύο μερών. Εντέλει, η κατασκευάστρια εταιρεία, αφού υπάρξει συμφωνία, ξεκινά την υλοποίηση του ΠΣ.

Ο κώδικας υλοποίησης μπορεί να έχει συμφωνηθεί να παραδοθεί αυτούσιος στον πελάτη, αν και αυτό δεν είναι το πιο σύνηθες, δεδομένου ότι ο πελάτης ενδιαφέρεται κυρίως για την εγγυημένη λειτουργία αυτού που αγοράζει και, ως εκ τούτου, τον ενδιαφέρει πρωτίστως η εγγύηση καλής λειτουργίας και συντήρησης του προϊόντος. Προφανώς, το προϊόν πρέπει να περάσει από προσυμφωνημένους ελέγχους της ορθότητας του κώδικα.

Οι έλεγχοι πραγματοποιούνται σύμφωνα με τα κριτήρια αποδοχής που έχει προδιαγράψει ο πελάτης, τα οποία αποτελούν ουσιαστικά περιπτώσεις χρήσης (use cases) της εφαρμογής. Εάν κάπου σημειωθεί απόκλιση, τότε ενεργοποιείται η διαδικασία διόρθωσης του κώδικα, με παράλληλη τήρηση ημερολογίου, για την καταγραφή των διορθώσεων και των αλλαγών στον κώδικα. Επίσης, καλό είναι να υπάρχει ένα σύστημα παρακολούθησης εκδόσεων λογισμικού (versioning system).

Ωστόσο, δεν αρκεί μόνο η αποδοχή από τη Διεύθυνση Πληροφορικής (IT Department) ή από αυτούς που παρακολουθούν την υλοποίηση του έργου. Θα πρέπει να υπάρχει και αποδοχή από τους χρήστες του νέου συστήματος ή της νέας υπηρεσίας. Εάν οι χρήστες είναι πελάτες του πελάτη, τότε μπορεί να απαιτηθεί η διενέργεια συνεντεύξεων μέσω ερωτηματολογίων. Έτσι, λοιπόν, οι παρακολουθούντες το έργο ή οι ιδιοκτήτες του έργου (project owners), και ανάλογα με την περίπτωση, σε συνεργασία με τον κατασκευαστή-προμηθευτή, βοηθούν τους τελικούς χρήστες να εκτελέσουν τους δικούς τους ελέγχους, τους ελέγχους αποδοχής από τους χρήστες (User Acceptance Tests/UAT), που βρίσκονται ένα ακριβώς βήμα πριν να μπει το σύστημα σε τελική λειτουργία (go live). Τα συμμετέχοντα μέρη, δηλαδή ο τελικός επιχειρησιακός χρήστης (business user) και ο άνθρωπος της πληροφορικής, που είναι και ο ιδιοκτήτης του έργου, σε στενή συνεργασία με την κατασκευάστρια εταιρεία, συμφωνούν ότι το νέο ΠΣ μπορεί να μπει στην παραγωγική λειτουργία. Αυτή η διαδικασία μπορεί να απαιτήσει επαναληπτικούς κύκλους, μέχρι να υπάρξει συμφωνία, η οποία θα πρέπει να επιτευχθεί σε πραγματικό χρόνο. Εάν τα προηγούμενα στάδια υλοποίησης του έργου έχουν γίνει σωστά, αυτό το στάδιο δεν πρέπει να παρουσιάζει επαναληπτικούς κύκλους, και σε κάθε περίπτωση όχι πάνω από δύο ή τρεις. Κατόπιν τούτου, το νέο ΠΣ μπαίνει σε παραγωγική διαδικασία.

Εάν το έργο είναι αρκετά μεγάλο, τότε, με το πέρασμα του χρόνου, μπορεί να εμφανισθούν και νέες απαιτήσεις ή απαιτήσεις για αλλαγές σε σχέση με το αρχικό πλάνο προδιαγραφών. Αυτό μπορεί να οδηγήσει στην ακύρωση του έργου, εάν οι αλλαγές είναι δραματικές και ασύμφορες να υλοποιηθούν, ή στην υλοποίηση νέου έργου. Βέβαια, το καλύτερο είναι, ανάλογα με τη φύση του έργου, να υπάρχει σχετική πρόβλεψη για κάτι τέτοιο, ώστε να γίνονται, ενδεχομένως με επιπρόσθετο, εύλογο για τον πελάτη, κόστος, οι συμπληρωματικές αναπτύξεις/αλλαγές.

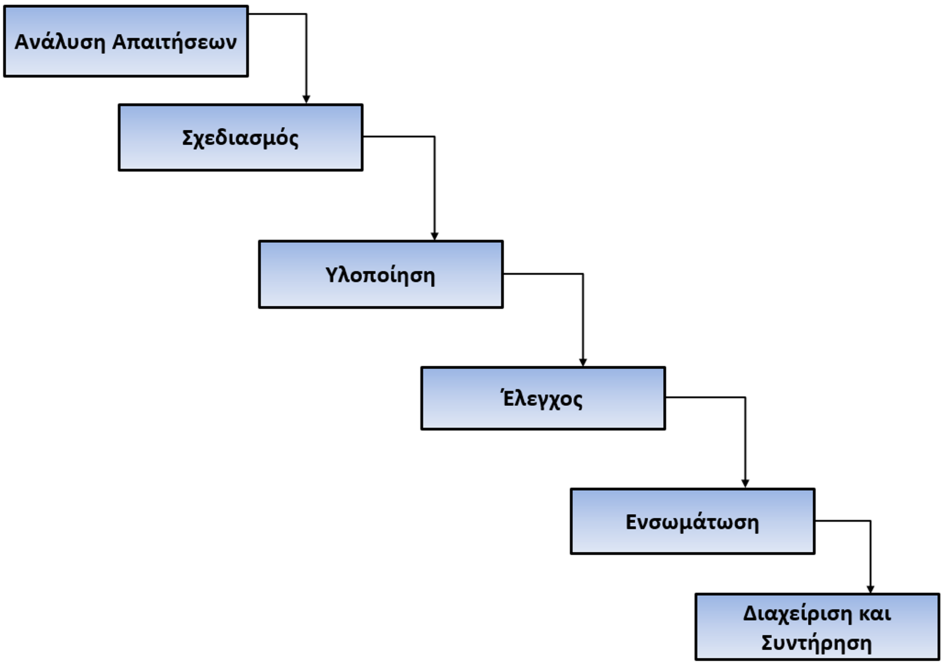
13.4 Μεθοδολογία ανάπτυξης λογισμικού – Μοντέλο του Καταρράκτη

Η τεχνολογία λογισμικού δεν ασχολείται μόνο με τις τεχνικές ανάπτυξης του λογισμικού, αλλά και με θέματα όπως η διαχείριση έργου και η ανάπτυξη εργαλείων, μεθόδων και θεωριών για την υποστήριξη της κατασκευής του λογισμικού. Παρακάτω παρουσιάζονται τα γενικά στάδια σχετικά με τον κύκλο ζωής και ανάπτυξης λογισμικού (Software Development Life Cycle):[3]

