Τ.Ε.Ι. ΠΑΤΡΩΝ – ΠΑΡΑΡΤΗΜΑ ΑΜΑΛΙΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗ ΔΙΟΙΚΗΣΗ ΚΑΙ ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: «Μοντελοποίηση και μεθοδολογική ανάλυση έργων λογισμικού»

<Modeling and methodological analysis of software projects>

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: Δρ. Ζαχαράκης Ιωάννης

ΣΠΟΥΔΑΣΤΗΣ/ΕΣ: Σκουρλής Δημήτριος, Δελαγραμμάτικα Βασιλική

<Αμαλιάδα, 13/09/2011>

Περιεχόμενα

ΕΙΚΟΝΕΣ - ΣΧΗΜΑΤΑ

ΠΙΝΑΚΕΣ

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε θερμά τον καθηγητή μας Δρ. Ι. Ζαχαράκη για τη βοήθεια και καθοδήγηση που μας προσέφερε, καθώς και για το υλικό του, το οποίο ήταν απαραίτητο για την ολοκλήρωση της πτυχιακής μας εργασίας.

Περίληψη

Στην ανάπτυξη λογισμικού εμπλέκεται ένα πλήθος προσώπων, αντικειμένων και διαδικασιών, προκειμένου να φτάσουμε στο τελικό αποτέλεσμα που είναι η δημιουργία ενός **προϊόντος λογισμικού,** το οποίο μπορεί να χρησιμοποιηθεί για να αυτοματοποιήσει κάποια παραγωγική διαδικασία ή για να λύσει κάποιο πρόβλημα με αυτόματο, ηλεκτρονικό - ψηφιακό τρόπο.

Ένα τέτοιο αντικείμενο, είναι τα **μοντέλα ανάπτυξης** και οι τεχνικές ή αλλιώς **μεθοδολογίες ανάπτυξης**. Ανάλογα με τις απαιτήσεις που έχει το έργο λογισμικού που πρόκειται να αναπτυχθεί, γίνεται και η χρήση της αντίστοιχης μεθοδολογίας και μοντέλου.

Σκοπός αυτής της πτυχιακής εργασίας είναι να καταγράψουμε και να περιγράψουμε τα πιο γνωστά και ευρέως χρησιμοποιούμενα μοντέλα και μεθοδολογίες ανάπτυξης λογισμικού, καθώς και τον τρόπο εφαρμογής τους στο περιβάλλον της επιχείρησης.

Abstract

Software development involves a number of persons, objects and processes in order to reach the final result, which is the creation of a software product that can be used to automate a process or to solve a problem with an automatic, electronic/digital way.

One such subject is the development of models and techniques or otherwise methodologies. Depending on the requirements of the project to be developed a certain methodology and model is used.

The aim of this thesis is to record and describe the most popular and widely used models and software development methodologies and their application in business environment.

Εισαγωγή

Στην ανάπτυξη λογισμικού εμπλέκεται ένα πλήθος προσώπων, αντικειμένων και διαδικασιών, προκειμένου να φτάσουμε στο τελικό αποτέλεσμα που είναι η δημιουργία ενός **προϊόντος λογισμικού,** το οποίο μπορεί να χρησιμοποιηθεί για να αυτοματοποιήσει κάποια παραγωγική διαδικασία ή για να λύσει κάποιο πρόβλημα με αυτόματο, ηλεκτρονικό - ψηφιακό τρόπο.

Ένα τέτοιο αντικείμενο, είναι τα **μοντέλα ανάπτυξης** και οι τεχνικές ή αλλιώς **μεθοδολογίες ανάπτυξης**. Ανάλογα με τις απαιτήσεις που έχει το έργο λογισμικού που πρόκειται να αναπτυχθεί, γίνεται και η χρήση της αντίστοιχης μεθοδολογίας και μοντέλου.

Στα πλαίσια αυτής της εργασίας, θα ασχοληθούμε με την καταγραφή και την παρουσίαση **μεθοδολογιών ανάπτυξης έργων λογισμικού** και **μοντέλων ανάλυσης απαιτήσεων**. Το θέμα αυτό είναι τόσο ευρύ όσο και οι πιθανές χρήσεις λογισμικού. Εμείς θα επικεντρωθούμε στην μοντελοποίηση και την μεθοδολογική ανάπτυξη έργων λογισμικού, στο πλαίσιο της επιχείρησης.

Ιδιαίτερα θα μας απασχολήσει η Ενοποιημένη Γλώσσα Μοντελοποίησης **UML**, η οποία αποτελεί ένα καθιερωμένο πρότυπο στον κύκλο ζωής ανάπτυξης λογισμικού, λόγω των ιδιαίτερων χαρακτηριστικών της που θα παρουσιασθούν παρακάτω.

Στο πρώτο κεφάλαιο, θα κάνουμε μια εισαγωγή στο θέμα, παρουσιάζοντας τον ορισμό της έννοιας του Λογισμικού, την τοποθέτησή του μέσα στις δραστηριότητες, οι οποίες πραγματοποιούνται με τη βοήθειά του, τον εντοπισμό των κυριότερων κατηγοριών προβλημάτων στην ανάπτυξη και συντήρησή του, καθώς και τον ορισμός της επιστημονικής περιοχής της Τεχνολογίας Λογισμικού.

Στο δεύτερο κεφάλαιο, θα δούμε με περισσότερη λεπτομέρεια την **Ανάλυση Απαιτήσεων**, ένα πολύ νευραλγικό κομμάτι του κύκλου ζωής ανάπτυξης λογισμικού που μπορεί να επηρεάσει θετικά ή αρνητικά, ανάλογα πόσο προσεκτική δουλειά θα έχει γίνει, όλη την αλυσίδα της επιλογής μοντέλου, της επιλογής μεθοδολογίας και εν τέλει όλη την ανάπτυξη και το τελικό αποτέλεσμα.

Στο τρίτο κεφάλαιο, θα μιλήσουμε για τα **μοντέλα κύκλου ζωής λογισμικού** Πληροφοριακών Συστημάτων, με στόχο την καλύτερη κατανόηση του συστήματος, που θέλουμε να αναπτύξουμε.

Στο τέταρτο κεφάλαιο, θα παρουσιάσουμε τις πιο γνωστές **μεθοδολογίες ανάπτυξης λογισμικού**, οι οποίες περιγράφουν τις εργασίες που πρέπει να γίνουν, κατά τη δημιουργία ενός συστήματος και θα δούμε με ποια κριτήρια επιλέγουμε την κατάλληλη μεθοδολογία, στο εκάστοτε έργο μας.

Στο πέμπτο κεφάλαιο, θα μιλήσουμε για τα **μοντέλα Πληροφοριακών Συστημάτων,** καθώς και για την γλώσσα **UML.**

Τέλος, στο έκτο κεφάλαιο, θα παρουσιάσουμε τη **μελέτη ενός ηλεκτρονικού καταστήματος**, το οποίο θα ασχολείται με την πώληση ηλεκτρονικών υπολογιστών, από την ανάλυση μέχρι και τη σχεδίασή του. Θα αναλύσουμε τη μεθοδολογία που χρησιμοποιήθηκε για την ανάπτυξη αυτού του πληροφοριακού συστήματος, θα ορίσουμε τις προδιαγραφές του και θα το παρουσιάσουμε διαγραμματικά.

Κεφάλαιο 1 Εισαγωγή στην Τεχνολογία Λογισμικού

Σκοπός του κεφαλαίου είναι ο ορισμός της έννοιας του Λογισμικού, η τοποθέτησή του μέσα στις δραστηριότητες, οι οποίες πραγματοποιούνται με τη βοήθειά του, ο εντοπισμός των κυριότερων κατηγοριών προβλημάτων στην ανάπτυξη και συντήρησή του, καθώς και ο ορισμός της επιστημονικής περιοχής της Τεχνολογίας Λογισμικού.

1.1 Ορισμός

Τεχνολογία Λογισμικού ονομάζεται η τυποποιημένη και συστηματική προσέγγιση στην ανάλυση, σχεδίαση, υλοποίηση και συντήρηση λογισμικού ηλεκτρονικών υπολογιστών (Βεσκούκης, 2000).

Σήμερα θεωρείται διακριτό επάγγελμα, στο πλαίσιο των τεχνολογιών των πληροφοριών, η επιστημονική βάση του οποίου αντιμετωπίζεται ως ξεχωριστός κλάδος της πληροφορικής, σε στενή όμως σχέση με την επιστήμη υπολογιστών και τα υπολογιστικά συστήματα.

Η τεχνολογία λογισμικού προέκυψε κατά τη δεκαετία του 1960 μέσω της αναγνώρισης των περιορισμών της αντιμετώπισης του λογισμικού, μόνο ως αποτέλεσμα προγραμματισμού, ενώ έχει προταθεί η συμπερίληψή της και στις επιστήμες μηχανικών, ως ξεχωριστού γνωστικού πεδίου (Sommerville, 2001).

1.2 Ιστορία και εξέλιξη της Τεχνολογίας Λογισμικού

Μέχρι το 1969, η ανάπτυξη του λογισμικού δεν υπόκειντο σε κάποιες βασικές αρχές. Ο όρος Software Engineering χρησιμοποιήθηκε από το NATO σε κάποιες μελέτες του 1968 και 1969 (Booch G., 1999).

Μεταξύ του 1969 και του 1971 εδραιώνονται οι πρώτες πρακτικές στον προγραμματισμό, όπως το **top-down design** και το **modularity**. Επίσης αναπτύσσονται νέες γλώσσες προγραμματισμού (Pascal), ενώ δημιουργείται και η PSL (Problem Statement Language) για να καταγράψει τις απαιτήσεις των χρηστών(Γερογιάννης Β., 2006).

Το 1972, ο δομημένος προγραμματισμός και τα προγραμματιστικά στυλ αναπτύσσονται, ενώ γίνεται λόγος για πρώτη φορά για τον κύκλο ζωής του λογισμικού και για τρόπους διαχείρισης του.

Το 1974, η αξιοπιστία και η επιβεβαίωση για ποιότητα στο λογισμικό έδωσε ώθηση στην ανάπτυξη ενός συστηματικού τρόπου για την αξιολόγηση του. Εμφανίζεται όλο και περισσότερο η ανάγκη για τον προσδιορισμό του κόστους ανάπτυξης του λογισμικού.

Το 1976, αναπτύσσονται εργαλεία για τον προσδιορισμό των απαιτήσεων (requirements), των προδιαγραφών (specifications) και για το σχεδιασμό του λογισμικού. Υπήρξε νέο ενδιαφέρον για τον καλό σχεδιασμό πριν την υλοποίηση σε κώδικα.

Την δεκαετία του ‘80 αρχίζουν να εμφανίζονται αυτόματα εργαλεία για την κάθε φάση ανάπτυξης και ζωής του λογισμικού, τα λεγόμενα εργαλεία τύπου CASE (Computer Aided Software Engineering).

Από την δεκαετία του ‘90 μέχρι σήμερα έχουμε την χρήση εκτεταμένων τεχνικών της Τεχνολογίας Λογισμικού (Software engineering) και αυτόματων εργαλείων για την ανάπτυξη λογισμικού (Βεσκούκης, 2000).

Η κατασκευή λογισμικού που να είναι σύμφωνο με τις απαιτήσεις να κάνει τη δουλειά για την οποία προορίζεται και να την κάνει σωστά, να κατασκευάζεται εντός χρονοδιαγράμματος και προϋπολογισμού, να μην περιέχει σφάλματα, να συντηρείται εύκολα, αλλά και να είναι επίκαιρο και ανταγωνιστικό ως προϊόν, είναι η πρόκληση με την οποία βρίσκονται αντιμέτωποι οι κατασκευαστές λογισμικού και σήμερα.

Όπως είναι αναμενόμενο, πολλά από τα χαρακτηριστικά της διαδικασίας ανάπτυξης λογισμικού να πρέπει να εξελίσσονται συνεχώς ώστε να επιτυγχάνεται το επιθυμητό αποτέλεσμα.

Η Τεχνολογία Λογισμικού οφείλει να παρέχει την απαιτούμενη υποστήριξη και τεκμηρίωση, ώστε η κατασκευή του λογισμικού να είναι μια ιδιαίτερα ευέλικτη και προσαρμόσιμη διαδικασία.

Ορισμένες από τις εξελίξεις που έπαιξαν σημαντικό ρόλο στη θεμελίωση της ανάγκης νέων προσεγγίσεων στην ανάπτυξη του λογισμικού, είναι οι ακόλουθες (Σκορδαλάκης, 1991):

- Η προσήλωση των κατασκευαστών σε αυστηρές διαδικασίες και πρότυπα που είτε ήταν αποτέλεσμα προσπάθειας σύγκλισης υπαρχουσών πρακτικών, είτε είχαν μικρή σχέση με τους κατασκευαστές λογισμικού, αποδείχθηκε αναποτελεσματική. Στην πράξη, οι παραδοσιακές προσεγγίσεις ξεπεράστηκαν ως απλά ανεπαρκείς, μιας και η ανάπτυξη του λογισμικού σπάνια γινόταν «κατά γράμμα». Από την άλλη πλευρά, η ανεπάρκεια που μόλις αναφέρθηκε, συχνά γεννούσε το αντίθετο φαινόμενο, δηλαδή την πλήρη «αναρχία» στην ανάπτυξη λογισμικού. Και πάλι όμως, η άναρχη ανάπτυξη οδηγούσε (και οδηγεί) με προδιαγεγραμμένη ακρίβεια σε αποτυχημένα έργα ανάπτυξης λογισμικού.

- Ο ρυθμός με τον οποίο αναπτύχθηκε η τεχνολογία του υλικού των υπολογιστών ξεπέρασε κάθε πρόβλεψη, δίνοντας στους χρήστες εργαλεία και δυνατότητες αξιοποίησης και νέες περιοχές εφαρμογών.

- Η ανάδειξη του Internet, το οποίο αρχικά δημιουργήθηκε για στρατιωτικούς σκοπούς στις ΗΠΑ, στο νέο μέσο επικοινωνίας του 21ου αιώνα, δημιούργησε μια νέα πλατφόρμα πάνω στην οποία τρέχει το λογισμικό. Τα σημαντικότερα χαρακτηριστικά αυτής είναι η ανομοιογένεια, η μεγάλη γεωγραφική κατανομή, η μη ασφαλής επικοινωνία, αλλά και η εκρηκτικά αυξανόμενη ζήτηση για υπηρεσίες και εφαρμογές λογισμικού που εκτελούνται πάνω σ' αυτή τη νέα πλατφόρμα.

- Η σύγκλιση, μετά από αρκετό καιρό, των επικρατέστερων από τις νέες αντικειμενοστραφείς προσεγγίσεις ανάπτυξης λογισμικού και η επαφή τους με τη βιομηχανική πρακτική.

- Τα μοντέλα κύκλου ζωής που παρουσιάζονται σε επόμενη ενότητα έχουν γίνει ευέλικτα και παραμετρικά, δηλαδή προσαρμόζονται στα χαρακτηριστικά κάθε συγκεκριμένου περιβάλλοντος ανάπτυξης λογισμικού, αλλά και του θεματικού πεδίου της υπό ανάπτυξη εφαρμογής (application domain).

- Η τεκμηρίωση του λογισμικού, δηλαδή το σύνολο των προϊόντων που περιγράφουν το λογισμικό και παράγονται σ' όλες τις φάσεις του κύκλου ζωής αυτού, δεν ακολουθεί αυστηρά πρότυπα δομής, όπως στο παρελθόν, αλλά προσαρμόζεται στις εκάστοτε συνθήκες ανάπτυξης, τα εργαλεία που χρησιμοποιούνται και την υπάρχουσα εμπειρία.

- Τα περιβάλλοντα ανάπτυξης (γλώσσες προγραμματισμού, βιβλιοθήκες συστατικών λογισμικού, εργαλεία συγγραφής κώδικα, εργαλεία ελέγχου προγράμματος, βιβλιοθήκες αναφοράς) έχουν γίνει ιδιαίτερα σύνθετα και ολοκληρωμένα, υποστηρίζοντας τον προγραμματιστή σε περισσότερες εργασίες απ' ότι στο παρελθόν.