- *Ανάλυση απαιτήσεων* (requirement analysis): Υλοποιείται από βασικά στελέχη της ομάδας ανάπτυξης λογισμικού, τα οποία βρίσκονται σε συνεχή επικοινωνία με τον πελάτη, και πιθανούς χρήστες του συστήματος, ώστε να πάρουν πληροφορίες μέσω αυτών. Στόχος είναι να πραγματοποιηθεί η βασική προσέγγιση της δημιουργίας του λογισμικού και να διαπιστωθεί εάν αυτή είναι εφικτή σε σχέση με οικονομικούς, τεχνικούς και λειτουργικούς παράγοντες.
- *Ορισμός απαιτήσεων* (defining requirements): Εφόσον πραγματοποιηθεί η ανάλυση απαιτήσεων, στη συνέχεια ορίζονται οι απαιτήσεις για το λογισμικό και πραγματοποιείται τεκμηρίωση (documentation). Ο ορισμός απαιτήσεων γίνεται με τη σύμφωνη γνώμη του πελάτη.
- *Σχεδιασμός αρχιτεκτονικής του προϊόντος* (product architecture design): Με βάση τον ορισμό των απαιτήσεων, επιλέγεται η βέλτιστη αρχιτεκτονική για τη δημιουργία του προϊόντος.
- *Δημιουργία του λογισμικού* (developing the product): Παράγεται ο κώδικας με βάση την επιλεγμένη αρχιτεκτονική για το προϊόν. Προτείνεται η χρήση κατευθυντήριων γραμμών για την παραγωγή κώδικα, οι οποίες μπορεί να προέρχονται από την επιχείρηση ή από τους μεταγλωττιστές της εκάστοτε γλώσσας.
- *Δοκιμή του λογισμικού* (product testing): Γίνονται δοκιμές, για τον εντοπισμό και την εξάλειψη των ελαττωμάτων του προϊόντος, σύμφωνα με τις απαιτήσεις που έχουν οριστεί.
- *Κυκλοφορία στην αγορά και συντήρηση* (deploying in the market and maintenance): Εφόσον έχουν γίνει οι απαραίτητες δοκιμές, το προϊόν κυκλοφορεί στην αγορά και συντηρείται μέσω αλλαγών στο λογισμικό και συνεχούς αξιολόγησης.

Για την ανάπτυξη λογισμικού, χρησιμοποιείται, μεταξύ άλλων μεθοδολογιών, το Μοντέλο του Καταρράκτη. Η ονομασία του οφείλεται στο γεγονός ότι το μοντέλο/μεθοδολογία αναπτύσσεται σταδιακά από τη μία φάση στην άλλη, με ροή προς τα κάτω, όπως οι καταρράκτες.[1] Στην Εικόνα 13.1 παρουσιάζεται το Μοντέλο του Καταρράκτη.

[1] http://www.waterfall-model.com/ (πρόσβαση: 30-6-2015).



Εικόνα 13.1 Κύκλος ζωής λογισμικού: Μοντέλο του Καταρράκτη.

Στη συνέχεια δίνεται μια σύντομη αναφορά στις φάσεις ανάπτυξης του λογισμικού με τη χρήση του Μοντέλου του Καταρράκτη:

- *Ανάλυση απαιτήσεων*: Διερευνώνται οι ανάγκες μιας επιχείρησης ή ενός οργανισμού σε σχέση με το ΠΣ, με αποτέλεσμα να αναλύονται τι ακριβώς θα κάνει και ποιες απαιτήσεις θα ικανοποιεί.
- *Βασική σχεδίαση:* Με την ολοκλήρωση της προηγούμενης φάσης, αρχίζει η φάση δημιουργίας ενός προσχεδίου, για τον τρόπο υλοποίησης του λογισμικού.
- *Τεχνική και λεπτομερής σχεδίαση*: Εφόσον η βασική σχεδίαση είναι αποδεκτή, σχεδιάζεται μια τεχνικά λεπτομερής υλοποίηση του λογισμικού, ενώ περιγράφονται οι λειτουργίες του κάθε στοιχείου.
- *Υλοποίηση*: Γράφεται ο πηγαίος κώδικας του προγράμματος.
- *Έλεγχος*: Ελέγχονται ο σχεδιασμός του λογισμικού, καθώς και η υλοποίησή του, για να διαπιστωθεί εάν συμφωνεί με τις προδιαγραφές λειτουργίας του. Αν βρεθούν λάθη,πρέπει να επιλυθούν.
- *Ενσωμάτωση*: Αρχίζουν η χρήση του λογισμικού και η παραγωγική του λειτουργία.
- *Διαχείριση και συντήρηση*: Εξασφαλίζεται ότι το λογισμικό θα συνεχίσει να λειτουργεί απρόσκοπτα και σύμφωνα με τις προδιαγραφές που έχουν τεθεί.

Σύμφωνα με τις βασικές αρχές, το αποτέλεσμα κάθε φάσης είναι ένα ή περισσότερα έγγραφα, που πρέπει να περάσουν τη διαδικασία της έγκρισης. Στην πράξη, αυτές οι φάσεις αλληλεπιδρούν και ανταλλάσσουν πληροφορίες μεταξύ τους. Κατά το σχεδιασμό, προκύπτουν διάφορα προβλήματα, σχετικά με τη μη τήρηση των απαιτήσεων, τη διάρκεια του προγραμματισμού κ.ά. Η ανάπτυξη του λογισμικού δεν είναι ένα γραμμικό μοντέλο, αλλά περιλαμβάνει επαναλήψεις των βημάτων εκτέλεσης.

Το αυξημένο κόστος της κατασκευής και της έγκρισης εγγράφων αλλαγής καθιστά απαγορευτικές τις συχνές επαναλήψεις των βημάτων εξέλιξης. Ως εκ τούτου, έπειτα από ένα μικρό αριθμό επαναλήψεων, είναι λογικό να σταματά η διαδικασία για ορισμένο χρονικό διάστημα. Στη διάρκεια της τελευταίας φάσης του κύκλου ζωής, το λογισμικό μπαίνει σε

λειτουργία. Γίνεται εύρεση και έλεγχος για τυχόν λάθη και παραλείψεις στις αρχικές απαιτήσεις/προδιαγραφές του λογισμικού, ενώ παράλληλα μπορεί να απαιτηθεί και η τροποποίησή του, σύμφωνα με νέα δεδομένα. Η συντήρηση του λογισμικού σημαίνει την επανάληψη κάποιων ή όλων των προηγούμενων φάσεων της διαδικασίας ανάπτυξης, συνήθως σε μικρότερη κλίμακα από την αρχική τους υλοποίηση.

Επειδή το Μοντέλο του Καταρράκτη δεν ανταποκρίνεται τόσο αποτελεσματικά σε τυχόν αλλαγές των απαιτήσεων από τον πελάτη, έχει σημασία να κατανοηθούν πλήρως οι απαιτήσεις στην αρχική φάση της σχεδίασης. Έτσι, μειώνονται δραστικά τυχόν παραλείψεις και λάθη.

Συνοπτικά και πριν από την παρουσίαση των επιμέρους φάσεων, ουσιαστικά ξεκινά η ανάπτυξη ενός ΠΣΔ, με την εκπόνηση μιας μελέτης για την αναγκαιότητά του, την εφικτότητά του, τις πιθανές τεχνολογίες ανάπτυξης και την ασφάλεια για την υλοποίησή του. Αφού γίνει η ανάλυση των απαιτήσεων και των περιορισμών, ορίζονται αναλυτικές προδιαγραφές για το σύστημα. Στη φάση σχεδιασμού του συστήματος, γίνεται λεπτομερέστερη ανάλυση του συστήματος στο σύνολό του και ανά διεργασία ξεχωριστά, και σχεδιάζονται οι επιμέρους λειτουργίες του, με τη βοήθεια διαγραμμάτων (π.χ. σε UML, όπως θα δούμε παρακάτω). Επιπλέον, σχεδιάζεται η βάση δεδομένων με διάγραμμα κλάσεων και σχεσιακά διαγράμματα, και ορίζεται η επιθυμητή μορφή της διεπαφής του χρήστη. Τέλος, ακολουθούν η υλοποίηση και η τελική εφαρμογή του ΠΣ.

13.5 Εντοπισμός και ανάλυση των απαιτήσεων

Σε αρκετές περιπτώσεις, τα προβλήματα που πρέπει να επιλύσουν οι μηχανικοί λογισμικού είναι πολύπλοκα. Η κατανόηση της φύσης των προβλημάτων μπορεί να είναι πολύ δύσκολη. Σε αυτό το στάδιο, προσδιορίζονται τα προβλήματα που καλείται να λύσει το ΠΣ, ο τρόπος με τον οποίο θα αντιμετωπιστεί τις υπάρχουσες λύσεις, καθώς και οι εναλλακτικές επιλογές. Επίσης, καθορίζονται οι λειτουργίες που πρέπει να υποστηρίζει το ΠΣ, καθώς και ο τρόπος με τον οποίο θα αντιμετωπίσει τις υπάρχουσες ανάγκες.

13.5.1 Είδη απαιτήσεων

Κάποια προβλήματα που προκύπτουν κατά τον προσδιορισμό των απαιτήσεων του συστήματος είναι συνέπεια της αποτυχίας να υπάρξει σαφής διαχωρισμός ανάμεσα στα διαφορετικά επίπεδα περιγραφής. Για το λόγο αυτό, θα γίνει ένας διαχωρισμός μεταξύ των απαιτήσεων από την πλευρά του χρήστη, δηλαδή των εισαγωγικών απαιτήσεων, και των απαιτήσεων που αφορούν το ΠΣ, μέσα από τη λεπτομερή περιγραφή της λειτουργίας του. Οι απαιτήσεις κατηγοριοποιούνται ως εξής:

- *Απαιτήσεις του χρήστη*: Είναι δηλώσεις σε φυσική γλώσσα και διαγράμματα, που αναφέρονται στις υπηρεσίες που αναμένεται να παρέχει το σύστημα και τους περιορισμούς υπό τους οποίους θα πρέπει να λειτουργεί.
- *Απαιτήσεις του ΠΣ*: Εκφράζουν τις υπηρεσίες και τους περιορισμούς του συστήματος, με λεπτομέρειες. Το έγγραφο των απαιτήσεων του συστήματος, που αναφέρονται ως λειτουργικές προδιαγραφές, πρέπει να είναι σαφές.
- *Προδιαγραφές του σχεδιασμού λογισμικού*: Είναι μια εισαγωγική περιγραφή της υλοποίησης του λογισμικού του ΠΣ, η οποία αποτελεί τη βάση για τον λεπτομερή σχεδιασμό και την υλοποίησή του. Αυτές οι προδιαγραφές καθιστούν σαφέστερο τον προσδιορισμό των απαιτήσεων του συστήματος και σχετίζονται με την υλοποίηση του συστήματος και τους μηχανικούς λογισμικού που ασχολούνται με την υλοποίηση και τον προγραμματισμό του ΠΣ.

Οι απαιτήσεις κατηγοριοποιούνται ως εξής:

- *Λειτουργικές απαιτήσεις*: Είναι οι απαιτήσεις που ορίζουν τον τρόπο λειτουργίας του ΠΣ, δηλαδή πώς το σύστημα θα πρέπει να αντιδρά στην εισαγωγή συγκεκριμένων δεδομένων και ποιος θα είναι ο τρόπος απόκρισής του σε συγκεκριμένες περιπτώσεις. Σε μερικές περιπτώσεις, οι λειτουργικές απαιτήσεις μπορεί επίσης να αναφέρονται σε ενέργειες που δεν θα πρέπει να πραγματοποιούνται από το σύστημα.
- *Μη λειτουργικές απαιτήσεις*: Είναι οι περιορισμοί που θέτει το σύστημα στις υπηρεσίες ή στις λειτουργίες του. Περιλαμβάνουν χρονικούς περιορισμούς, περιορισμούς στη διαδικασία της ανάπτυξης, διάφορα πρότυπα κ.ά.
- *Απαιτήσεις τομέα*: Είναι απαιτήσεις που προκύπτουν από την εφαρμογή του ΠΣ και αντικατοπτρίζουν χαρακτηριστικά αυτού του τομέα. Μπορεί να είναι είτε λειτουργικές είτε μη λειτουργικές απαιτήσεις.

13.5.2 Έγγραφο προσδιορισμού απαιτήσεων

Το έγγραφο προσδιορισμού απαιτήσεων αποτελεί την επίσημη δήλωση των ενεργειών που θα πρέπει να κάνουν οι αναλυτές του συστήματος. Θα πρέπει να περιλαμβάνει και τις απαιτήσεις του χρήστη σε σχέση με το σύστημα, αλλά και, λεπτομερώς, τις απαιτήσεις του συστήματος. Επίσης, πρέπει να περιλαμβάνει ένα σύνολο χρηστών, από τους υπεύθυνους της διαχείρισης του οργανισμού, για τους οποίους υλοποιείται το σύστημα, μέχρι τους μηχανικούς λογισμικού.

13.6 Σχεδίαση

Η σωστή σχεδίαση ενός ΠΣΔ, αλλά και γενικά οποιουδήποτε ΠΣ, είναι μια πολύ σημαντική διαδικασία. Προφανώς, ένα ΠΣ πρέπει να σχεδιάζεται με γνώμονα την ευχρηστία του και τη φιλικότητά του προς το χρήστη. Η ευχρηστία κερδίζει το χρήστη ως «πελάτη» του συστήματος, αλλά επίσης δημιουργεί μεγαλύτερη ανοχή σε σφάλματα (fault tolerance).

Η *γραφική σχεδίαση* αναφέρεται στην εμφάνιση του συστήματος και την οργάνωση, εν γένει, του περιβάλλοντος διεπαφής χρήστη. Η εφαρμογή των αρχών γραφικής σχεδίασης μπορεί να διασφαλίσει ότι το ΠΣ θα είναι, τουλάχιστον, ευχάριστο οπτικά στους χρήστες.