Με την ωρίμανση της τεχνολογίας δημιουργήθηκαν οι συνθήκες για την ανάπτυξη ενός προτύπου συμβολισμών των σχετικών με την αντικειμενοστραφή τεχνολογία εννοιών.

Τα πρώτα βήματα έγιναν στις αρχές της δεκαετίας του 1990, από τους συγγραφείς των τριών επικρατέστερων αντικειμενοστραφών προσεγγίσεων ανάπτυξης λογισμικού. Αναφερόμαστε στους Booch, Jacobson και Rumbaugh (Kenneth C. Laudon).

Αργότερα, αναγνωρίζοντας τη δυναμική που δημιουργείτο, συνέπραξαν και μεγάλοι κατασκευαστές λογισμικού και εργαλείων, όπως η Digital Equipment, η Hewlett Packard, η Oracle, η Texas Instruments, η Unisys, η MCI, και η Microsoft.

Τελικώς, καταλήξαμε στην περιγραφή ενός προτύπου συμβολισμών, το οποίο ονομάστηκε Unified Modelling Language και εν συντομία UML. Το πρότυπο, στην τελευταία του έκδοση υιοθετήθηκε από τον οργανισμό OMG (Object Management Group), ως βιομηχανικό πρότυπο παράστασης λογισμικού.

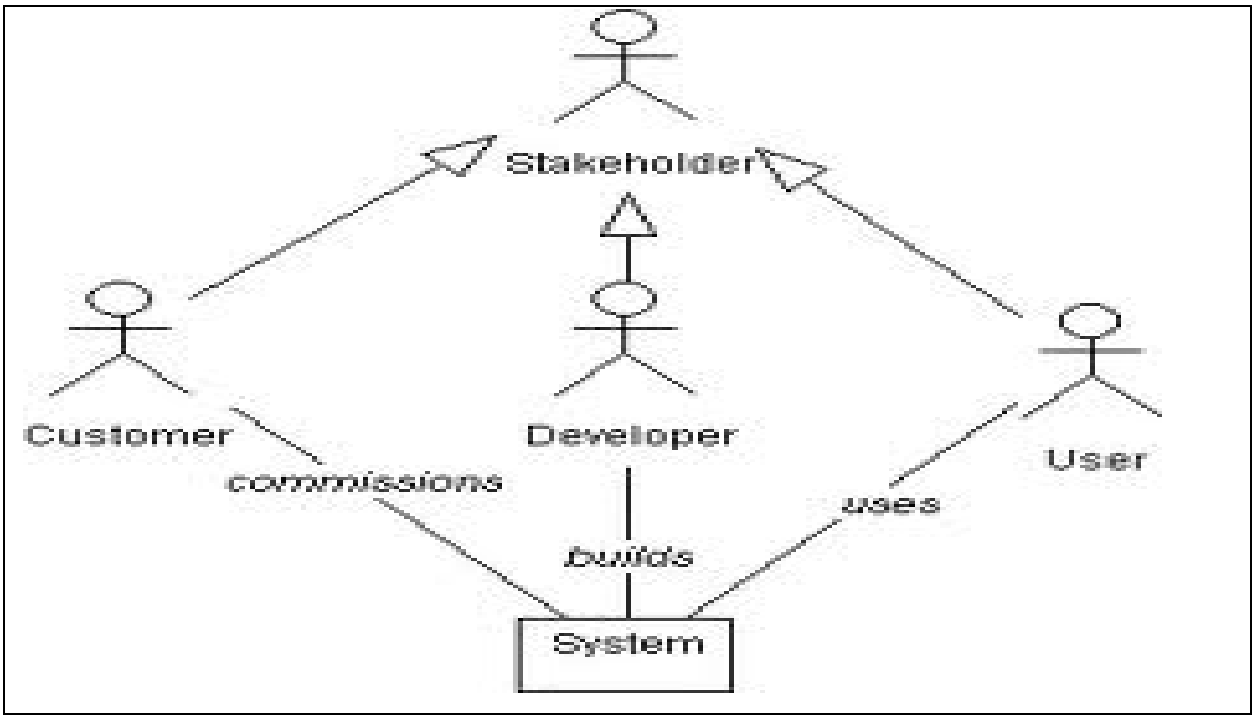
1.3 Στάδια ανάπτυξης λογισμικού

Οι γενικές φάσεις που αναγνωρίζονται, είναι αυτές της σύλληψης της **ιδέας**, της **επεξεργασίας** της λύσης, της **κατασκευής** και της **μετάβασης**.

Κάθε γενική φάση αναλύεται σε κύκλους ανάπτυξης και σε κάθε κύκλο ανάπτυξης λαμβάνουν χώρα οι εργασίες της προδιαγραφής των απαιτήσεων, της ανάλυσης, της σχεδίασης, της υλοποίησης και του ελέγχου.

Στο κεφάλαιο αυτό, θα ασχοληθούμε με την περιγραφή των εργασιών της ανάλυσης των απαιτήσεων και των προδιαγραφών λογισμικού.

Κάθε κύκλος ανάπτυξης αφορά ένα υποσύνολο του λογισμικού υπό ανάπτυξη, το οποίο οριοθετείται είτε από το πλήθος των λειτουργιών που περιλαμβάνει, είτε από το βαθμό λεπτομέρειας με τον οποίο τις αντιμετωπίζει. Στο τέλος κάθε κύκλου ανάπτυξης έχουμε μια εκδοχή (release) του λογισμικού.



Εικόνα 1. Τα πρόσωπα που εμπλέκονται στην ανάπτυξη και η μεταξύ τους αλληλεπίδραση.

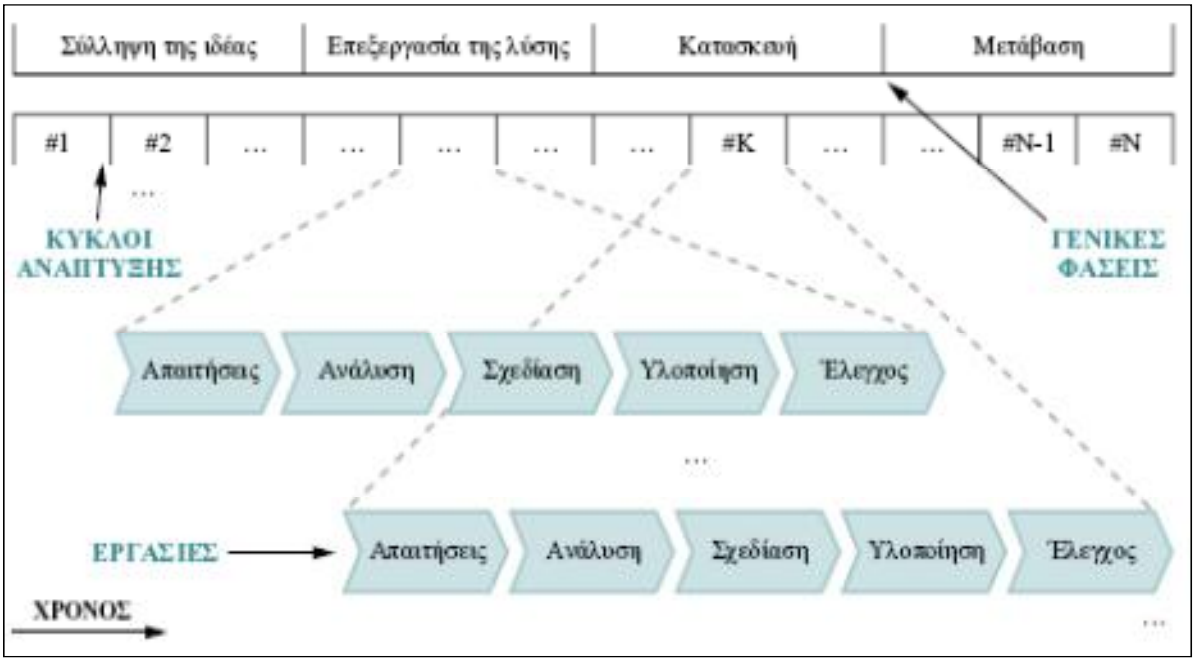
Κατά τις πρώτες δύο φάσεις (σύλληψη της ιδέας και επεξεργασία της λύσης) η εκδοχή αυτή είναι ένα σύνολο από κείμενα και μοντέλα παράστασης λογισμικού, ενώ κατά τη φάση της κατασκευής η εκδοχή αυτή περιλαμβάνει πηγαίο και εκτελέσιμο κώδικα με ολοένα και περισσότερα από τα απαιτούμενα λειτουργικά χαρακτηριστικά του λογισμικού.

Τέλος, κατά τη φάση της μετάβασης το λογισμικό τοποθετείται σε δοκιμαστική λειτουργία, όπου επαληθεύεται η ικανοποίηση των απαιτήσεων των χρηστών και γίνονται οι απαραίτητες διορθώσεις. Ας σημειωθεί ότι δεν εκτελούνται όλες οι εργασίες (προδιαγραφή, ανάλυση, σχεδίαση, υλοποίηση, έλεγχος) σ' όλες τις γενικές φάσεις.

Για παράδειγμα, σύμφωνα με όσα γνωρίζουμε από τη δομημένη ανάλυση και σχεδίαση, δεν είναι δυνατό κατά τη φάση της σύλληψης της ιδέας να εκτελεστεί οποιαδήποτε εργασία σχεδίασης λογισμικού.

Αυτό είναι εν μέρει μόνο αληθές στην περίπτωση της αντικειμενοστρεφούς τεχνολογίας, όπου θα δούμε ότι σχετικά νωρίς στην ανάπτυξη εισάγονται έννοιες όπως «κλάση» και «συνεργασία», οι οποίες τελικά καταλήγουν να είναι πραγματικά κλάσεις και στο πεδίο της υλοποίησης.

Ενδέχεται, ασφαλώς, οι οντότητες που εντοπίζονται στις πρώτες φάσεις ανάπτυξης του λογισμικού να μην απεικονιστούν στη συνέχεια αυτούσιες σε κλάσεις. Αυτό, ωστόσο, δεν αναιρεί το γεγονός ότι από πολύ νωρίς στην αντικειμενοστρεφή ανάπτυξη εισάγονται όροι συστατικών στοιχείων υλοποίησης του λογισμικού.



Εικόνα 2. Φάσεις ανάπτυξης λογισμικού.

Έχοντας κατά νου τη σαφή διάκριση ανάλυσης και σχεδίασης η οποία ισχύει στη δομημένη προσέγγιση, παρατηρούμε ότι στην αντικειμενοστρεφή εκτελούμε νωρίς εργασίες που μπορούν να χαρακτηριστούν και ως εργασίες ανάλυσης και ως εργασίες σχεδίασης. Στην εικόνα 2 μπορούμε να δούμε τις γενικές φάσεις ανάπτυξης του λογισμικού.

1.4 Τεχνολογία Λογισμικού και επιχειρηματικό περιβάλλον

Το τελικό αποτέλεσμα της διαδικασίας ανάπτυξης λογισμικού, είναι συνήθως η δημιουργία ενός Πληροφοριακού Συστήματος. Οι επιχειρήσεις χρησιμοποιούν ολοένα και σε μεγαλύτερο βαθμό πληροφοριακά συστήματα και γενικότερα λογισμικό για να ανταποκριθούν στις ανάγκες της παραγωγής.

Τα πληροφοριακά συστήματα συλλέγουν, αποθηκεύουν, μεταδίδουν και επεξεργάζονται δεδομένα για την παροχή χρήσιμων, ολοκληρωμένων και έγκαιρων πληροφοριών όπου είναι αυτό απαραίτητο. Τα πληροφοριακά συστήματα χρησιμοποιούνται από τις επιχειρήσεις:

- Για ταχύτατη και ακριβή επεξεργασία των δεδομένων.
- Λόγω μεγάλης αποθηκευτικής ικανότητας.
- Ταχύτατη επικοινωνία μεταξύ τοποθεσιών.
- Άμεση πρόσβαση σε πληροφορίες που πρέπει να αντλήσει η επιχείρηση για την δραστηριότητά της.
- Λόγω δυνατότητας συντονισμού ατόμων, ομάδων και οργανισμών.
- Για την υποστήριξη των αποφάσεων που θα ληφθούν από την επιχείρηση.
- Για αυτοματοποίηση και βελτίωση των διαδικασιών και των ροών εργασιών.
- Για καλύτερη αξιοποίηση των πολύτιμων δεδομένων της επιχείρησης.
- Για την αύξηση της αποτελεσματικότητας της επιχείρησης.

Υπάρχουν αρκετοί παράγοντες και εμπλεκόμενοι φορείς με τα πληροφοριακά συστήματα, όπως οι χρήστες αυτών, οι υπεύθυνοι λειτουργίας και ανάπτυξής τους, το απαραίτητο υλικό για την ύπαρξη και υποστήριξη των συστημάτων αυτών, όπως επίσης και διάφοροι εξωτερικοί παράγοντες που μπορούν να επηρεάσουν τα συστήματα αυτά.

Με την εγκατάσταση πληροφοριακών συστημάτων δημιουργήθηκαν αυτόματα και πολλές σχετικές (απαραίτητες για τη σωστή λειτουργία τους) θέσεις εργασίας, όπως:

- Διευθυντής Πληροφορικής (Chief Information Officer).
- Διευθυντής Μηχανογράφησης (IT Manager).
- Προϊστάμενος Τμήματος Μηχανογράφησης (IT Supervisor).
- Υπεύθυνος Λογαριασμών & Εφαρμογών (Administrator).
- Υπεύθυνος Εξυπηρετητών (Servers Manager).
- Υπεύθυνος Δικτύου (Network Manager).
- Υπεύθυνος Τεχνικής Υποστήριξης (Technician)
- Διάφοροι Αναλυτές, Σχεδιαστές και Προγραμματιστές, Βιβλιοθηκάριοι.

Υπάρχουν πολλά είδη πληροφοριακών συστημάτων που μπορούν να χρησιμοποιηθούν ανάλογα με τις ανάγκες και τις οικονομικές δυνατότητες της επιχείρησης. Τα σημαντικότερα πληροφοριακά συστήματα είναι τα εξής:

- SCMS (Supplier and Contract Management System / Συστήματα Διαχείρισης Αλυσίδας Εφοδιασμού).
- KMS (Knowledge Management Systems / Συστήματα Διαχείρισης Γνώσης).
- OAS (Office Automation Systems / Συστήματα Αυτοματοποίησης Γραφείου).
- TPS (Transaction Processing Systems / Συστήματα Επεξεργασίας Συναλλαγών).
- ERP (Enterprise resource planning / Συστήματα Ενδοεπιχειρησιακού Σχεδιασμού).
- ESS (Executive Support Systems/ Συστήματα Υποστήριξης Διοίκησης).

- DSS (Decision Support Systems / Συστήματα Υποστήριξης Απόφασης).
- MIS (Management Information Systems / Διοικητικά Συστήματα Πληροφόρησης).

Το ποιο ή ποια από τα παραπάνω πληροφοριακά συστήματα θα επιλέξει η επιχείρηση εξαρτάται από αρκετούς παράγοντες. Υπάρχουν θετικά αλλά και αρνητικά για το καθένα σύστημα, ανάλογα βέβαια την επιχείρηση.

1.5 Τεχνολογία Λογισμικού – Παρόν και Μέλλον

Τα πρώτα βήματα της Τεχνολογίας Λογισμικού έγιναν μέσα σ' ένα περιβάλλον, όπου ο ρυθμός των αλλαγών που συντελούνταν στην τεχνολογία των υπολογιστών, αλλά και σε όλους τους τομείς των ανθρώπινων δραστηριοτήτων όπου αυτοί χρησιμοποιούνται, δεν ήταν συνειδητός, τουλάχιστον όχι στο βαθμό που είναι σήμερα.