Η *αναλυτική σχεδίαση* αναφέρεται στον καλύτερο δυνατό τρόπο αναπαράστασης των πληροφοριών, δηλαδή εάν είναι ποιοτικές ή ποσοτικές, καθώς και στη σωστή και αποτελεσματική επικοινωνία των διαφόρων μερών ενός ΠΣ.

Κάθε ΠΣ έχει ποσοτικές διαστάσεις σχετικές με τη διαχείριση του έργου ανάπτυξής του, όπως είναι το κόστος και η ανάλυσή του στα επιμέρους κόστη, το χρονοδιάγραμμα, το ανθρώπινο δυναμικό, καθώς και η απαιτούμενη υποδομή.[2]

13.6.1 Η UML: Ένα χρήσιμο εργαλείο σχεδίασης συστημάτων λογισμικού

Η UML (Unified Modeling Language) είναι γλώσσα μοντελοποίησης, η οποία χρησιμοποιείται ευρέως στην τεχνολογία λογισμικού. Στόχος της είναι να οπτικοποιήσει τη σχεδίαση ενός συστήματος λογισμικού. Παρακάτω παρουσιάζονται μερικοί από τους επικρατέστερους τύπους διαγραμμάτων που διαθέτει η γλώσσα UML, όπως των περιπτώσεων χρήσης, των κλάσεων, της ακολουθίας, των καταστάσεων, των δραστηριοτήτων, των στοιχείων και της συνεργασίας.[4]

13.6.1.1 Διαγράμματα περιπτώσεων χρήσης

[2] http://catalog.flatworldknowledge.com/bookhub/2579?e=frost-ch01_s02 (πρόσβαση: 30-6-2015).

Τα διαγράμματα περιπτώσεων χρήσης (use case diagrams) αναπαριστούν περιπτώσεις χρήσης (use cases) του ΠΣ, ενεργοποιητές (actors) και σχέσεις μεταξύ περιπτώσεων χρήσης και ενεργοποιητών. Μια περίπτωση χρήσης στη UML παρουσιάζεται στην Εικόνα 13.2.

Εικόνα 13.2 Συμβολισμός περίπτωσης χρήσης.

Μια περίπτωση χρήσης αντιστοιχεί σε συγκεκριμένο τρόπο χρήσης του συστήματος. Έχει σχέση με τη λειτουργικότητα του συστήματος, το οποίο ενεργοποιείται για να ανταποκριθεί σε έναν εξωτερικό ενεργοποιητή. Ο συμβολισμός του ενεργοποιητή στη UML φαίνεται στην Εικόνα 13.3.

Εικόνα 13.3 Συμβολισμός ενεργοποιητή.

Ένας ενεργοποιητής αντιπροσωπεύει το ρόλο ενός χρήστη ή μιας οντότητας που αλληλεπιδρά με το σύστημα. Οι ενεργοποιητές προσδιορίζονται παρατηρώντας τους άμεσους χρήστες του συστήματος (αυτούς που το χρησιμοποιούν και το συντηρούν).

Εικόνα 13.4 «Σχέση επικοινωνεί».

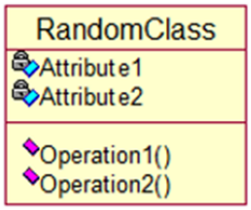
Η συμμετοχή του ενεργοποιητή συμβολίζεται με μια γραμμή μεταξύ του ενεργοποιητή και της περίπτωσης χρήσης. Η «σχέση επικοινωνεί» (communicates) είναι η μοναδική σχέση που μπορεί να υπάρξει μεταξύ ενεργοποιητών και περιπτώσεων χρήσης (Εικόνα 13.4).

Εικόνα 13.5 «Σχέση χρησιμοποιεί».

Η «σχέση χρησιμοποιεί» (uses) ορίζεται μεταξύ δύο περιπτώσεων χρήσης και δηλώνει ότι ένα στιγμιότυπο της πηγής συμπεριλαμβάνει τη συμπεριφορά του στόχου (Εικόνα 13.5). Χρησιμοποιείται όταν πολλές περιπτώσεις χρήσης μοιράζονται στοιχεία της ίδιας λειτουργικότητας. Αυτή η λειτουργικότητα τοποθετείται σε μια ξεχωριστή περίπτωση χρήσης.

13.6.1.2 Διαγράμματα κλάσεων

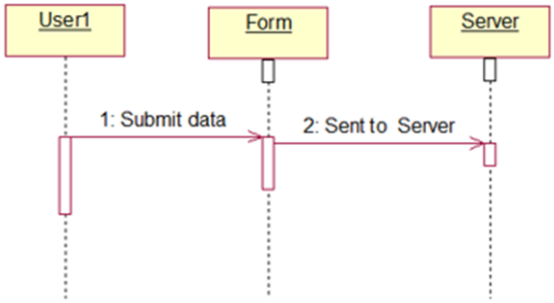
Τα διαγράμματα κλάσεων (class diagrams) χρησιμοποιούνται στη φάση ανάλυσης και σχεδιασμού. Επειδή κατά την ανάλυση δεν έχει προσδιοριστεί η ακριβής λειτουργία των κλάσεων, το διάγραμμα σε αυτήν τη φάση δημιουργείται μόνο με τα ονόματα των κλάσεων. Στη UML, μια κλάση αντιπροσωπεύεται από ένα ορθογώνιο με μία ή περισσότερες οριζόντιες γραμμές. Στο πάνω επίπεδο του διαγράμματος βρίσκεται το όνομα της κλάσης, το οποίο είναι και το μόνο απαραίτητο στοιχείο στο εν λόγω διάγραμμα. Το όνομα μιας κλάσης αρχίζει με κεφαλαίο γράμμα. Στο μεσαίο επίπεδο (εφόσον υπάρχει) βρίσκονται τα μέρη και τα χαρακτηριστικά (attributes) της κλάσης και στο κάτω επίπεδο (εφόσον υπάρχει) οι μέθοδοι της κλάσης. Ένα διάγραμμα κλάσεων παρουσιάζεται στην Εικόνα 13.6.



Εικόνα 13.6 Διάγραμμα κλάσεων.

13.6.1.3 Διαγράμματα ακολουθίας

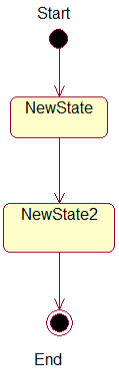
Τα διαγράμματα ακολουθίας (sequence diagrams) χρησιμοποιούνται στη φάση ανάλυσης και σχεδιασμού. Αναπαριστούν μια περίπτωση χρήσης μέσω μιας ακολουθίας γεγονότων. Αρχικά, γίνεται ταυτοποίηση των αντικειμένων που παίρνουν μέρος σε μια περίπτωση χρήσης. Έπειτα, τα αντικείμενα κατατάσσονται σε στήλες, με το όνομά τους υπογραμμισμένο. Σε αυτόν τον τύπο διαγράμματος, η οριζόντια διάσταση αντιπροσωπεύει το χρόνο, ενώ η κάθετη διάσταση τα διάφορα αντικείμενα. Η ροή γεγονότων γίνεται από τα αριστερά προς τα δεξιά. Ένα διάγραμμα ακολουθίας παρουσιάζεται στην Εικόνα 13.7.



Εικόνα 13.7 Διάγραμμα ακολουθίας.

13.6.1.4 Διαγράμματα καταστάσεων

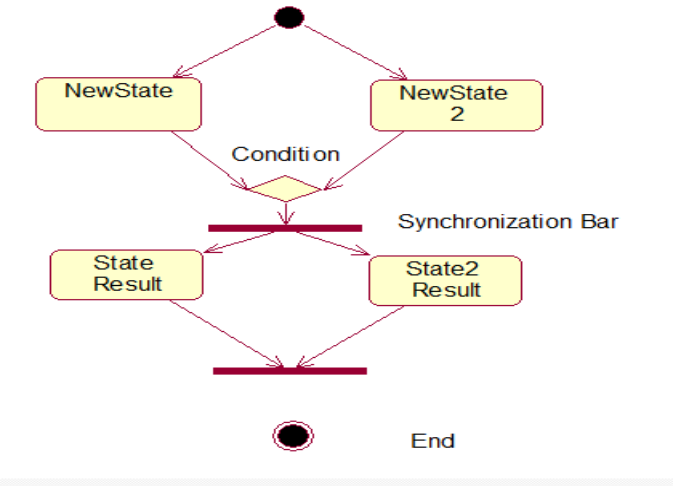
Τα διαγράμματα καταστάσεων (state diagrams) χρησιμοποιούνται στη φάση ανάλυσης και σχεδιασμού, και περιγράφουν τη συμπεριφορά των αντικειμένων, δηλαδή όλες τις πιθανές καταστάσεις τους στη διάρκεια του χρόνου. Μια κατάσταση αντιπροσωπεύεται από ένα παραλληλόγραμμο με κυκλικές άκρες και η αρχή της με έναν κύκλο. Το τέλος μιας κατάστασης αντιπροσωπεύεται από έναν κύκλο εσωτερικό σε έναν άλλον κύκλο. Μια κατάσταση μετάβασης είναι η αλλαγή κατάστασης ενός αντικειμένου και αντιπροσωπεύεται από ένα βέλος, το οποίο ενώνει δύο καταστάσεις. Ένα διάγραμμα καταστάσεων παρουσιάζεται στην Εικόνα 13.8.



Εικόνα 13.8 Διάγραμμα καταστάσεων.

13.6.1.5 Διαγράμματα δραστηριοτήτων

Τα διαγράμματα δραστηριοτήτων (activity diagrams) χρησιμοποιούνται στη φάση σχεδιασμού των λειτουργιών του συστήματος και περιγράφουν τη σειρά ενεργειών μιας περίπτωσης χρήσης. Είναι μια τροποποιημένη έκδοση των διαγραμμάτων καταστάσεων. Μια δραστηριότητα είναι μια ενέργεια που προκαλείται από έναν εξωτερικό παράγοντα. Τα διαγράμματα δραστηριοτήτων χρησιμοποιούν τους ίδιους συμβολισμούς με αυτούς των διαγραμμάτων καταστάσεων. Ωστόσο, έχουν δύο άλλους συμβολισμούς: το σύμβολο απόφασης (decision symbol) και τη γραμμή συγχρονισμού (synchronization bar). Το σύμβολο απόφασης έχει σχήμα ρόμβου και αξιολογεί αν μια συνθήκη είναι αληθής ή ψευδής. Η γραμμή συγχρονισμού επιτρέπει το συμβολισμό παράλληλης εκτέλεσης προγραμμάτων. Ένα διάγραμμα δραστηριοτήτων παρουσιάζεται στην Εικόνα 13.9.



Εικόνα 13.9 Διάγραμμα δραστηριοτήτων.

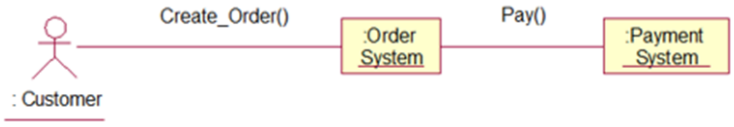
13.6.1.6 Διαγράμματα στοιχείων

Τα διαγράμματα στοιχείων (component diagrams) παρουσιάζουν τις δομικές σχέσεις μεταξύ των στοιχείων (όπως εκτελέσιμα αρχεία ή βιβλιοθήκες) που συνθέτουν ένα ΠΣ. Ένα ενδεικτικό παράδειγμα παρουσιάζεται στην Εικόνα 13.10, η οποία δείχνει τη σχέση μεταξύ δύο στοιχείων.

Εικόνα 13.10 Διάγραμμα στοιχείων.

13.6.**1.**7 **Διαγράμματα συνεργασίας**

Τα διαγράμματα συνεργασίας (collaboration diagrams) παρουσιάζουν τη ροή των μηνυμάτων μεταξύ των αντικειμένων σε μια αντικειμενοστραφή εφαρμογή. Επίσης, υποδηλώνουν τις βασικές συσχετίσεις μεταξύ των κλάσεων. Ένα ενδεικτικό παράδειγμα διαγράμματος συνεργασίας παρουσιάζεται στην Εικόνα 13.11.



Εικόνα 13.11 Διάγραμμα συνεργασίας.

13.7 Σχεδιασμός της βάσης δεδομένων

Οι βάσεις δεδομένων είναι ζωτικής σημασίας για ένα ΠΣ, είτε αυτό λειτουργεί στο διαδίκτυο είτε όχι. Ο σχεδιασμός μιας βάσης δεδομένων πρέπει να είναι προσανατολισμένος στην υλοποίηση των απαιτήσεων που έχουν τεθεί κατά την «ανάλυση των απαιτήσεων», στη σωστή οργάνωση των πληροφοριών, καθώς και στη βέλτιστη απόδοση της άντλησης των δεδομένων, όπως είναι ο χρόνος απόκρισης για την επεξεργασία ενός ερωτήματος.

Οι κύριες φάσεις του σχεδιασμού μιας βάσης δεδομένων είναι οι εξής:[5]

- *Ανάλυση απαιτήσεων*: Αναγνωρίζονται και περιγράφονται τα δεδομένα που χρησιμοποιούνται από την επιχείρηση. Επίσης, εξετάζονται οι πηγές δεδομένων που χρησιμοποιούνται από την εφαρμογή, ενώ ταυτοποιούνται και οι χαρακτηριστικοί τύποι των χρησιμοποιούμενων δεδομένων, όπως κείμενο και ημερομηνία. Επιπρόσθετα, εξετάζονται οι λειτουργίες που χρησιμοποιούν τα δεδομένα της βάσης δεδομένων, καθώς και οι αναφορές που παράγονται από τα δεδομένα αυτά.
- *Εννοιολογικός σχεδιασμός*: Καθορίζονται οι οντότητες, δηλαδή τα αντικείμενα για τα οποία διατηρούνται πληροφορίες. Κατόπιν, καθορίζονται οι ιδιότητες και τα χαρακτηριστικά των οντοτήτων, καθώς και οι μεταξύ τους συσχετίσεις.
- *Επιλογή του συστήματος*: Για παράδειγμα, γίνεται επιλογή μεταξύ των MySQL, Oracle, MS SQL, Sybase και PostgreSQL.
- *Λογικός σχεδιασμός*: Καθορίζονται ο τρόπος, δηλαδή το μοντέλο, με τον οποίο θα αντιπροσωπεύονται οι οντότητες, καθώς και οι μεταξύ τους σχέσεις στο επιλεγμένο σύστημα της βάσης δεδομένων (π.χ. σχεσιακό ή αντικειμενοστραφές).
- *Φυσικός σχεδιασμός*: Επιλέγεται η δομή του αρχείου ευρετηρίου (index file), τα μορφότυπα αποθήκευσης των δεδομένων, καθώς και η διάταξή τους στο μέσο αποθήκευσης.