Οι πρωτοπόροι στην έρευνα και στην πρακτική της ανάπτυξης λογισμικού αναζητούσαν τον καλύτερο τρόπο να κατασκευάζεται και να συντηρείται λογισμικό, χωρίς να γνωρίζουν ότι η έννοια «καλύτερο» ήταν σχετική, χωρίς να μπορούν να φανταστούν πόσο γρήγορα θα μεταβαλλόταν η άποψη, οι απαιτήσεις και τελικά η συνείδηση των χρηστών των υπολογιστικών συστημάτων, αλλά και χωρίς να μπορούν να συλλάβουν ποια μορφή θα έχουν οι υπολογιστές στο μέλλον.

Σήμερα, γνωρίζουμε ότι οι υπολογιστές και κατά συνέπεια το λογισμικό βρίσκονται πρακτικά παντού και μας υποστηρίζουν σε πολλές από τις δραστηριότητές μας. Η αίσθηση και η μορφή του λογισμικού έχει εξελιχθεί.

Η αλληλεπίδραση με λογισμικό, την οποία βιώνουμε σήμερα, δε γίνεται με λίγες και μεγάλες εφαρμογές, αλλά με πολλές και μικρές, οι οποίες βρίσκονται κατανεμημένες οπουδήποτε σε κάποιο δίκτυο, όπου πρόκειται να συγκλίνουν στο κοντινό μέλλον όλα τα πληροφοριακά και επικοινωνιακά συστήματα.

Ως εκ τούτου, ο τρόπος και η φιλοσοφία της ανάπτυξης λογισμικού ήταν αναγκαίο να εξελιχθούν, ώστε να ικανοποιούνται οι σύγχρονες απαιτήσεις.

Το λογισμικό μεταβάλλεται με ραγδαίους ρυθμούς. Εκτελείται, πλέον, πάνω σε δίκτυο και γίνεται ολοένα και περισσότερο φιλικό αλλά και απαραίτητο σε πολλές πλευρές της ζωής μας, με όλα τα θετικά και τα αρνητικά που αυτό συνεπάγεται.

Όλα αυτά επιφέρουν, ασφαλώς, αύξηση της πολυπλοκότητας του λογισμικού, το οποίο πλέον λειτουργεί σε ανομοιογενή και κατανεμημένα περιβάλλοντα. Η Τεχνολογία Λογισμικού βρίσκεται μπροστά σε μια νέα πραγματικότητα και καλείται να προσαρμόσει τις προσεγγίσεις της στις νέες αυτές συνθήκες.

Κεφάλαιο 2 Ανάλυση απαιτήσεων

Στο κεφάλαιο αυτό θα επικεντρωθούμε στην φάση της ανάλυσης των απαιτήσεων, περιγράφοντας συνοπτικά τον σχεδιασμό και την υλοποίηση ενός Πληροφοριακού συστήματος.

2.1 Γενικά

Η ανάλυση απαιτήσεων είναι μια διαδικασία κατάρτισης μιας λίστας, όπου αναφέρονται οι προδιαγραφές που πρέπει να πληροί το προϊόν που πρόκειται να δημιουργηθεί ή να εγκατασταθεί (Βεσκούκης, 2000).

Το προϊόν μπορεί να είναι μια νέα εφαρμογή που αναπτύσσεται κατά παραγγελία, η εγκατάσταση ενός έτοιμου προγράμματος, η δημιουργία ενός δικτυακού τόπου, ηλεκτρονικού καταστήματος και πολλά άλλα. Οι προδιαγραφές που προσδιορίζονται μπορεί να είναι τεχνολογικές, επιχειρηματικές, λειτουργικές, να σχετίζονται με τη μορφή, το κόστος, τη διάρκεια, το χρόνο απόσβεσης και άλλα.

Η λίστα που καταρτίζεται χρησιμεύει τόσο σε αυτούς που θα εγκαταστήσουν ή θα αναπτύξουν τη λύση λογισμικού όσο και σ' εκείνους που θα τη χρησιμοποιήσουν (χρήστες, πελάτες) και εφαρμόζεται σε μεγάλα αλλά και μικρά έργα πληροφορικής. Η μόνη διάφορα τους είναι ότι στα μικρά έργα το ανθρώπινο δυναμικό που απασχολείται είναι αισθητά μειωμένο, έναντι των εργαζομένων που λαμβάνουν μέρος στην ανάλυση απαιτήσεων μεγάλων έργων.

Πέραν αυτών, η ανάλυση απαιτήσεων είναι μία συνεργατική διαδικασία όπου διαφορετικά άτομα με διαφορετικές αφετηρίες συναντιούνται, αλληλεπιδρούν, διαφωνούν και συμφωνούν γύρω από την ίδια αντικειμενική σφαίρα: το έργο. Ο πελάτης ζητά κάτι στο οποίο ο προγραμματιστής αναλαμβάνει να δώσει μορφή. Ο αναλυτής βοηθά τον προγραμματιστή να δημιουργήσει κάτι εύκολο και φιλικό στη χρήση. Ο project manager εποπτεύει το έργο εξ αρχής μέχρι την ολοκλήρωσή του και ούτω καθ’ εξής.

Συχνά προγραμματιστές, αναλυτές, πελάτες και δυνητικοί χρήστες υποτιμούν την ανάλυση απαιτήσεων και δεν της αποδίδουν τη δέουσα σημασία. Αυτό έχει ως αποτέλεσμα τη δημιουργία έργων που χρήζουν βελτιώσεων ή αλλαγών, γιατί δεν ανταποκρίνονται στους στόχους και τις επιδιώξεις που είχαν αρχικά τεθεί.

Ωστόσο, οι βελτιώσεις και αλλαγές στα έργα πληροφορικής είναι ιδιαίτερα δαπανηρές, δύσκολες και χρονοβόρες, ενώ η πραγματοποίησή τους μπορεί να απαιτήσει εκ θεμελίων αναδημιουργία. Για το λόγο αυτό, η ανάλυση απαιτήσεων θεωρείται εκ των ουκ άνευ συστατικό για επιτυχημένη υλοποίηση εφαρμογών.

Για την κατανόηση, τον προσδιορισμό και την έκφραση των απαιτήσεων από το λογισμικό είναι απαραίτητο ένα ιδεατό μοντέλο (conceptual model) των διεργασιών του συστήματος στο οποίο θα λειτουργήσει το λογισμικό. Τα μοντέλα αυτά χρησιμοποιούν τις παρακάτω τεχνικές παράστασης:

- Ροή δεδομένων (data flow)
- Μηχανή πεπερασμένων καταστάσεων (finit estate machine)
- Επικοινωνούσες ταυτόχρονες διεργασίες (communicating concurrent processes)
- Μοντέλο οντοτήτων σχέσεων (entity relationship models)
- Εξομοίωση (simulation)
- Λειτουργική σύνθεση (functional composition)

Η δομημένη ανάλυση (structured analysis) χρησιμοποιεί ως βάση το μοντέλο ροής δεδομένων και μια σειρά από εξειδικευμένους συμβολισμούς για την καταγραφή των απαιτήσεων:

- το διάγραμμα ροής δεδομένων (data flow diagram),
- τα δομημένα αγγλικά (structured english),
- το λεξικό δεδομένων (data dictionary) και

- το διάγραμμα οντοτήτων συσχετίσεων (entity relationship diagram)

Εκτός από τη δομημένη ανάλυση, η οποία ταιριάζει με το διαδικασιακό παράδειγμα προγραμματισμού, όπως υποστηρίζεται από τις γλώσσες C, Pascal, Fortran κλπ, στις μέρες μας χρησιμοποιείται συχνά και η αντικειμενοστρεφής ανάλυση (object oriented analysis) σε συνδυασμό με αντικειμενοστρεφείς γλώσσες όπως η C++, Java, Eiffel, Smalltalk κλπ.

2.2 Οφέλη

Οι σημαντικότερες ωφέλειες που απορρέουν από τη χρήση της ανάλυσης των απαιτήσεων είναι οργανωτικές, λειτουργικές και οικονομικές, με αυτήν ακριβώς τη σειρά, όχι βάσει σπουδαιότητας αλλά χρονικής ακολουθίας. Η ανάλυση απαιτήσεων συντελεί στην καλή οργάνωση και εκτέλεση του έργου, που με τη σειρά τους εξασφαλίζουν τη λειτουργικότητά του για όλες τις εμπλεκόμενες πλευρές. Στο τέλος, τα οφέλη αυτά έχουν άμεσο αντίκρισμα στη μείωση του κόστους, τόσο για την επιχείρηση που υλοποιεί το έργο όσο και για τον πελάτη που θα το χρησιμοποιήσει.

Η ανάλυση απαιτήσεων, για τον σχεδιασμό και την ανάπτυξη ενός πληροφοριακού συστήματος, προσφέρει στο σύνολο των εμπλεκομένων (προγραμματιστών, αναλυτών, διευθυντών, πελατών, χρηστών) σημαντική ευκαιρία για καλύτερη οργάνωση, προγραμματισμό και διαχείριση του έργου. Τους φέρνει σε επαφή με την πραγματική φύση του έργου, φωτίζει αθέατες πλευρές και αδυναμίες, εντοπίζει τις ελλείψεις, προφυλάσσει και προειδοποιεί.

Αναλυτικότερα, ο διευθυντής του έργου γνωρίζει ποια πρέπει να είναι η τελική του μορφή και κατευθύνει τους υφισταμένους του βάσει συγκεκριμένου σχεδίου. Οι προγραμματιστές και οι αναλυτές γνωρίζουν εκ των προτέρων τι ακριβώς πρέπει να δημιουργήσουν, τι και υπό ποιες συνθήκες να δοκιμάσουν (τεστάρουν). Οι πελάτες είναι προετοιμασμένοι για το τι θα παραλάβουν. Εν ολίγοις, η εξέλιξη του έργου είναι -σε μεγάλο βαθμό- προβλέψιμη και οι δυσάρεστες εκπλήξεις περιορίζονται αισθητά.

Επιπλέον, η ανάλυση απαιτήσεων συμπιέζει σημαντικά το κόστος υλοποίησης του έργου, καθώς εξασφαλίζει ότι η ολοκλήρωσή του θα γίνει βάσει χρονοδιαγράμματος, πλαισίου και

συγκεκριμένων προδιαγραφών. Αυτό έχει μεγάλη αξία, γιατί -όχι σπάνια- σε έργα για τα οποία δεν έχει προηγηθεί ανάλυση απαιτήσεων παρατηρούνται φαινόμενα όπως:

α) Το πρόγραμμα αποδεικνύεται ότι έχει αρκετά ελαττώματα (bugs), τα οποία πρέπει να διορθωθούν με τη συγγραφή νέου κώδικα.

β) Η εφαρμογή που δημιουργήθηκε αποκλειστικά για ένα συγκεκριμένο πελάτη αποδεικνύεται ελλιπής στην πράξη και χρήζει βελτίωσης.

γ) Ο πελάτης συνειδητοποίησε -την τελευταία στιγμή- ότι θέλει το πληροφοριακό σύστημα που έχει ζητήσει να υλοποιηθεί να περιέχει επιπλέον γνωρίσματα ή μπορεί να μην θέλει ένα γνώρισμα που ήθελε σε προηγούμενη φάση.

Ασφαλώς, σε αυτά τα τρία παραδείγματα θα μπορούσαν να προστεθούν και πολλά άλλα. Αυτό στο οποίο οφείλουμε να εστιάσουμε είναι ότι η προσθήκη, η βελτίωση, η "επισκευή" σε ένα έργο πληροφορικής αποτελούν ιδιαίτερα κοστοβόρα και χρονοβόρα υπόθεση. Δεν είναι λίγες οι φορές που η προσθήκη κάποιας νέας εφαρμογής απαιτεί την εκ θεμελίων αναδόμηση και ανασυγκρότηση του προγράμματος, γεγονός που ισοδυναμεί με πολλές εργατοώρες και ασφαλώς κόστος.

Για το ζήτημα αυτό διάφορες έρευνες έχουν επισημάνει ότι κάθε εργατοώρα που επενδύεται στην ανάλυση απαιτήσεων εξοικονομεί δεκάδες εργατοώρες για μετέπειτα εργασίες βελτίωσης και αναπροσαρμογής.

2.3 Συνοπτική ανάλυση απαιτήσεων για ένα Πληροφοριακό Σύστημα

Ας υποθέσουμε ότι ένας πελάτης αναθέτει στην εταιρεία μας το σχεδιασμό και την ανάπτυξη ενός συστήματος Ηλεκτρονικού Εμπορίου (e-commerce), μέσω του οποίου θα μπορεί να πωλεί τα προϊόντα που παράγει η δική του επιχείρηση.

Η ανάλυση απαιτήσεων θα πρέπει να προηγείται της δημιουργίας δικτυακού τόπου, ειδικά όταν αυτός προορίζεται για την άσκηση ηλεκτρονικού εμπορίου. Ο ορθολογικός σχεδιασμός, η

μακρόπνοη στρατηγική και η ορθή εκτίμηση των προοπτικών θα συμβάλουν στη δημιουργία ενός site που θα έχει όλα τα εχέγγυα για να θεωρείται επιτυχημένο.

Ας δούμε αναλυτικά πώς μπορεί να διαρθρωθεί η ανάλυση των απαιτήσεων για ένα ηλεκτρονικό κατάστημα: το καλύτερο γι' αυτόν το σκοπό είναι να συνταχθεί μία λίστα με ερωτήματα, στα οποία θα πρέπει να δοθούν απαντήσεις.

Στην αρχή θα πρέπει να τεθούν προς απάντηση ορισμένα επιχειρηματικά ζητήματα. Μεταξύ άλλων, ποιο ή ποια είναι τα προϊόντα που θέλουμε να εμπορευόμαστε, αν υπάρχει ανταγωνισμός στα συγκεκριμένα προϊόντα και τι επιπέδου, ποιο είναι το στοχευμένο κοινό (target group), στο οποίο θα απευθυνθούμε, τι ποσό μπορούμε να επενδύσουμε για την ανάπτυξη του καταστήματος, τι μεθόδους διανομής και πληρωμών θα ακολουθήσουμε και ούτω καθεξής.

Επιπλέον, ποιος είναι ο μέγιστος αριθμός πελατών που θα μπορούν να εξυπηρετούνται ηλεκτρονικά ταυτόχρονα; Τι μέτρα θα λάβουμε για την ασφάλεια των συναλλαγών και των προσωπικών δεδομένων;

Στη συνέχεια θα πρέπει να τεθούν τεχνικά ζητήματα, σε συνεννόηση με κάποιον τεχνικό. Για παράδειγμα: θα επιλεγεί κάποια προκατασκευασμένη πλατφόρμα ή το ηλεκτρονικό κατάστημα θα αναπτυχθεί εκ του μηδενός; Αν επιλεγεί προκατασκευασμένη πλατφόρμα θα προτιμηθεί κλειστού ή ανοιχτού κώδικα και γιατί; Πού θα φιλοξενηθεί (hosting) το ηλεκτρονικό κατάστημα; Θα ενοικιάσουμε χώρο σε δικό μας διακομιστή ή σε κάποιον πάροχο; Ποιες ακριβώς λειτουργίες θέλουμε να επιτελεί και τι δυνατότητες θέλουμε να προσφέρει στους χρήστες; Ποια επιθυμούμε να είναι η δομή του; Σε ποιους φυλλομετρητές (browsers) επιθυμεί ο πελάτης να είναι καλύτερα προσβάσιμο; Επιθυμεί ο πελάτης να συνεργάζεται με κάποια βάση δεδομένων και με ποια; Επιθυμεί ο πελάτης κάποιο εισαγωγικό με animation (κίνηση εικόνας); Επιθυμεί ο πελάτης να γίνεται η διαχείριση του περιεχομένου και η συντήρηση του site; Πόσο γρήγορα θα ανοίγουν οι σελίδες που καλούνται μέσω αργών συνδέσεων;

Τα παραπάνω δεν αποτελούν παρά ελάχιστα από τα ερωτήματα που θα μπορούσαν να τεθούν. Ωστόσο, οι απαντήσεις που θα δοθούν αρκούν για τη χάραξη της αρχικής στρατηγικής. Έτσι

στα ερωτήματα που τέθηκαν θα μπορούσαμε να απαντήσουμε τα εξής ενδεικτικά: "Επιθυμούμε να δημιουργήσουμε ένα ηλεκτρονικό κατάστημα για να εμπορευόμαστε 40 περίπου βιολογικά προϊόντα¨.

Ο ανταγωνισμός στο συγκεκριμένο είδος είναι μικρός και το κοινό στο οποίο θέλουμε να απευθυνθούμε απαρτίζεται από χρήστες με οικονομική άνεση και οικολογική ευαισθησία. Μας ενδιαφέρει να διανέμουμε τα προϊόντα μας σε όλη την Ελλάδα και θέλουμε ο πελάτης να μπορεί να πληρώνει είτε με αντικαταβολή είτε με πιστωτική κάρτα. Για την ανάπτυξη του καταστήματος μπορούμε να διαθέσουμε ως και 15 χιλιάδες ευρώ και θέλουμε το κατάστημα να μπορεί να υποστηρίξει την ταυτόχρονη παραγγελία έως και 500 χρηστών.

Ακόμα, να έχει εγκατεστημένο το πλέον προηγμένο σύστημα για την ταυτοποίηση των χρηστών και την ασφάλεια των δεδομένων. Δεν μας απασχολεί αν θα επιλεγεί μία έτοιμη πλατφόρμα ή θα κατασκευαστεί κάποια εξ αρχής.

Αυτό που μας ενδιαφέρει είναι η διαχείριση του περιεχομένου να γίνεται εύκολα, να μην απαιτείται συντήρηση και να συνεργάζεται με μία βάση δεδομένων, για παράδειγμα υλοποιημένη σε Oracle, που διαθέτουμε. Επίσης, θέλουμε οι σελίδες να ανοίγουν γρήγορα ακόμα και στους επισκέπτες που δεν διαθέτουν γρήγορες συνδέσεις και να υποστηρίζει βέλτιστα τους browsers που διαθέτουν οι περισσότεροι χρήστες.

Κατόπιν αυτών, η εταιρία που πρόκειται να αναλάβει την ανάπτυξη του έργου έχει ήδη λάβει τα πρώτα βασικά στοιχεία. Βεβαίως, η ανάλυση απαιτήσεων δεν σταματά εδώ. Γίνεται όμως έτσι μια πολύ καλή αρχή που θέτει γερά θεμέλια για την αποτελεσματικότερη δόμηση του έργου.

2.4 Συμπεράσματα

Η ανάλυση απαιτήσεων λογισμικού ταξινομείται βασικά σε δύο ευδιάκριτες κατηγορίες: α) ανάλυση απαιτήσεων για την ανάπτυξη νέας εφαρμογής, και β) ανάλυση απαιτήσεων για κάποιο ήδη υπάρχον πακέτο που πρόκειται να παραμετροποιηθεί για μια επιχείρηση (λ.χ. ERP, CRM, κ.λπ.).

Στην πρώτη περίπτωση, η ανάλυση απαιτήσεων συνίσταται στην κατάρτιση των προδιαγραφών (λειτουργικών, τεχνολογικών, κ.ά.) που πρέπει να έχει η καινούργια εφαρμογή. Σε αντίθεση με υπολογίσιμη μερίδα προγραμματιστών που γράφουν κώδικα βάσει ενός πολύ γενικού οδηγού (ή ακόμα και απουσία αυτού), η ανάλυση απαιτήσεων υπαγορεύει λεπτομερή κατάρτιση ενεργειών και στόχων.

Για παράδειγμα, τι ακριβώς θα επιτελεί η εφαρμογή και ποιες δυνατότητες θα προσφέρει στο χρήστη, πόσο γρήγορα θα "τρέχει" και πόσους υπολογιστικούς πόρους θα καταναλώνει, αν θα μπορεί να αναβαθμιστεί, αν θα είναι συμβατή με συγκεκριμένα λειτουργικά προγράμματα, ποιες θα είναι οι λεγόμενες απαιτήσεις συστήματος ("system requirements"), και τελικά ποια θα είναι τα κριτήρια που θα καθορίσουν αν η εφαρμογή πληροί τους στόχους που τέθηκαν εξ αρχής. Στις προδιαγραφές αυτές εντάσσονται και θέματα κανονιστικά.

Ένα τέτοιο πρόσφατο και σχετικό παράδειγμα είναι το ύψος του ΦΠΑ στις εμπορολογιστικές εφαρμογές. Πριν από λίγο καιρό, ο φόρος αυτός αυξήθηκε κατά μερικές μονάδες. Τα προγράμματα εμπορικής και λογιστικής διαχείρισης που είχαν προβλέψει ότι ο ΦΠΑ μπορεί να αυξομειώνεται, αντιμετώπισαν εύκολα τη νέα εξέλιξη. Οι λοιπές εφαρμογές εμφάνισαν προβλήματα. Το συγκεκριμένο παράδειγμα ισχύει και για κάθε είδους κανονιστικές ρυθμίσεις που ενσωματώνονται σε κάποιο πρόγραμμα.

Στη δεύτερη περίπτωση (παραμετροποίηση υπάρχουσας εφαρμογής), η ανάλυση απαιτήσεων σχετίζεται με τον καθορισμό των προδιαγραφών που πρέπει να τηρεί το πρόγραμμα που θα εγκατασταθεί. Στον καθορισμό αυτό εξέχουσα θέση κατέχουν τα εξής:

- Οι δυνατότητες του προγράμματος, λεπτομερώς
- Ο χρόνος αποπεράτωσης (υλοποίησης) του έργου
- Η σειρά με την οποία θα γίνει η ολοκλήρωση
- Ο χρόνος που θα απαιτηθεί για την εκπαίδευση των χρηστών

- Η συμβατότητα και η επικοινωνία του με άλλες εφαρμογές λογισμικού αλλά και συσκευές που ήδη υπάρχουν στην επιχείρηση
- Οι δυνατότητες αναβάθμισης
- Το συνολικό κόστος του έργου, έστω και κατά προσέγγιση
- Το κόστος για ετήσια συντήρηση
- Ο χρόνος επενδυτικής απόσβεσης.

Η ανάλυση των απαιτήσεων γίνεται με δύο τρόπους. Είτε με το ιδεατό μοντέλο (conceptual model) που παριστάνει τον κόσμο και περιγράφεται με τα διαγράμματα κλάσεων, δραστηριότητας, και κατάστασης είτε με το μοντέλο απαιτήσεων (requirements model) που παριστάνει τις απαιτήσεις του λογισμικού και περιγράφεται με τα διαγράμματα περιπτώσεων χρήσης, κλάσεων, συνεργασίας, ακολουθίας και κατάστασης.

Κεφάλαιο 3 Μοντέλα κύκλου ζωής λογισμικού

Σκοπός του κεφαλαίου είναι η γνωριμία και η κατανόηση των μοντέλων κύκλου ζωής, με στόχο την καλύτερη κατανόηση του συστήματος, που θέλουμε να αναπτύξουμε.

3.1 Ορισμός

Μοντέλο κύκλου ζωής λογισμικού ονομάζουμε την περιγραφή των δραστηριοτήτων και των επιμέρους φάσεων, από τις οποίες διέρχεται μια εφαρμογή λογισμικού από τη σύλληψη μέχρι και την απόσυρσή της, καθώς και των εργασιών που λαμβάνουν χώρα σε καθεμιά απ' αυτές τις φάσεις. Πιο απλά, μπορούμε να δούμε το μοντέλο ως μια απλοποίηση της πραγματικότητας, προκειμένου να κατανοήσουμε καλύτερα το σύστημα που έχουμε σκοπό να αναπτύξουμε. Μια δραστηριότητα (ή διαδικασία ανάπτυξης λογισμικού) προσδιορίζει ποιές ενέργειες πρέπει να γίνουν, ώστε να πραγματοποιηθεί το επιθυμητό αποτέλεσμα σε κάποια από τις φάσεις του κύκλου ζωής. Αξίζει να αναφερθεί, ότι η δραστηριότητα μπορεί να αναλυθεί σε μία ή περισσότερες επιμέρους φάσεις.

Οι διαδικασίες ανάπτυξης λογισμικού μπορούν να ταξινομηθούν ως εξής:

- Προδιαγραφή, κατά την οποία ορίζονται οι εργασίες που θα επιτελεί το λογισμικό και οι περιορισμοί του.
- Ανάπτυξη, κατά την οποία κατασκευάζεται το λογισμικό.
- Επαλήθευση, κατά την οποία ελέγχουμε αν οι προδιαγραφές ικανοποιούνται πλήρως και επιβεβαιωνόμαστε ότι δεν υπάρχουν ή θα υπάρξουν σφάλματα και,
- Εξέλιξη, κατά την οποία πραγματοποιείται επαύξηση των λειτουργικών χαρακτηριστικών του λογισμικού ή τροποποίηση των υπαρχουσών λειτουργιών, προκειμένου να μπορεί να ανταποκριθεί σε μελλοντικές ανάγκες.

Τα μοντέλα κύκλου ζωής περιγράφουν με ακρίβεια τις διαδικασίες ανάπτυξης, οι οποίες λαμβάνουν χώρα κατά τις γενικές φάσεις κατασκευής και συντήρησης, προσδιορίζοντας τις

επιμέρους φάσεις, στις οποίες αναλύονται τα προϊόντα που παράγονται σε κάθε μία από αυτές, καθώς και την σειρά εκτέλεσής τους.

Ένα μοντέλο κύκλου ζωής στοχεύει στην καθοδήγηση του κατασκευαστή, προκειμένου να επιτύχει την άριστη παραγωγή των διαδικασιών ανάπτυξης λογισμικού, με όσο το δυνατόν λιγότερα σφάλματα και ρίσκο.

Ένα μοντέλο κύκλου ζωής μπορεί να υλοποιείται με περισσότερες από μία διαδικασίες ανάπτυξης, ενώ μία διαδικασία ανάπτυξης αφορά ακριβώς ένα μοντέλο κύκλου ζωής.

Μία μεθοδολογία μπορεί να υποστηρίζεται από περισσότερα του ενός εργαλεία, όπως και ένα εργαλείο μπορεί να υποστηρίζει περισσότερες της μίας μεθοδολογίες.

Η διαδικασία της μοντελοποίησης μας βοηθά να αστικοποιήσουμε τις λειτουργίες είτε του παρόντος συστήματος, είτε του συστήματος με τις λειτουργίες που θα θέλαμε να έχει, αφού τα μοντέλα μας επιτρέπουν να προσδιορίσουμε τη δομή και τη συμπεριφορά του συστήματος. Τα μοντέλα, κατά την ανάπτυξη ενός συστήματος, αποτελούν ένα πρότυπο, το οποίο το χρειαζόμαστε ως οδηγό στην πορεία της υλοποίησης, ενώ ταυτόχρονα τεκμηριώνουν τις αποφάσεις που λαμβάνουμε.

Στην ιστορική πορεία της ανάπτυξης του λογισμικού εμφανίστηκαν αρκετές διαφορετικές προσεγγίσεις, όπου κάθε προσέγγιση μπορεί να εκφραστεί με διαφορετικά επίπεδα ανάλυσης και ακρίβειας. Τα σημαντικότερα μοντέλα ανάπτυξης λογισμικού θα παρουσιάσουμε παρακάτω.

3.2 Χαρακτηριστικά μοντέλα κύκλου ζωής λογισμικού

Τα κυριότερα μοντέλα κύκλου ζωής λογισμικού είναι:

- Μοντέλο καταρράκτη.
- Μοντέλο πρωτοτυποποίησης
- Αντικειμενοστρεφές μοντέλο
- Σπειροειδές μοντέλο
- Μοντέλο πίδακα
- Μοντέλο λειτουργικής επαύξησης
- Σύγχρονα μοντέλα κύκλου ζωής λογισμικού.

Παρακάτω αναλύονται τα μοντέλα κύκλου ζωής.

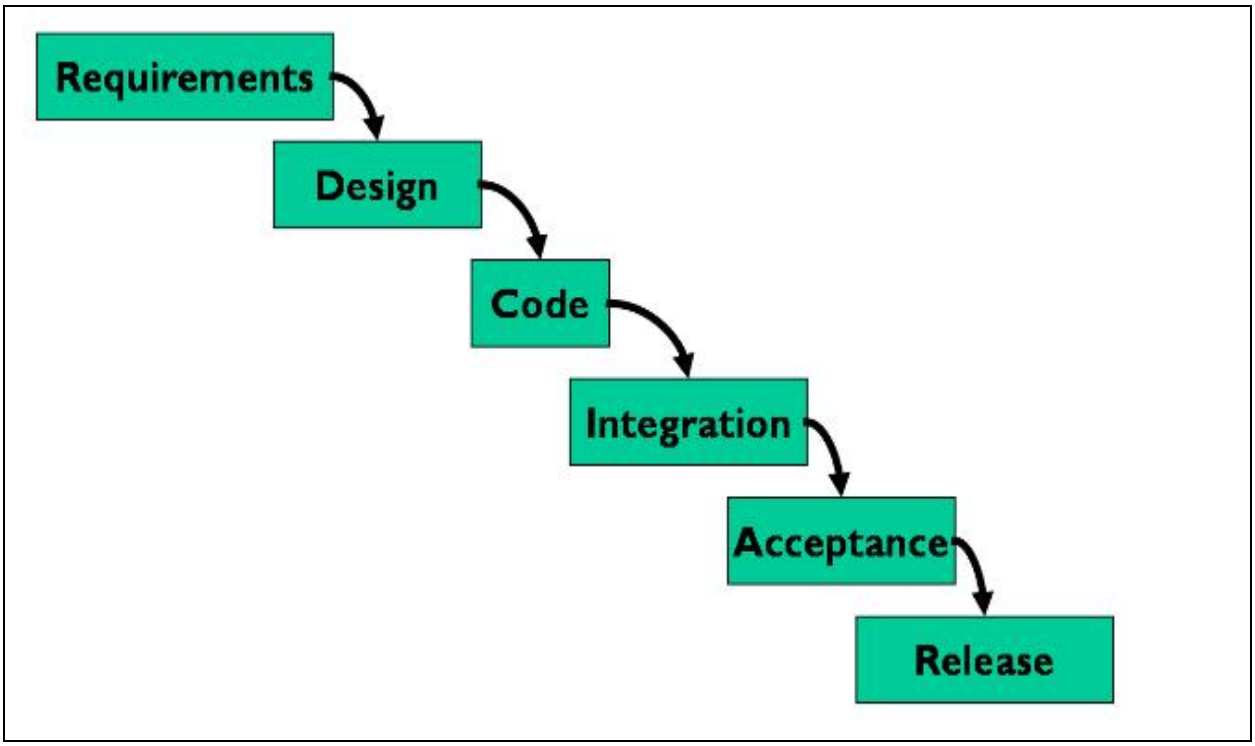
3.2.1 Μοντέλο καταρράκτη

Το μοντέλο του Καταρράκτη (Kenneth C. Laudon) (waterfall ή linear sequential model) ήταν το πρώτο που δημιουργήθηκε και αναπτύχθηκε το 1970 από τον Royce (Royce, 1970) και αποτελεί ένα από τα ποιο διαδεδομένα μοντέλα μέχρι και σήμερα, ιδιαίτερα για μικρά ή μεσαία μεγέθη εφαρμογών, αφού συμβάλλει στην επιτυχή κατασκευή αξιόπιστων προϊόντων σε μικρό χρονικό διάστημα .

Το μοντέλο αυτό είναι χρήσιμο όταν οι απαιτήσεις από το λογισμικό είναι γνωστές από την αρχή και δεν μεταβάλλονται κατά την ανάπτυξη του λογισμικού. Η κύρια ιδέα του είναι ότι το σύστημα λογισμικού αναπτύσσεται περνώντας ολόκληρο από διαδοχικές επιμέρους φάσεις, καθεμιά απ’ τις οποίες θεωρείται περατωμένη με την παραγωγή ορισμένων συστατικών λογισμικού.

Κάθε επιμέρους φάση ολοκληρώνεται με την εργασία επαλήθευση/επικύρωση των προϊόντων της, κατά την οποία αποφασίζεται αν θα μεταβεί ή όχι στην επόμενη. Το λογισμικό περιλαμβάνει όλα τα λειτουργικά του χαρακτηριστικά από τη φάση της συνένωσης και μετά.