- *Υλοποίηση*: Τίθεται σε λειτουργία η βάση δεδομένων.[3]

13.8 Υλοποίηση της εφαρμογής

Το επόμενο βήμα αφορά την υλοποίηση των επιμέρους διεργασιών της εφαρμογής και της βάσης δεδομένων, αφού πρώτα ελεγχθούν ξεχωριστά, αλλά και συνολικά. Επίσης, μπορεί να αφορά και την επαναξιολόγηση και βελτίωση του συστήματος. Γενικότερα, η ανατροφοδότηση πρέπει να είναι συνεχής στη διάρκεια του κύκλου ανάπτυξης του ΠΣ. Μόλις υλοποιηθεί η τελική έκδοση του συστήματος, τότε πραγματοποιείται αναλυτικός έλεγχος των διαδικασιών και των αλληλεπιδράσεων μεταξύ των διαφόρων μερών του συστήματος, μέσω δοκιμών. Σημαντικό ρόλο παίζει και η ασφάλεια ενός συστήματος, για την υλοποίηση και εξασφάλιση της οποίας πρέπει να γίνονται σχετικοί έλεγχοι.

Σημειώνεται ότι, πριν από την υλοποίηση της διαδικτυακής εφαρμογής χρειάζονται ενδεικτικά:

- ένας εξυπηρετητής, στον οποίο θα φιλοξενείται η εφαρμογή, όπως ο Glassfish,
- ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού (Integrated Development Environment/IDE), όπως το Visual Studio της Microsoft, το Eclipse ή το NetBeans,
- ένα πλαίσιο εφαρμογής υποστήριξης (back-end), για τη δημιουργία του γραφικού περιβάλλοντος, όπως το twitter bootstrap, και
- μια βάση δεδομένων, στην οποία θα αποθηκεύονται οι πληροφορίες του ΠΣ, όπως η MySQL, η Oracle ή η MS SQL, καθώς και εργαλεία διαχείρισής τους.

13.8.1 Γενικές τεχνικές για την ανάπτυξη κώδικα λογισμικού

Οι προγραμματιστές δεν πρέπει να προγραμματίζουν αυτοσχεδιάζοντας, αλλά να ακολουθούν τεχνικές ανάπτυξης κώδικα λογισμικού, έτσι ώστε να εξασφαλίζεται η ομοιομορφία του κώδικα, για να γίνεται κατανοητός από τρίτους. Θα πρέπει ο κώδικας να συνοδεύεται από επαρκή σχόλια, αλλά και να εφαρμόζεται η κατάλληλη στοίχιση, για την εύκολη ανάγνωσή του λόγω ομοιομορφίας. Προφανώς, θα πρέπει να χρησιμοποιούνται κατάλληλα κενά διαστήματος, όπως και κενές γραμμές μεταξύ των μερών του κώδικα, τα οποία να εκτελούν, για παράδειγμα, διαφορετικές λειτουργίες (τμηματοποίηση του κώδικα).

Η αναγνωσιμότητα του κώδικα υποβοηθιέται, ιδιαίτερα όταν η ονοματοδοσία των στοιχείων του υποδηλώνει τι κάνει ένα στοιχείο και όχι με ποιον τρόπο το κάνει. Γενικά, σε μια μεταβλητή θα πρέπει να δίνεται ένα αντιπροσωπευτικό ως προς τη λειτουργία της όνομα.[4]

Ως γνωστόν, κάθε κώδικας θα πρέπει να είναι όσο πιο δομημένος γίνεται. Σε αυτή την κατεύθυνση για τα ΠΣΔ κινούνται τα μοντέλα MVC, που παρουσιάζονται στο Κεφάλαιο 8 τους παρόντος.

13.8.2 Γενικές τεχνικές για τον έλεγχο του κώδικα λογισμικού

Ο έλεγχος της ορθότητας του κώδικα λογισμικού έχει πολύ μεγάλη σημασία και είναι απαραίτητος, προκειμένου να διασφαλισθεί η ποιότητά του. Συνήθως παίρνει περισσότερο χρόνο από ό,τι η ίδια η ανάπτυξη λογισμικού.

Ο έλεγχος κώδικα λογισμικού γίνεται στα παρακάτω επίπεδα:[5]

- *Έλεγχοι μονάδων* (unit/module tests): Γίνονται από τον προγραμματιστή, για τον έλεγχο της σωστής εκτέλεσης των λειτουργιών των μονάδων που έχει

---

[3] http://courses.ischool.berkeley.edu (πρόσβαση: 30-6-2015).

[4] http://www.ucl.ac.uk/~ucappgu/seminars/good-practice.pdf/ (πρόσβαση: 30-6-2015).

[5] http://blog.softfluent.com/2013/04/26/software-testing-best-practices/ (πρόσβαση: 30-6-2015).

αναπτύξει προγραμματιστικά. Συνήθως, οι λειτουργίες ελέγχονται ανεξάρτητα και με ένα περιορισμένο σύνολο των δεδομένων του λογισμικού.

- *Έλεγχοι ενοποίησης* (integration tests): Στοχεύουν στην επαλήθευση της επιτυχούς συνεργασίας μεταξύ διαφορετικών στοιχείων του λογισμικού και πραγματοποιούνται από έναν δοκιμαστή (tester) της ομάδας ανάπτυξης λογισμικού ή μια ομάδα διασφάλισης ποιότητας λογισμικού.
- *Έλεγχοι συστήματος*: Ελέγχεται η σωστή λειτουργία συγκεκριμένων περιπτώσεων χρήσης του λογισμικού, χωρίς να λαμβάνεται υπόψη η υλοποίηση ή τα εσωτερικά στοιχεία του λογισμικού.
- *Έλεγχοι αποδοχής*: Πραγματοποιούνται με τη συνδρομή χρηστών-δοκιμαστών του λογισμικού, για να επαληθευτούν η επάρκειά του στην κάλυψη των αναγκών της επιχείρησης ή του οργανισμού, καθώς και η αποδοχή του από τους μελλοντικούς του χρήστες. Ο έλεγχος του κώδικα λογισμικού μπορεί να κατηγοριοποιηθεί ως εξής, με βάση τον τύπο δοκιμών:
    - *Λειτουργικοί έλεγχοι:* Επικυρώνουν τη σωστή λειτουργία του λογισμικού, ως προς την υπηρεσία που αυτό καλείται να παρέχει.
    - *Έλεγχοι συμβατότητας*: Πραγματοποιούνται σε διαφορετικά περιβάλλοντα (π.χ. λειτουργικά συστήματα, βάσεις δεδομένων) ή αφορούν δοκιμές για συγκεκριμένη πλατφόρμα.
    - *Έλεγχοι αντοχής*: Ελέγχεται η λειτουργία του λογισμικού σε μη ιδανικές συνθήκες, δηλαδή σε συνθήκες δυσμενείς, όπως κατά τον απρόσμενο τερματισμό της λειτουργίας του φιλοξενούντος μηχανήματος ή την αποσύνδεση ενός καλωδίου δικτύου. Η κατανόηση της συμπεριφοράς του λογισμικού σε απρόσμενες συνθήκες είναι σημαντική, προκειμένου να αναπτυχθεί το πλάνο ανάκαμψής του.
    - *Έλεγχοι κλιμάκωσης*: Δοκιμάζεται το λογισμικό, για τον έλεγχο της σταθερής λειτουργίας του σε περίπτωση αύξησης του αριθμού των χρηστών του.
    - *Έλεγχοι χρηστικότητας*: Προϋποθέτει τη συμμετοχή των χρηστών, οι οποίοι γνωμοδοτούν για την τήρηση των αρχικών προδιαγραφών του λογισμικού, καθώς και για την ευκολία χρήσης του.

13.9 Αξιολόγηση ενός ΠΣΔ

Όπως αναφέρθηκε στην αρχή, η αξιολόγηση ενός ΠΣΔ ως προς την εισαγωγή του σε ένα επιχειρησιακό περιβάλλον είναι μια εκ των ων ουκ άνευ διαδικασία. Τα κριτήρια αξιολόγησης ενός ΠΣΔ είναι τα ακόλουθα:[6]

- *Επίδοση* (efficiency): Οι λειτουργίες του συστήματος πρέπει να εκτελούνται άμεσα και με την ελάχιστη δυνατή χρησιμοποίηση πόρων.
- *Αποτελεσματικότητα* (effectiveness): Το πώς οι λειτουργίες του συστήματος εκτελούνται πρέπει να συγκρίνεται με το πώς αυτές ορίζονται στην ανάλυση απαιτήσεων.
- *Ανταποκρισιμότητα* (responsiveness): Το ΠΣΔ πρέπει να έχει την αναμενόμενη ανταπόκριση από τον αγοραστή.
- *Κόστος* (cost effective): Το κόστος αγοράς του ΠΣΔ πρέπει να είναι ανάλογο με τις υπηρεσίες που προσφέρει.
- *Ωφελιμότητα* (usefulness): Το ΠΣΔ πρέπει να ικανοποιεί υπαρκτές ανάγκες, για να έχει λόγο ύπαρξης.

13.9.1 Μέθοδος ερωτηματολογίου

Για να αξιολογηθεί η ωφέλεια ενός ΠΣΔ και να διαπιστωθεί σε ποιο βαθμό ικανοποιεί τις ανάγκες των χρηστών, δηλαδή ποια είναι η χρηστικότητά του, χρησιμοποιείται ένα *ερωτηματολόγιο ευχρηστίας* (usability questionnaire),[7] το οποίο αναλύεται μέσω ενός

μαθηματικού μοντέλου, με σκοπό τη μέτρηση της αύξησης της αναμενόμενης αποδοτικότητας που προσφέρει το ΠΣΔ στους χρήστες του. Η ευχρηστία είναι μια γενική έννοια, που σχετίζεται με τη χρηστικότητα της εφαρμογής.

Κατά την ανάπτυξη εφαρμογών που προορίζονται για ευρεία χρήση και στοχεύουν να βοηθήσουν τους χρήστες να αυξήσουν την παραγωγικότητά τους, είναι σημαντικό για τους προγραμματιστές να έχουν γνώση της γνώμης των χρηστών και να διαθέτουν προγνωστικά σχετικά με αυτήν για την αναμενόμενη αύξηση της παραγωγικότητάς τους. Προκειμένου να πραγματοποιηθεί αυτή η μέτρηση, χρησιμοποιούνται ερωτηματολόγια ευχρηστίας, τα οποία είναι στοχευμένα στη διερεύνηση της πρόθεσης μιας ή περισσότερων ομάδων χρηστών να χρησιμοποιήσουν το προσφερόμενο ΠΣΔ, καθώς και στην αξιολόγηση της χρησιμότητάς του για αυτές.

13.9.2 Ισορροπημένη Κάρτα Επιδόσεων

Η *Ισορροπημένη Κάρτα Επιδόσεων* (Balanced Scorecard/BSC)[8] είναι μια άλλη μέθοδος αξιολόγησης της εισαγωγής των ΠΣ και των ΠΣΔ σε ένα επιχειρησιακό περιβάλλον, και μάλιστα ολοκληρωμένη, γιατί αφενός εξετάζει την εισαγωγή αυτή σε σχέση με τις εσωτερικές διαδικασίες της επιχείρησης, τον προσανατολισμό στον πελάτη-χρήστη και, εντέλει, το οικονομικό όφελος για την επιχείρηση, και αφετέρου χρησιμοποιεί βασικούς δείκτες απόδοσης (Key Performance Indicators/KPI). Επίσης, εκτός από μέθοδος αξιολόγησης μπορεί να λειτουργήσει και ως μέθοδος επιχειρησιακής στρατηγικής.

Η BSC στηρίζεται στην αρχή ότι οι επιχειρήσεις χρειάζονται ένα σύνολο από KPI που να αποτιμούν τους κρίσιμους παράγοντες επιτυχίας (Crucial Success Factors/CSF) της στρατηγικής της. Όπως οι CSF, έτσι και οι KPI είναι αλληλεξαρτώμενοι και ομαδοποιούνται στις εξής προοπτικές:

- της χρηματοοικονομικής,
- των πελατών,
- της εσωτερικής επιχειρησιακής διαδικασίας και
- της μάθησης και ανάπτυξης.

Οι τιμές των συγκεκριμένων δεικτών προσφέρουν πλούσια πληροφόρηση στη διοίκηση, προκειμένου αυτή να πετύχει τη στρατηγική της, μέσω της ομοφωνίας από όλα τα αρμόδια στελέχη, την επικοινωνία της στρατηγικής σε όλη την επιχείρηση, τη σύνδεση των μακροχρόνιων με τους βραχυχρόνιους στόχους, την επισκόπηση της στρατηγικής, την ευθυγράμμιση των διαφόρων συστατικών μερών της και την ανάδραση για τη βελτίωσή της.

Εδώ η έννοια της αξιολόγησης συνίσταται στην ανάπτυξη κατάλληλων συνδέσεων για την εισαγωγή του νέου ΠΣΔ, με τις εσωτερικές διαδικασίες της επιχείρησης, με τη βελτίωση της ποιότητας υπηρεσίας προς τους πελάτες της και, εντέλει, με τα καλύτερα αποτελέσματά της, μέσω κατάλληλων διασυνδεδεμένων KPI.