Το βασικό χαρακτηριστικό του μοντέλου καταρράκτη είναι ότι δεν μπορεί να ξεκινήσει η επόμενη φάση, αν δεν έχει ολοκληρωθεί η προηγούμενη πλήρως. Για αυτόν ακριβώς τον λόγο, η ανάπτυξη του λογισμικού μέσω αυτού του μοντέλου, καλείται ακολουθιακή.



Εικόνα 3. Το μοντέλο καταρράκτη.

Τα πλεονεκτήματα που παρουσιάζει το μοντέλο του καταρράκτη είναι τα εξής:

- Περιορίζεται το κόστος όλης της διαδικασίας παραγωγής του λογισμικού.
- Επιτρέπει την επιστροφή της πληροφορίας στα προηγούμενα στάδια.
- Υπάρχει δυνατότητα ελέγχου του ολοκληρωμένου συστήματος.
- Είναι εύκολα κατανοητή και αποδεκτή από όσους συμμετέχουν στη διαδικασία ανάπτυξης (πελάτες, χρήστες, ανάδοχοι).
- Βοηθά στον καταμερισμό της εργασίας μεταξύ προγραμματιστών, αναλυτών, πωλητών, και των επικεφαλής και
- Είναι χρήσιμο για λόγους διαχείρισης του έργου.

Τα μειονεκτήματα που παρουσιάζει το μοντέλο του καταρράκτη είναι τα εξής:

- Δεν γνωρίζουμε αν θα είναι ικανοποιημένος ο χρήστης, παρά μόνο στο τελικό στάδιο
- Η γραμμικότητα που υποθέτει το μοντέλο σπανίως συναντάται σε πραγματικά έργα.
- Η ανάλυση του συστήματος και ο εντοπισμός των απαιτήσεων σπανίως μπορούν να ολοκληρωθούν στην αρχή ενός έργου.
- Μεσολαβεί μεγάλο χρονικό διάστημα από την έναρξη του έργου μέχρι την πρώτη παραδοτέα έκδοση του συστήματος. Στο μεταξύ παραδίδεται μόνο τεκμηρίωση.
- Οι πελάτες και οι χρήστες αργούν να πάρουν μια λειτουργική εικόνα του συστήματος. Είναι εύκολο να υποτιμηθεί η καταβαλλόμενη προσπάθεια.
- Η ομάδα ανάπτυξης αργεί να αποκτήσει μια απτή έκδοση του συστήματος. Είναι εύκολο να απογοητεύσει μια προσπάθεια που για μεγάλο διάστημα δε φαίνεται να αποδίδει.
- Το μοντέλο εύκολα οδηγεί σε πλήρη διαχωρισμό των ρόλων των προγραμματιστών, των αναλυτών, των πωλητών και των επικεφαλής, με πιθανές αρνητικές συνέπειες.
- Δεν προβλέπει επαναχρησιμοποίηση του λογισμικού που πιθανά υπάρχει.
- Είναι απαιτητικό τόσο σε χρόνο όσο και σε κόστος και αρκετά δύσκαμπτο εργαλείο.
- Όσο μεγαλώνει η έκταση της εφαρμογής λογισμικού, τόσο δυσκολότερη η μετάβαση από τη μία φάση στην επόμενη και η αποφυγή σφαλμάτων τα οποία εντοπίζονται πολύ αργότερα
- Όσο αργότερα στην ανάπτυξη εντοπίζεται ένα σφάλμα, τόσο μεγαλύτερες οι επιπτώσεις που η διόρθωσή του μπορεί να έχει σε κόστος οπισθοδρόμησης, παρενέργειες, καθυστερήσεις, δημιουργία νέων σφαλμάτων

Στον παρακάτω πίνακα μπορούμε να δούμε τους κινδύνους που συνεπάγονται την εφαρμογή αυτού του μοντέλου στην ανάπτυξη λογισμικού.

Μοντέλο Καταρράκτη	
Ολοκλήρωση (Integration)	ΚΙΝΔΥΝΟΣ: Δεν πραγματοποιείται παρά μόνο στο τελευταίο στάδιο του έργου.
Ικανοποίηση Χρήσης (Usability)	ΚΙΝΔΥΝΟΣ: Εξετάζεται όταν το έργο έχει σχεδόν ολοκληρωθεί.
Έλεγχος Φόρτου (Load Test)	ΚΙΝΔΥΝΟΣ: Πραγματοποιείται όταν το έργο έχει σχεδόν ολοκληρωθεί.

Πίνακας 1. Κίνδυνοι από την εφαρμογή του μοντέλου του καταρράκτη.

3.2.2 Μοντέλο Πρωτοτυποποίησης

Το μοντέλο της πρωτοτυποποίησης έχει ως κεντρική ιδέα την ανάπτυξη λογισμικού σε τμήματα και όχι εξ' ολοκλήρου. Τα τμήματα αυτά ονομάζονται πρωτότυπα και περιλαμβάνουν τις βασικές λειτουργίες που θα πρέπει να εκτελεί το λογισμικό. Οι διαδικασίες ανάπτυξης επαναλαμβάνονται κάθε φορά, για ένα τμήμα του συστήματος, γι' αυτό και χαρακτηρίζεται ως επαναληπτικό.

Ένας τρόπος να δούμε την πρωτοτυποποίηση είναι ως μια τεχνική για μείωση του ρίσκου. Το πιο σημαντικό ρίσκο στην ανάπτυξη λογισμικού είναι τα λάθη και οι παραλείψεις που προκύπτουν από μη σαφείς απαιτήσεις των χρηστών για το τελικό σύστημα.

Το κόστος της διόρθωσης αυτών των λαθών και παραλείψεων σε επόμενα στάδια μπορεί να είναι πολύ υψηλό. Είναι προφανές ότι η δημιουργία ενός πρωτοτύπου μπορεί να μειώσει τον αριθμό των προβλημάτων των απαιτήσεων και ως εκ τούτου να μειώσει το συνολικό κόστος ανάπτυξης. Το μοντέλο της πρωτοτυποποίησης είναι χρήσιμο όταν οι απαιτήσεις είναι ασαφείς, προτείνεται για μικρού μεγέθους έργα και δίνει την έμφαση στη διεπαφή του χρήστη.

Μια σχηματική αναπαράσταση της διαδικασίας ανάπτυξης ενός πρωτοτύπου φαίνεται στο σχήμα που ακολουθεί.

Εικόνα 4. Διαδικασία ανάπτυξης πρωτοτύπου.

Αρχικά θα πρέπει να καθοριστούν επακριβώς οι στόχοι του πρωτότυπου. Το πρωτότυπο σύστημα μπορεί να αφορά τη διεπιφάνεια χρήστη (user interface) ή να περιέχει τις λειτουργίες εκείνες που θεωρούνται περισσότερο κρίσιμες.

Είναι προφανές ότι ένα πρωτότυπο δεν μπορεί να καλύπτει όλες τις απαιτήσεις του συστήματος. Για το λόγο αυτό κάθε φορά θα πρέπει να ορίζονται πλήρως οι απαιτήσεις που αυτό θα καλύπτει, αλλιώς τελικά μπορεί να μη λάβουμε τα πλεονεκτήματα που μας προσφέρει η μέθοδος αυτή.

Το επόμενο στάδιο αφορά στο να αποφασιστεί ποιες ενδεχομένως λειτουργίες του τελικού συστήματος δεν θα πρέπει να συμπεριληφθούν γιατί έχει πολύ υψηλό κόστος, εάν το πρωτότυπο δημιουργηθεί με όλες τις λειτουργίες του τελικού συστήματος. Βέβαια θα μπορούσε να αποφασιστεί να περιλαμβάνονται όλες οι λειτουργίες που έχει αποφασιστεί αλλά σε μειωμένο επίπεδο (πχ. χωρίς διαχείριση λαθών).

Η τελευταία φάση, μετά την ανάπτυξη του πρωτοτύπου, είναι η επαλήθευση του πρωτότυπου και είναι ίσως η πιο σημαντική φάση. Θα πρέπει να καταγραφούν συμπεράσματα για το πως νιώθουν οι χρήστες με το σύστημα, αν γίνεται κατανοητό το περιβάλλον και η λειτουργία του και να βρεθούν τυχόν λάθη και προβλήματα.

Το πλεονέκτημα της χρήσης πρωτότυπου είναι μας δίνει τη δυνατότητα απόκτησης άποψης για το λογισμικό νωρίτερα από το μοντέλο του καταρράκτη με λιγότερες καθυστερήσεις και κόστη, καθώς κάθε κατασκευή πρωτοτύπου αποτελεί ένα μικρό έργο λογισμικού. Σημαντικό πλεονέκτημα είναι η ευκολία που μας δίνει στη διοίκηση του έργου καθώς και στην υλοποιησιμότητα, όπως επίσης μας επιτρέπει με ευκολία την οποιαδήποτε τροποποίηση πρωτοτύπου.

Λόγω των πλεονεκτημάτων του, το μοντέλο προτυποποίησης χρησιμοποιείται σε εφαρμογές που οι απαιτήσεις δεν είναι γνωστές από την αρχή διότι,

- Αποφεύγονται παρεξηγήσεις μεταξύ των χρηστών και των δημιουργών.
- Παρέχει ευκολία για συμπλήρωση παραλειπόμενων υπηρεσιών στο σύστημα
- Λύνει δυσκολίες στη χρήση
- Καλύπτει ασυνέχειες και κενά στις προδιαγραφές

Η πρωτοτυποποίηση μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς, όπως στην εκπαίδευση των χρηστών, δηλαδή το πρωτότυπο να χρησιμοποιηθεί ως εκπαιδευτικό εργαλείο για την εκμάθηση του τελικού συστήματος, αλλά ταυτόχρονα είναι και ένας τρόπος μείωσης του ρίσκου, αφού περιορίζονται τα λάθη και οι παραλείψεις.

Αν τα λάθη αφεθούν για διόρθωση στις τελευταίες φάσεις του κύκλου ζωής το κόστος αυξάνεται κατακόρυφα.

Το βασικό μειονέκτημα του μοντέλου της πρωτοτυποποίησης είναι ότι το κόστος ανάπτυξής του αποτελεί ένα μεγάλο μέρος του συνολικού κόστους του συστήματος που αναπτύσσεται.

Πολλές φορές είναι οικονομικά πιο συμφέρον να μεταβληθεί το τελικό προϊόν από το να δημιουργηθεί ένα πρωτότυπο.

Είναι προφανές, ότι είναι πολύ δύσκολο να προβλεφτεί ποιες ακριβώς δυσκολίες θα αντιμετωπίσει ο τελικός χρήστης από την καθημερινή χρήση ενός νέου συστήματος λογισμικού. Ιδιαίτερα εάν αναφερόμαστε σε μεγάλα συστήματα λογισμικού η δυσκολία αυτή μπορεί να καταφανεί μόνο όταν το ολοκληρωμένο σύστημα αναπτυχθεί και τεθεί σε λειτουργία.

Για να αντιμετωπιστεί αυτή η δυσκολία μπορεί να χρησιμοποιηθεί η εξελικτική (evolutionary) πρωτοτυποποίηση. Κατά τη προσέγγιση αυτή δημιουργείται μια περιορισμένη (ατελής) έκδοση του συστήματος πάνω στην οποία γίνονται διορθώσεις και προσθήκες καθώς απαιτήσεις είτε διευκρινίζονται είτε ανακαλύπτονται είτε βελτιώνονται μέχρι να καταλήξουμε σε μια έκδοση που να είναι επαρκής και ικανοποιητική.

Το βασικό πρόβλημα του μοντέλου της εξελικτικής πρωτοτυποποίησης είναι ότι με τις συνεχείς διορθωτικές και προσθετικές παρεμβολές παράγεται ‘μπλεγμένος’ κώδικας (spaghetti code) που είναι πολύ δύσκολα συντηρήσιμος. Εναλλακτικά και προκειμένου να αποφύγουμε το φαινόμενο του ‘μπλεγμένου κώδικα’, θα μπορούσε να δημιουργηθεί ένα πρωτότυπο το οποίο αποκλειστικό σκοπό θα είχε την αποσαφήνιση των απαιτήσεων και να παράσχει πληροφορίες για την εκτίμηση του ρίσκου του τελικού συστήματος.

3.2.3 Αντικειμενοστρεφές μοντέλο

Το αντικειμενοστρεφές μοντέλο (object-oriented model) βασίζεται στον αντικειμενοστρεφή προγραμματισμό και αναπτύσσεται με τρόπο παρόμοιο με το μοντέλο του καταρράκτη. Ενδείκνυται για την μοντελοποίηση οντοτήτων του πραγματικού κόσμου. Από το μοντέλο του καταρράκτη διαφέρει σε δύο βασικά σημεία:

- Οι διάφορες φάσεις υπερκαλύπτονται μεταξύ τους.
- Η ανάπτυξή του, αν χρειαστεί οπισθοδρομεί στην προηγούμενη φάση, εκτός από την τελευταία που οπισθοδρομεί στην αρχή.

Το κύριο πλεονέκτημα του μοντέλου είναι ότι κάνει χρήση επαναχρησιμοποιήσιμων μονάδων και με αυτό τον τρόπο συντομεύεται τόσο η φάση της ανάπτυξης όσο και η φάση της συντήρησης.

Με το πέρασμα του χρόνου κωδικοποιήθηκαν κάποιες ανεπίσημες αρχές για την ορθή σχεδίαση αντικειμενοστραφών συστημάτων λογισμικού. Οι αρχές αυτές παρουσιάστηκαν κατά καιρούς σε βιβλία και άρθρα ακαδημαϊκών και αναγνωρισμένων μηχανικών λογισμικού και είναι οι παρακάτω:

Αρχή ανοιχτότητας-κλειστότητας (open-closed principle), του δημιουργού της γλώσσας προγραμματισμού Eiffel Μπέρτραντ Μέιερ. Η αρχή αυτή δηλώνει πως τα συστατικά ενός προγράμματος πρέπει να είναι "ανοιχτά" ως προς την επέκταση των δυνατοτήτων του συστήματος, αλλά "κλειστά" ως προς αλλαγές στην υλοποίηση του.

Πρακτικώς αυτό σημαίνει οι διάφορες κλάσεις και τα υπόλοιπα τμήματα λογισμικού να μη χρειάζεται να τροποποιηθούν σε περίπτωση που προστεθεί νέα λειτουργικότητα στο σύστημα (π.χ. μία νέα κλάση) προκειμένου να την αξιοποιήσουν.

Βεβαίως είναι αδύνατο να μη χρειάζεται να τροποποιηθεί τίποτα, οπότε αυτό που επιτάσσει στην πραγματικότητα η εν λόγω αρχή είναι η ελαχιστοποίηση και η συγκέντρωση, κατά προτίμηση σε ένα μικρό τμήμα του κώδικα, των γραμμών που θα πρέπει να αλλάξουν.

Αυτό συνήθως επιτυγχάνεται μέσω αφαίρεσης (με αφηρημένες κλάσεις ή διασυνδέσεις και πραγματικές κλάσεις που κληρονομούν από αυτές) και με χρήση του πολυμορφισμού.

Αρχή υποκατάστασης Λίσκοφ (Liskov substitution principle), της επιστήμονα υπολογιστών Μπάρμπαρα Λίσκοφ. Η αρχή αυτή συμπυκνώνεται στον παρακάτω κανόνα για σχηματισμό μίας ορθής ιεραρχίας κλάσεων: μία κλάση Κ1 μπορεί να υλοποιηθεί ως υποκλάση μίας κλάσης Κ2 αν κάθε πρόγραμμα Π το οποίο λειτουργεί με αντικείμενα Κ2 συμπεριφέρεται με τον ίδιο τρόπο και με αντίστοιχα αντικείμενα Κ1.