Βιβλιογραφικές αναφορές

[1] S. Shaffi και M. A. Obaidy, (2013), «Analysis and Comparative Study of Traditional and Web Information Systems Development Methodology (WISDM) Towards Web Development Applications», διαθέσιμο στο http://www.ijetae.com/files/Volume3Issue11/IJETAE_1113_47.pdf (πρόσβαση: 30-6-2015).

[2] P. Isaias και T. Issa (2015), «High Level Models and Methodologies for Information Systems», διαθέσιμο στο http://www.springer.com/cda/content/document/cda_downloaddocument/9781461492535-c2.pdf?SGWID=0-0-45-1479416-p175478101 (πρόσβαση: 30-6-2015).

[3] «Software Development Life Cycle», διαθέσιμο στο http://www.tutorialspoint.com/sdlc/sdlc_tutorial.pdf (πρόσβαση: 29-11-2015).

[4] «An Introduction to the Unified Modeling Language», διαθέσιμο στο http://agile.csc.ncsu.edu/SEMaterials/UMLOverview.pdf (πρόσβαση: 9-10-2015).

[5] J. Pavićević (2006), «Information System Design And Prototyping Using Form Types», διαθέσιμο στο http://homepages.mcs.vuw.ac.nz/~pmogin/ICSOFT_2006.pdf (πρόσβαση: 30-6-2015).

[6] G. Platisa και N. Balaban (2009), «Methodological Approaches to Evaluation of Information System Functionality Performances and Importance of Successfulness Factors Analysis», διαθέσιμο στο http://www.ef.uns.ac.rs/mis/archive-pdf/2009%20-%20No2/MIS2009_2_2.pdf (πρόσβαση: 30-6-2015).

[7] P. A. Kraemer και L. Kenneth (1993), «Surevy Research Methodology In Management Information Systems: An Assessment», διαθέσιμο στο https://escholarship.org/uc/item/6cs4s5f0#page-2 (πρόσβαση: 30-6-2015).

[8] M. Martinsons, R. Davison και D. Tse (1999), «The Balanced Scorecard: A Foundation for the Strategic Management of Information Systems», *Decision support systems,* τόμ. 25, τχ. 1, σ. 71- 88.

Κριτήρια αξιολόγησης

Ερώτηση 1

Ποιο είναι το πρώτο στάδιο στον κύκλο ζωής ενός λογισμικού;
Α) Η δημιουργία του λογισμικού.
Β) Η ανάλυση απαιτήσεων.
Γ) Ο σχεδιασμός αρχιτεκτονικής του προϊόντος.
Δ) Η διάθεση του προϊόντος στην αγορά.

Ερώτηση 2

Εφόσον κυκλοφορήσει στη αγορά, το λογισμικό δεν χρειάζεται επιπλέον δοκιμές.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 3

Σε ποια διαγράμματα περιλαμβάνοται οι ενεργοποιητές;
Α) Καταστάσεων.
Β) Δραστηριοτήτων.
Γ) Συνεργασίας.
Δ) Περιπτώσεων χρήσης.

Ερώτηση 4
Ποιο από τα παρακάτω αντιπροσωπεύει την κλάση στην UML;
Α) Ο ρόμβος.
Β) Το ορθογώνιο.
Γ) Ο κύκλος.

Δ) Η ευθεία γραμμή.

Ερώτηση 5

Όταν δύο στοιχεία μοιράζονται την ίδια λειτουργικότητα, ποια από τις παρακάτω είναι η μεταξύ τους σχέση;
Α) «Χρησιμοποιεί».
Β) «Μοιράζεται».
Γ) «Λειτουργεί».
Δ) «Επικοινωνεί».

Ερώτηση 6

Τα διαγράμματα δραστηριοτήτων χρησιμοποιούνται στη φάση ανάλυσης και σχεδιασμού.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 7

Το έγγραφο προσδιορισμού απαιτήσεων αφορά μόνο τους μηχανικούς λογισμικού.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 8

Με ποιο από τα παρακάτω έχει σχέση η τεχνική της ισορροπημένης κάρτας επιδόσεων;
Α) Την αξιολόγηση ενός Πληροφοριακού Συστήματος.
Β) Τη γλώσσα UML.
Γ) Το Μοντέλο του Καταρράκτη.
Δ) Την υλοποίηση ενός Πληροριακού Συστήματος.

Ερώτηση 9

Οι προγραμματιστές πραγματοποιούν την ανάλυση απαιτήσεων σε συνεχή συνεννόηση με τους χρήστες.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 10

Τα διαγράμματα δραστηριοτήτων στη UML είναι τροποποιημένη έκδοση των διαγραμμάτων καταστάσεων.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 11

Η ανάπτυξη λογισμικού αποτελεί γραμμικό μοντέλο.
Α) Σωστό.
Β) Λάθος.

Ερώτηση 12

Ποιο είναι το τελευταίο στάδιο στο Μοντέλο του Καταρράκτη;

Α) Η ενσωμάτωση.
Β) Η υλοποίηση.
Γ) Η διαχείριση και η συντήρηση.
Δ) Ο έλεγχος.

Ερώτηση 13

Σε ποιο στάδιο του Μοντέλου του Καταρράκτη αρχίζει η χρησιμοποίηση του λογισμικού;
Α) Της ενσωμάτωσης.
Β) Της υλοποίησης.
Γ) Του ελέγχου.
Δ) Της διαχείρισης και της συντήρησης.

Ερώτηση 14

Στα διαγράμματα καταστάσεων στη UML, από τι αντιπροσωπεύεται το τέλος μιας κατάστασης;
Α) Από έναν κύκλο.
Β) Από έναν κύκλο στο εσωτερικό του οποίου υπάρχει άλλος κύκλος.
Γ) Από ένα ρόμβο.
Δ) Από ένα ρόμβο στο εσωτερικό του οποίου υπάρχει άλλος ρόμβος.

Ερώτηση 15

Στην UML, σε ποιον τύπο διαγράμματος η ροή είναι από τα αριστερά προς τα δεξιά;
Α) Στο διάγραμμα ακολουθίας.
Β) Στο διάγραμμα κλάσεων.
Γ) Στο διάγραμμα δραστηριοτήτων.
Δ) Στο διάγραμμα περιπτώσεων χρήσης.

Ερώτηση 16

Στο σχεδιασμό της βάσης δεδομένων ενός Πληροφοριακού Συστήματος, πότε επιλέγονται τα μορφότυπα αποθήκευσης των δεδομένων;
Α) Κατά τον λογικό σχεδιασμό.
Β) Κατά τον φυσικό σχεδιασμό.
Γ) Κατά την ανάλυση απαιτήσεων.
Δ) Κατά την επιλογή του συστήματος της βάσης δεδομένων.

Κεφάλαιο 13

1. Β
2. Β
3. Δ
4. Β
5. Α
6. Β
7. Β
8. Α
9. Β
10. Β
11. Β
12. Γ
13. Α
14. Β
15. Α

16. B