Έτσι με την αρχή υποκατάστασης Λίσκοφ φαίνεται πως για να οριστεί μία κλάση ως θυγατρική μίας άλλης δεν αρκεί να έχουν διαισθητικά μία ανάλογη εννοιολογική σχέση (π.χ. μία κλάση που αναπαριστά όχημα και μία που αναπαριστά αυτοκίνητο) αλλά, στο πλαίσιο του υπό εξέταση προγράμματος, τα αντικείμενα της υποκλάσης να έχουν πάντα την ίδια προγραμματιστική συμπεριφορά με τα αντικείμενα της υπερκλάσης υπό τις ίδιες συνθήκες.

Αρχή αντιστροφής εξαρτήσεων (dependency inversion principle), του γνωστού μηχανικού λογισμικού Ρόμπερτ Σέσιλ Μάρτιν. Η αρχή αυτή πρακτικά αποτελεί εκλέπτυνση της αρχής ανοιχτότητας-κλειστότητας, προϋποθέτοντας όμως χρήση και της αρχής υποκατάστασης Λίσκοφ. Αφορά ιεραρχίες κληρονομικότητας κλάσεων και τη χρήση αντικειμένων αυτών των ιεραρχιών από εξωτερικά προγράμματα.

Στα πλαίσια της αρχής αντιστροφής εξαρτήσεων ένα τμήμα λογισμικού Α (π.χ. μία κλάση) το οποίο χρησιμοποιεί τις υπηρεσίες που παρέχει ένα άλλο τμήμα λογισμικού Β, καλώντας για παράδειγμα μία μέθοδό του, θεωρείται στοιχείο "υψηλότερου επιπέδου" σε σχέση με το Β. Η αρχή λέει πως τα υψηλού επιπέδου στοιχεία δεν πρέπει να εξαρτώνται από την υλοποίηση χαμηλότερου επιπέδου στοιχείων, αλλά πως και τα δύο πρέπει να βασίζονται σε ενδιάμεσα επίπεδα αφαίρεσης.

Στην πράξη αυτή η αφαίρεση είναι μία διασύνδεση (ή αφηρημένη κλάση) την οποία γνωρίζει το υψηλού επιπέδου στοιχείο Α και υλοποιεί το χαμηλού επιπέδου στοιχείο Β. Ακόμα και αν το Β αλλαχθεί με μία κλάση Γ η οποία επίσης υλοποιεί την ίδια διασύνδεση, το Α θα πρέπει να συνεχίσει να λειτουργεί χωρίς καμία τροποποίηση. Η αρχή αντιστροφής εξαρτήσεων δεν είναι

παρά ένα απτό παράδειγμα χρήσης ιεραρχικών επιπέδων με τη βοήθεια ενδιάμεσων αφαιρέσεων, μίας πρακτικής που εφαρμόζεται κατά κόρον στην επιστήμη υπολογιστών.

Αρχή διαχωρισμού διασυνδέσεων (interface segregation principle), του μηχανικού λογισμικού, Ρόμπερτ Σέσιλ Μάρτιν. Η εν λόγω αρχή σημαίνει ότι σε περιπτώσεις όπου διαφορετικά υποσύνολα μεθόδων μίας κλάσης αφορούν διαφορετικές περιπτώσεις χρήσης της κλάσης, σκόπιμο είναι να ορίζουμε επιμέρους διασυνδέσεις, τις οποίες η κλάση θα υλοποιεί. Κάθε τέτοια διασύνδεση θα ορίζει μόνο το αντίστοιχο υποσύνολο των μεθόδων.

Αρχή μοναδικής αρμοδιότητας (single responsibility principle), των Τομ Ντε Μάρκο και Μέιρ Πέιτζ Τζόουνς. Σύμφωνα με την αρχή αυτή κάθε κλάση θα πρέπει να έχει μόνο μία, καλά ορισμένη και διαχωρισμένη από το υπόλοιπο πρόγραμμα αρμοδιότητα, η ύπαρξη της οποίας να εξυπηρετεί έναν συγκεκριμένο σκοπό. Αν μπορούμε να εντοπίσουμε σε μία κλάση Α δύο διαφορετικές αρμοδιότητες, τότε η καλύτερη λύση είναι η διάσπαση της σε δύο κλάσεις Β' και Γ', καθεμία από τις οποίες θα λάβει ένα υποσύνολο των πεδίων και των μεθόδων της Α. Τα υποσύνολα αυτά θα είναι ξένα μεταξύ τους, οπότε με το ανάποδο σκεπτικό αν μπορούμε να διασπάσουμε μία κλάση Α σε δύο άλλες κλάσεις (π.χ. σε περίπτωση που κάποιες μέθοδοι δε χρησιμοποιούν κάποια γνωρίσματα, οπότε οι μεν μπορούν να καταλήξουν στη μία κλάση Β' και τα πεδία στην άλλη κλάση Γ') τότε πιθανώς η κλάση να παραβιάζει την αρχή μοναδικής αρμοδιότητας.

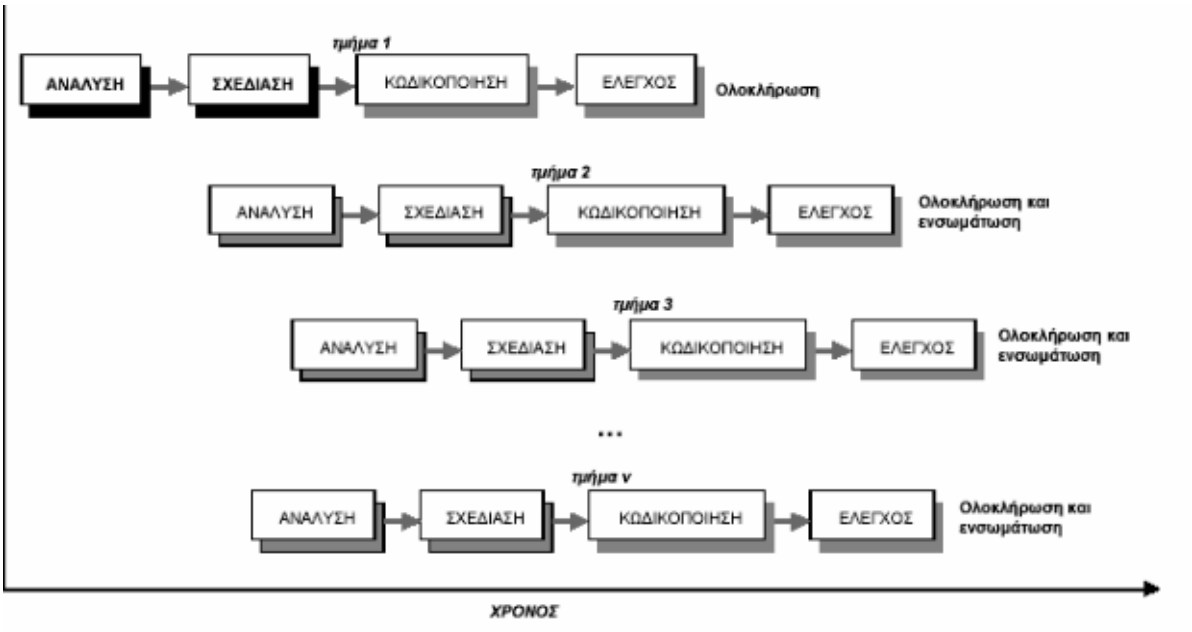
Έτσι έχουν προταθεί κάποιες μετρικές οι οποίες επιχειρούν να προσδιορίσουν την έλλειψη συνοχής (cohesion) σε μία κλάση, δηλαδή το κατά πόσον οι μέθοδοι της δε σχετίζονται με τα γνωρίσματα της. Συνήθως, η συνοχή αντιπαραβάλλεται με τη σύζευξη (coupling), δηλαδή το βαθμό στον οποίον μία κλάση εξαρτάται από κάποια/ες άλλη/ες, και τα δύο αυτά μεγέθη είναι αντιστρόφως ανάλογα.

3.2.4 Μοντέλο λειτουργικής επαύξησης

Το μοντέλο λειτουργικής επαύξησης συνδυάζει την ακολουθιακή ανάπτυξη του μοντέλου καταρράκτη και την τμηματική ανάπτυξη του μοντέλου πρωτοτυποποίησης. Η κύρια ιδέα του είναι η κατάτμηση του υπό κατασκευή λογισμικού σε τμήματα που αναπτύσσονται ανεξάρτητα, τα οποία προπορεύονται σε ακολουθιακή ανάπτυξη, σύμφωνα με το μοντέλο καταρράκτη.

Διαχωρίζεται σε 4 διαφορετικές φάσεις, οι οποίες δεν είναι απαραίτητο να ξεκινούν από το ίδιο σημείο. Κάθε φορά που τελειώνει το κάθε τμήμα, έχουμε και την ολοκλήρωσή του, ενώ στο επιμέρους μοντέλο είναι δυνατή η εφαρμογή ενός ή περισσοτέρων μοντέλων, διαφορετικού είδους. Επιτρέπει την ανάπτυξη τμημάτων ανεξάρτητα και παράλληλα. Λειτουργική επαύξηση σημαίνει η ενσωμάτωση κάθε τμήματος στο σύνολο της εφαρμογής, μετά την ολοκλήρωση της ανάπτυξής του.

Για να είναι επιτυχημένη μια εφαρμογή θα πρέπει να χρησιμοποιείται σε ανάπτυξη εφαρμογών με σαφή γνώση των απαιτήσεων και μικρή ή καμιά μεταβλητότητά τους.



Εικόνα 5. Μοντέλο λειτουργικής επαύξησης

Πλεονεκτήματα

Με το μοντέλο λειτουργικής επαύξησης έχουμε τη δυνατότητα παράλληλης ανάπτυξης και διαδοχικού εμπλουτισμού των λειτουργικών χαρακτηριστικών του λογισμικού, διαδοχικό εμπλουτισμό των λειτουργικών χαρακτηριστικών, καθώς και μικρότερο χρόνο ανάπτυξης της εφαρμογής

Μειονεκτήματα

Τα κύρια μειονεκτήματα είναι ότι στην περίπτωση που δημιουργηθούν σφάλματα στην αρχική κατάτμηση και τη γενική σχεδίαση του συστήματος μπορούν να έχουν ως αποτέλεσμα σημαντικές επιπτώσεις στο λογισμικό. Επίσης, είναι απαραίτητη η αρχική κατάτμηση και γενική σχεδίαση του συστήματος, και σε περίπτωση μεταβολής των λειτουργικών απαιτήσεων κατά τη χρήση του ημιτελούς συστήματος, μπορεί να προκύψει πρόβλημα λόγω ακατάλληλης κατάτμησης εφαρμογής.

Σε περίπτωση μεταβολής των λειτουργικών απαιτήσεων κατά τη χρήση του ημιτελούς συστήματος, μπορεί η αρχιτεκτονική αυτών, να μεταβληθεί σε βαθμό που να κλονιστεί η ανάπτυξη των υπολοίπων τμημάτων του, πράγμα που καθιστά μη λειτουργική οποιαδήποτε μεταβολή του μοντέλου ακόμη και αν σε μελλοντικές καταστάσεις ο χρήστης του μοντέλου έχει επιπλέον ανάγκες.

Το Μοντέλο Λειτουργικής Επαύξησης χρησιμοποιείται σε μεγάλες εφαρμογές λογισμικού όπου είναι γνωστές οι απαιτήσεις που έχουμε καθώς έχουμε μικρή μεταβλητότητα κατά την ανάπτυξη.

3.2.5 Σπειροειδές μοντέλο

Τα μοντέλα κύκλου ζωής που παρουσιάστηκαν μέχρι τώρα αποτελούν παραλλαγές της βασικής ιδέας του μοντέλου του καταρράκτη. Η ανάπτυξη παραμένει επί της ουσίας μια ακολουθιακή διαδικασία η οποία εφαρμόζεται είτε σε ολόκληρο, είτε σε ένα μέρος του συστήματος.

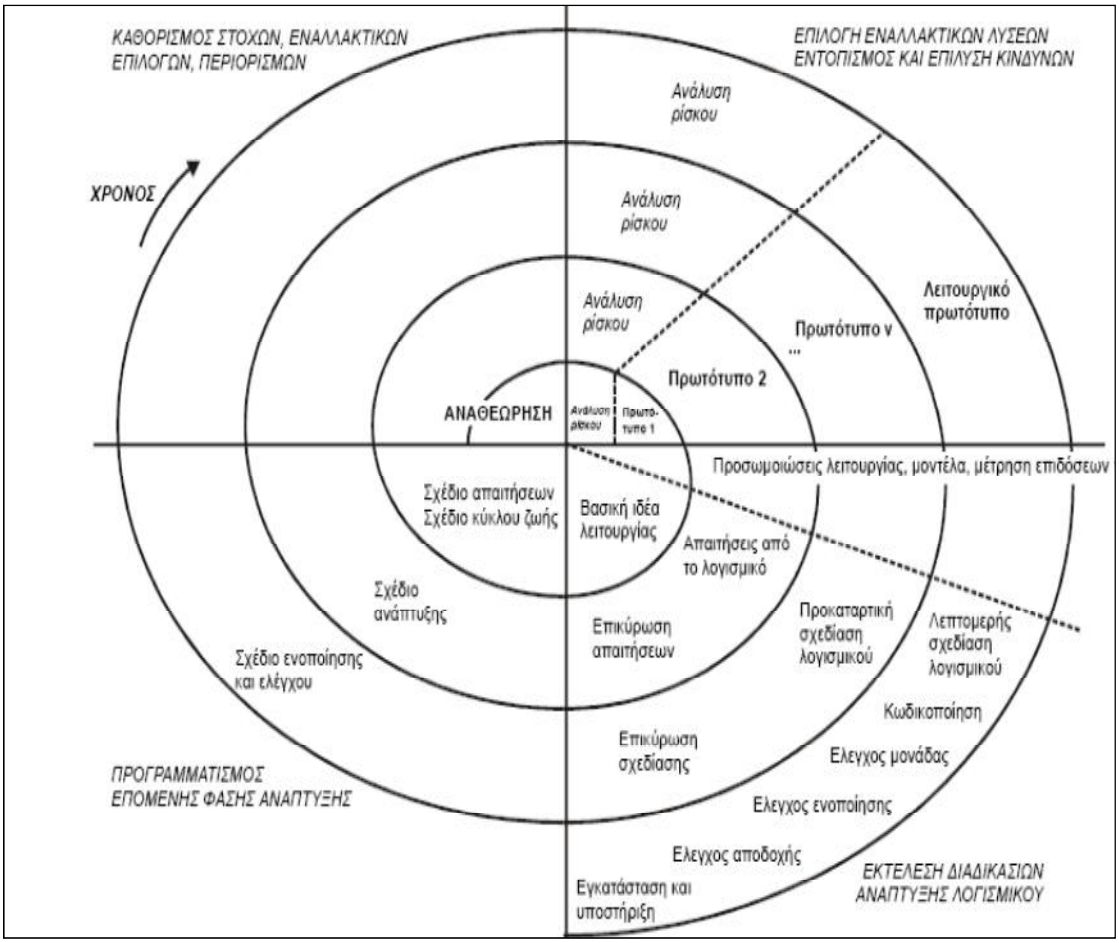
Από ότι φαίνεται, δεν είναι η σύλληψη των διαδικασιών ανάπτυξης λογισμικού που διαφοροποιεί τα μοντέλα κύκλου ζωής, αλλά η διάταξή τους. Στο μοντέλο της πρωτυποποίησης καθώς και σε αυτό της λειτουργικής επαύξησης η κατάτμηση είναι λίγο ως πολύ αυθαίρετη. Το ρίσκο δεν αποτιμάται, με αποτέλεσμα κάθε οπισθοδρόμηση ή ανατροπή να κοστίζει, σε χρόνο και σε οικονομικούς πόρους, συχνά δε και στη συνολική επιτυχία των έργων.

Από την άλλη, η μετά πειθαρχίας αποδοχή των αυστηρών φάσεων που προτείνονται από το μοντέλο του καταρράκτη δεν είναι εφικτό να ακολουθείται σε όλες τις περιπτώσεις και από όλους τους κατασκευαστές, με αποτέλεσμα η ανάπτυξη λογισμικού είτε να γίνεται άναρχα με βάση τη διαίσθηση των κατασκευαστών, είτε να είναι μια δαπανηρή και στριφνή διαδικασία στην οποία "πρέπει" να ακολουθηθούν κάποια συγκεκριμένα βήματα, ανεξάρτητα από τις εκάστοτε συνθήκες.

Απάντηση στα παραπάνω έρχεται να δώσει το σπειροειδές μοντέλο κύκλου ζωής, το οποίο πήρε το όνομά του από την απεικόνιση σε διάγραμμα, όπως φαίνεται στο παρακάτω σχήμα. Πρόκειται για μια γενίκευση των μοντέλων της λειτουργικής επαύξησης και της πρωτυποποίησης, με σημαντικά νέα στοιχεία. Το σπειροειδές μοντέλο, ουσιαστικά, είναι το μοντέλο της πρωτοτυποποίησης, όπου στο τέλος κάθε βήματος πραγματοποιούμε έλεγχο σκοπιμότητας και ανάλυση ρίσκου. Γενικά, είναι κατάλληλο για μεγάλα έργα, λόγω του μεγαλύτερου κόστους διαχείρισης.

Οι φάσεις και οι διαδικασίες ανάπτυξης λογισμικού δεν είναι προκαθορισμένες από το μοντέλο αλλά εξειδικεύονται στο χώρο της εφαρμογής του. Η ανάπτυξη ολόκληρου του συστήματος χωρίζεται σε πολλούς κύκλους, σε καθέναν από τους οποίους προστίθενται νέα λειτουργικά χαρακτηριστικά στο σύστημα.

Πριν από την έναρξη κάθε κύκλου γίνεται μια μελέτη σκοπιμότητας και ανάλυση κινδύνων από την οποία προκύπτουν αφ' ενός οι συγκεκριμένες εργασίες που θα εκτελεστούν μέσα στον κύκλο, αφ' ετέρου η ίδια η εφικτότητα εκτέλεσης του κύκλου αυτού.



Εικόνα 6. Λειτουργία σπειροειδούς μοντέλου.

Όπως φαίνεται στο σχήμα, στο σπειροειδές μοντέλο διακρίνονται τέσσερις κατηγορίες εργασιών: προσδιορισμός στόχων, εντοπισμός και επίλυση κινδύνων, εκτέλεση διαδικασιών ανάπτυξης και επαλήθευση, καθώς και εργασίες προγραμματισμού.

Κατά τον προσδιορισμό στόχων, καθορίζονται τα αντικείμενα εργασιών κάθε επανάληψης, καταγράφονται οι περιορισμοί επί του προϊόντος αλλά και επί της διαδικασίας για την οποία

κατασκευάζεται ένα αναλυτικό πλάνο διοίκησης. Επίσης καταγράφονται οι κίνδυνοι που εμπεριέχει η διαδικασία και οι εναλλακτικές λύσεις, όπου υπάρχουν.

Κατά τις εργασίες επίλυσης κινδύνων, αναλύονται οι κίνδυνοι που έχουν καταγραφεί και αποτιμάται κάθε εναλλακτική λύση. Στο σημείο αυτό λαμβάνονται αποφάσεις για τη συνέχιση ή όχι της ανάπτυξης, για το μοντέλο που θα ακολουθηθεί στη συγκεκριμένη επανάληψη, για την κατασκευή ή όχι πρωτοτύπου και άλλα σχετικά ζητήματα.

Ακολουθεί η εκτέλεση των βημάτων της διαδικασίας ανάπτυξης λογισμικού που έχει επιλεγεί για το τμήμα εκείνο του συστήματος που αφορά η τρέχουσα επανάληψη.

Τέλος, μετά την επαλήθευση των αποτελεσμάτων – ενδιάμεσων προϊόντων λογισμικού, γίνεται προγραμματισμός της συνέχισης της ανάπτυξης.

Το σπειροειδές μοντέλο δεν καθορίζει εκ των προτέρων ποιες ακριβώς είναι οι εργασίες ανάπτυξης λογισμικού που πρέπει να γίνουν, ούτε σε ποια έκταση του συστήματος αυτές θα εφαρμοστούν. Διαφορετικές διαδικασίες ανάπτυξης μπορεί να επιλεγούν για διαφορετικά τμήματα του λογισμικού.

Αυτό που προτείνει, είναι ότι ο καθορισμός των λεπτομερειών υλοποίησης πρέπει να γίνεται συνεχώς κατά την ανάπτυξη (και όχι μία φορά, όπως συμβαίνει με τα μοντέλα κύκλου ζωής που αναφέρθηκαν μέχρι τώρα) με ευθύνη και με τεκμηρίωση του ίδιου του κατασκευαστή.

Η εφαρμογή του σπειροειδούς μοντέλου στην πράξη δεν είναι πάντα εύκολη υπόθεση. Εισάγονται νέες εργασίες που δεν ανήκουν καθαρά στις εργασίες ανάπτυξης λογισμικού, αλλά αφορούν την τεκμηρίωση της σκοπιμότητας και τον τμηματικό προγραμματισμό της ανάπτυξης. Οι εργασίες αυτές επιφέρουν ασφαλώς κάποιο κόστος, το οποίο όμως μπορεί να αποσβεστεί από τον έγκαιρο εντοπισμό προβλημάτων και την αποφυγή πιθανού ναυαγίου, κάτι που έχει συμβεί σε αρκετές περιπτώσεις.

Στη διαδικασία της επιλογής εναλλακτικών λύσεων αναλύονται οι κίνδυνοι που έχουν καταγραφεί ενώ αποτιμάται κάθε εναλλακτική λύση. Μετά από αυτό το στάδιο λαμβάνονται

σημαντικές αποφάσεις για τη συνέχιση ή όχι της ανάπτυξης του μοντέλου που θα ακολουθεί στη συγκεκριμένη επανάληψη για την δημιουργία ή όχι του προτύπου.

Από τη στιγμή που κριθεί σκόπιμη η συνέχιση της ανάπτυξης για το μοντέλο γίνεται η εκτέλεση των διαδικασιών που έχει επιλεγεί για το τμήμα εκείνου, του συστήματος που αφορά την τρέχουσα επανάληψη. Μετά την επαλήθευση των αποτελεσμάτων ενδιάμεσων προϊόντων λογισμικού γίνεται ο προγραμματισμός της επόμενης φάσης ανάπτυξης.

Το μοντέλο του Boehm έχει τη μορφή μιας σπείρας (Βεσκούκης, 2000), όπου κάθε σπείρα αναπαριστά και μια φάση έτσι όπως αυτή καθορίζεται κάθε φορά από την διοίκηση.

Η διαδοχή των φάσεων δεν γίνεται ούτε σταθερά ούτε γραμμικά, ενώ η εκτέλεσή τους μπορεί να γίνει είτε με τη φορά της σπείρας, είτε με την αντίθετη φορά, ανάλογα με το ρίσκο που λαμβάνεται και το οποίο αποτελεί θεμελιώδη έννοια στο σπειροειδές μοντέλο. Σε κάθε σπείρα που διανύεται υπάρχουν οι εξής φάσεις:

- Καθορισμός στόχων, εναλλακτικών λύσεων και υπολογισμός περιορισμών.
- Ανάλυση και υπολογισμός του ρίσκου και προσπάθεια μείωσης του.
- Ανάπτυξη και επαλήθευση ενδιάμεσου προϊόντος - εφόσον η προηγούμενη φάση δεν έδειξε κάποιο σοβαρό ρίσκο – πρόσθεση νέων λειτουργικών προδιαγραφών.
- Σχεδιασμός των επόμενων βημάτων.

Η βασική διαφορά του παραπάνω μοντέλου από τα υπόλοιπα είναι ότι σε αυτό υπολογίζεται πριν την έναρξη κάθε φάσης το ρίσκο.

Τα δυνατά σημεία του Σπειροειδούς Μοντέλου είναι ότι δίνει έμφαση σε εναλλακτικές λύσεις και περιορισμούς, ενώ ταυτόχρονα επιτρέπει την επαναχρησιμοποίηση του λογισμικού και την αύξηση της ποιότητάς του. Μέσω της Ανάλυσης Ρίσκου που μας παρέχει, μας δίνει λύσεις για το βαθμό που θα πραγματοποιηθεί ο έλεγχος ενώ δεν υπάρχει διάκριση ανάμεσα στην ανάπτυξη και τη συντήρηση του προϊόντος, καθώς η συντήρηση είναι απλά ένας καινούργιος δακτύλιος.

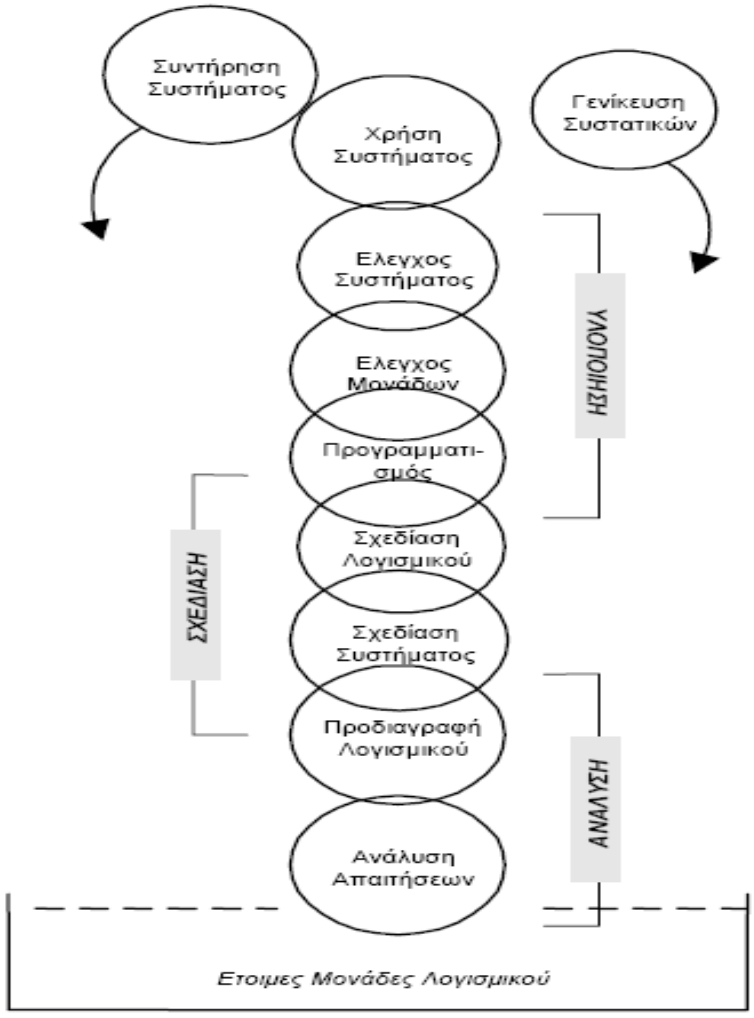
Το βασικό μειονέκτημα του μοντέλου είναι, πως εάν αποτύχει η ανάλυση ρίσκου, τότε το έργο διακόπτεται, για τον λόγο αυτό, η ομάδα υλοποίησης πρέπει να είναι εξαιρετικά ικανή στην Ανάλυση Ρίσκου, όπως επίσης ότι εφαρμόζεται μόνο σε λογισμικά μεγάλης κλίμακας.

Επιπρόσθετα, είναι δυνατόν να εισαχθούν νέες εργασίες που αφορούν την τεκμηρίωση της σκοπιμότητας και τον τμηματικό προγραμματισμό της ανάπτυξης και δεν ανήκουν καθαρά στις εργασίες ανάπτυξης λογισμικού, με απόρροια να επιφέρουν κάποιο παραπάνω κόστος, το οποίο μπορεί να αποσβεστεί με τον έγκαιρο εντοπισμό των προγραμμάτων.

3.2.6 Μοντέλο Πίδακα

Το μοντέλο του πίδακα προσεγγίζει τα επιθυμητά χαρακτηριστικά της Τεχνολογίας Λογισμικού σύμφωνα με την αντικειμενοστρεφή λογική. Τα αποτελέσματα κάθε διαδικασίας λογισμικού είναι επαναχρησιμοποιήσιμες μονάδες, οι οποίες μπορούν να χρησιμοποιηθούν από τις πρώτες φάσεις της ανάπτυξης μελλοντικών συστημάτων.

Το μοντέλο του πίδακα κατατμίζεται στις εξής φάσεις: ανάλυση, σχεδιασμός, υλοποίηση, οι οποίες επικαλύπτονται και στο τέλος της ανάπτυξης ορισμένα από τα συστατικά του λογισμικού που έχουν παραχθεί ενσωματώνονται σε μια δεξαμενή συστατικών και διατίθενται για να χρησιμοποιηθούν στην ανάπτυξη νέων συστημάτων.

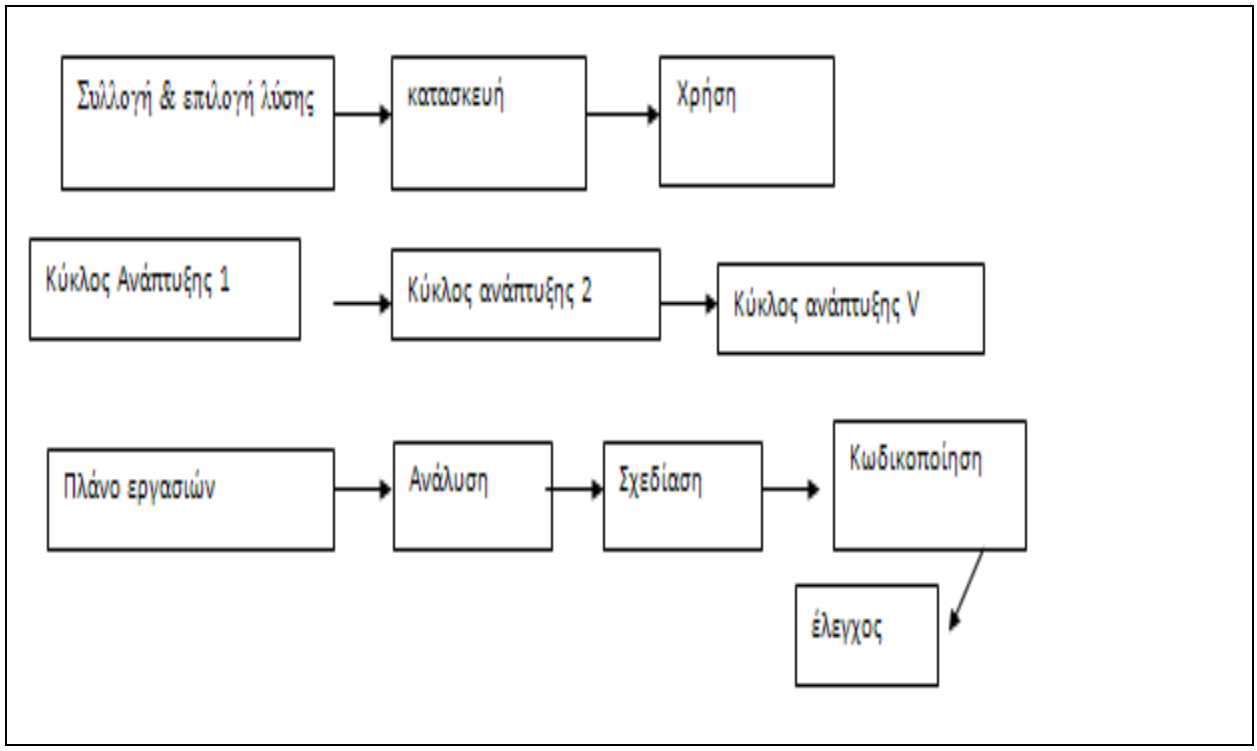


Εικόνα 7. Μοντέλο Πίδακα

Πλεονέκτημα του μοντέλου πίδακα είναι ότι κάθε τμήμα του μπορεί, αφού αποφασιστεί, να επαναχρησιμοποιηθεί και να αποτελέσει σημαντικό στάδιο υλοποίησης μελλοντικών συστημάτων.

3.2.7 Σύγχρονα Μοντέλα Κύκλου Ζωής Λογισμικού

Τα σύγχρονα μοντέλα κύκλου ζωής του λογισμικού δίνουν μόνο μια γενική κατεύθυνση εφαρμογής των υπαρχόντων ιδεών, ενώ παράλληλα αφήνουν σημαντικούς βαθμούς ελευθερίας στον κατασκευαστή που τα ακολουθεί.



Εικόνα 8. Σύγχρονο μοντέλο ανάπτυξης.

Το μοντέλο της λειτουργικής επαύξησης διαχωρίζεται σε 4 φάσεις, οι οποίες δεν ξεκινούν απαραίτητα από το ίδιο σημείο. Μετά το τέλος κάθε τμήματος έχουμε την ολοκλήρωση και στο επιμέρους μοντέλο, μπορούμε να εφαρμόσουμε άλλη μορφή ελέγχου, για παράδειγμα, το μοντέλο καταρράκτη. Αυτά είναι αυτόνομα κομμάτια και λειτουργούν μεταξύ τους.

Οι απαιτήσεις αυτού του μοντέλου είναι να υπάρχει το πρωτότυπο και αναθεωρώντας το αρχικό μοντέλο να μπορούμε να πάρουμε κάποιες τελικές αποφάσεις. Για το λόγο αυτό, κατασκευάζουμε διαβαθμίζοντας, λαμβάνοντας υπ’ όψιν μας τα ρίσκα των αποφάσεων.

Δεν επαναλαμβάνουμε τα προαναφερθέντα βήματα σε κάθε κύκλο, ενώ, σε κάθε πρωτότυπο μπορούμε να χρησιμοποιήσουμε οποιοδήποτε από τα προηγούμενα μοντέλα.

Η επανάληψη του συγκεκριμένου μοντέλου είναι έμμεση και οι κύκλοι δεν αναφέρονται στην αναθεώρηση του συνολικού κύκλου. Στο ξεκίνημα της κάθε σπείρας, αναλύουμε το επίπεδο του ρίσκου. Κάθε φορά που αποφασίζουμε να πάρουμε το ρίσκο, λαμβάνουμε υπ' όψιν τις αποφάσεις που έχουμε πάρει, συνεκτιμώντας τα πλεονεκτήματα και μειονεκτήματα των ρίσκων.

3.3 Σύγκριση μοντέλων κύκλου ζωής λογισμικό.

Στον παρακάτω πίνακα απεικονίζονται οι συγκρίσεις των μοντέλων κύκλου ζωής λογισμικού, σχετικά με το μέγεθος των εφαρμογών, τις μεταβολές στις απαιτήσεις και στη διάδοση τους. (Βεσκούκης, 2000)

ΜΟΝΤΕΛΟ	**ΜΕΓΕΘΟΣ ΕΦΑΡΜΟΓΩΝ**	**ΜΕΤΑΒΟΛΕΣ ΣΤΙΣ ΑΠΑΙΤΗΣΕΙΣ**	**ΔΙΑΔΟΣΗ**
Καταρράκτη	Μικρό-μεσαίο	Ανεπιθύμητες	Μεγάλη με τάση μείωσης
Πρωτοτυποποίησης	Μικρό-μεσαίο	Δεκτές	Μικρή με τάση αύξησης
Λειτουργικής επαύξησης	Μεσαίο-μεγάλο	Ανεπιθύμητες	Μεγάλη με τάση μείωσης
Σπειροειδές	Μεσαίο-μεγάλο	Δεκτές	Μεγάλη με τάση μείωσης
Πίδακα	Οποιοδήποτε	Δεκτές	Μικρή
Γενικά	Οποιοδήποτε	Δεκτές	Μικρή με ισχυρές τάσεις αύξησης.

Πίνακας 2. Σύγκριση μοντέλων κύκλου ζωής

Κεφάλαιο 4 Μεθοδολογίες Ανάπτυξης Λογισμικού

Σκοπός του κεφαλαίου είναι η γνωριμία των κύριων μεθοδολογιών ανάπτυξης λογισμικού, οι οποίες περιγράφουν τις εργασίες που πρέπει να γίνουν, κατά τη δημιουργία ενός συστήματος και θα δούμε με ποια κριτήρια επιλέγουμε την κατάλληλη μεθοδολογία, στο εκάστοτε έργο μας.

4.1 Ορισμός και φάσεις

Σε κάθε έργο και σε κάθε μεθοδολογία ανάπτυξης μπορούμε να ξεχωρίσουμε τέσσερεις θεμελιώδεις φάσεις.

- 1η φάση- Προγραμματισμός (Planning)
- 2η φάση- Ανάλυση (Analysis)
- 3η φάση- Σχεδίαση (Design)
- 4η φάση- Υλοποίηση (implementation)

Κάθε φάση αποτελείται από στάδια, όπου κάθε στάδιο βασίζεται σε τεχνικές και έχει παραδοτέα. Οι μεθοδολογίες που υπάρχουν είναι πολλές, κάθε μία έχει διαφορετικά στάδια ή διαφορετική σειρά σταδίων.

Οι μεθοδολογίες ανάπτυξης λογισμικού, παρέχουν συστηματική προσέγγιση στη διαδικασία του προγραμματισμού, ανάλυσης, σχεδίασης και κατασκευής ενός έργου, ουσιαστικά, αποτελείται από μια σειρά σταδίων που περιγράφουν συγκεκριμένες εργασίες που πρέπει να γίνουν. Είναι σημαντικό να τονίσουμε εξαρχής πως η γνωστή σε όλους UML, που θα αναλύσουμε εκτενέστερα στο παρακάτω κεφάλαιο, δεν αποτελεί μεθοδολογία ανάπτυξης λογισμικού, αλλά γλώσσα μοντελοποίησης και πως είναι ουδέτερη σε σχέση με τις μεθοδολογίες, καθώς χρησιμοποιείται με διάφορους τρόπους και έχει εφαρμογή σε οποιαδήποτε μεθοδολογία ανάπτυξης.

4.1.1 Πρώτη φάση- Προγραμματισμός (Planning)

Η πρώτη φάση του προγραμματισμού, μας απαντά στα αρχικά ερωτήματά μας για την κατασκευή του συστήματός μας, στο Γιατί και το Πώς, ενώ πραγματοποιείται η μελέτη σκοπιμότητας. Στην πορεία της φάσης αναλύεται η Διαχείριση του Έργου, η οποία διαρκεί όσο και το ίδιο το έργο, διότι, ο Συντονιστής του έργου (Project Manager), δημιουργεί και ενημερώνει το πλάνο εργασίας, ενώ παράλληλα παρακολουθεί και ελέγχει την πρόοδό του.
Παραδοτέο της Πρώτης φάσης είναι το Πλάνο του Έργου (Project Plan).

4.1.2 Δεύτερη φάση- Ανάλυση (Analysis)

Η φάση της Ανάλυσης προχωράει ακόμα πιο βαθιά στις ανάγκες των χρηστών για το σύστημα καθώς, μας απαντάει στα εξής ερωτήματα. Ποιός πρόκειται να χρησιμοποιήσει το σύστημα; Τί λειτουργίες θα έχει το σύστημα; Πού και πότε θα χρησιμοποιείται το σύστημα; Η φάση της ανάλυσης διαχωρίζεται σε δύο στάδια, τη Συλλογή των Απαιτήσεων, μέσω ερωτηματολογίων και συνεντεύξεων, και μέσα από την ανάλυση του υπάρχοντος συστήματος και των προβλημάτων που παρουσιάζει , αφού ολοκληρωθεί η Συλλογή των Απαιτήσεων, γίνεται η μετάβαση στο επόμενο στάδιο που είναι η Μοντελοποίηση των βασικών πλευρών του συστήματος. Παραδοτέο της Φάσης Ανάλυσης είναι το έγγραφο περιγραφής απαιτήσεων και μοντελοποίησης του συστήματος.

4.1.3 Τρίτη φάση- Σχεδίαση (Design)

Η φάση της Σχεδίασης είναι το τελευταίο στάδιο πριν ξεκινήσουν οι προγραμματιστές την υλοποίηση του συστήματος. Η φάση της ανάλυσης διαχωρίζεται στα εξής στάδια:

Στρατηγική Σχεδιασμού, όπου πραγματοποιείται η ανάπτυξη από τον ίδιο τον οργανισμό, ή εξωτερική ανάθεση, στην περίπτωση που η εταιρεία που έχει αναλάβει το έργο δεν έχει την δυνατότητα σχεδιαστικής υποστήριξης.

Αρχιτεκτονικός Σχεδιασμός, ο οποίος αφορά το υλικό, το λογισμικό, τη δικτυακή υποδομή, τη διεπαφή με τον χρήστη κ.α.

Περιγραφή Αρχείων και Βάσεων Δεδομένων, όπου πλέον γνωρίζουμε ποια ακριβώς είναι τα δεδομένα που θα αποθηκεύονται αλλά και το σημείο της αποθήκευσής τους.

Σχεδιασμός Προγραμμάτων, στο στάδιο αυτό, γνωρίζουμε ποια προγράμματα πρέπει να γραφτούν, καθώς και τις λειτουργίες που θα έχει το κάθε ένα.

Παραδοτέα της φάσης Σχεδίασης είναι οι Προδιαγραφές του Συστήματος, που στη συνέχεια δίνονται στην ομάδα των προγραμματιστών, έτσι ώστε να προχωρήσουν στην υλοποίηση.

4.1.4 Τέταρτη φάση- Υλοποίηση (implementation)

Η φάση της υλοποίησης είναι συνήθως το πιο χρονοβόρο και ακριβό τμήμα, καθώς πρέπει η ομάδα των προγραμματιστών να υλοποιήσει τα τρία στάδια της φάσης, που είναι, αρχικά, η κατασκευή του συστήματος, η εγκατάσταση και η μετάβαση από το παλαιό σύστημα στο νέο, είτε με απευθείας αντικατάσταση του, είτε τμηματικά, δουλεύοντας οι χρήστες παράλληλα και τα δύο συστήματα, ενώ ταυτόχρονα ξεκινάει η εκπαίδευσή τους. Το τελικό στάδιο είναι το πλάνο Υποστήριξης, κατά το οποίο γίνεται η αξιολόγηση του συστήματος μετά την εγκατάσταση και καταγράφονται οι αλλαγές που πρέπει να γίνουν. Το ερώτημα που γενάται, είναι πως σχετίζονται οι φάσεις με την έννοια της μεθοδολογίας;

Η μεθοδολογία μας προτείνει έναν τρόπο εργασίας, οργανώνει τις διαδικασίες που πρέπει να πραγματοποιηθούν καθοδηγώντας μας για τα βήματα που πρέπει να γίνουν και με ποια σειρά θα τα πραγματοποιήσουμε. Μεθοδολογία είναι μια τυπική προσέγγιση που υλοποιεί τον πλήρη κύκλο ζωής της ανάπτυξης του συστήματος, μας καθοδηγεί για τα παραδοτέα μας και για το πότε πρέπει να τα εξάγουμε αφού επικεντρώνεται στις διεργασίες, τα δεδομένα και τα αντικείμενα. Οι κατηγορίες των μεθοδολογιών είναι η δομημένη σχεδίαση (structured design), ταχεία ανάπτυξη εφαρμογής (rapid application development) και της ευκίνητης ανάπτυξης (agile development).

Όπως έχει γίνει φανερό μέχρι τώρα, η ανάπτυξη λογισμικού και γενικότερα η ανάπτυξη Πληροφοριακών Συστημάτων έγκειται στη σχεδίαση και τη δημιουργία ενός μηχανισμού που θα λύνει ένα συγκεκριμένο πρόβλημα σε συγκεκριμένο χρόνο και με συγκεκριμένο τρόπο.

Όπως λοιπόν, υπάρχουν αρκετές διαφορετικές μεθοδολογίες επίλυσης προβλημάτων, έτσι υπάρχουν και αρκετές μεθοδολογίες ανάπτυξης συστημάτων, που μάλιστα μπορεί και να διαφέρουν αρκετά μεταξύ τους.

4.2 Δομημένες μεθοδολογίες

Οι δομημένες μεθοδολογίες χρησιμοποιούνται για την τεκμηρίωση, ανάλυση, και σχεδιασμό συστημάτων πληροφοριών από τη δεκαετία του 1970 και συνεχίζουν ακόμη να είναι μια σημαντική μεθοδολογική προσέγγιση.

Ο όρος δομημένες μεθοδολογίες αναφέρεται στο γεγονός ότι οι τεχνικές αποτελούνται από βήματα, καθένα από τα οποία βασίζεται στο προηγούμενό του. Οι δομημένες μεθοδολογίες είναι αναλυτικές και προχωρούν από το ανώτερο, πιο αφηρημένο, επίπεδο στο κατώτερο επίπεδο λεπτομέρειας - από το γενικό στο ειδικό.

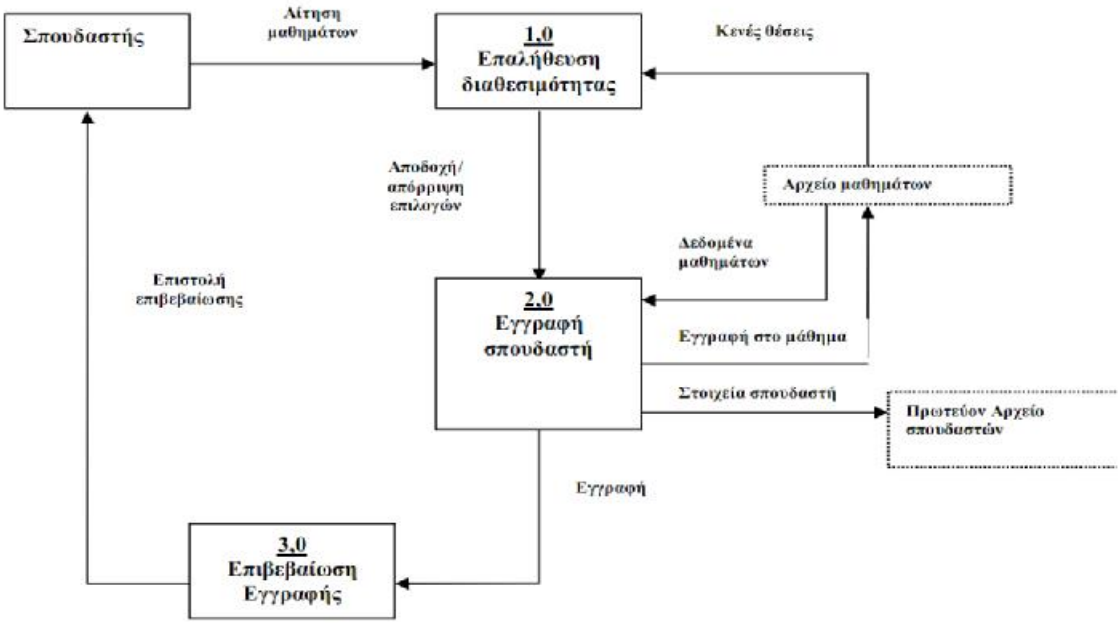
Για παράδειγμα, το ανώτερο επίπεδο μιας αναλυτικής περιγραφής ενός συστήματος ανθρώπινων πόρων θα έδειχνε τις κύριες λειτουργίες ανθρώπινων πόρων: προσωπικό, πρόσθετες παροχές, αμοιβές, και την πολιτική ίσων ευκαιριών. Κάθε μία από αυτές θα αναλυθεί περισσότερο στο επόμενο επίπεδο.

Οι πρόσθετες παροχές, για παράδειγμα, μπορεί να περιλαμβάνουν συνταξιοδοτικό πρόγραμμα, αποταμιευτικό πρόγραμμα, υγειονομική περίθαλψη, και ασφάλιση. Καθένα από αυτά τα επίπεδα αναλύεται με τη σειρά του περισσότερο μέχρι το κατώτερο επίπεδο λεπτομέρειας.

Οι παραδοσιακές δομημένες μεθοδολογίες είναι προσανατολισμένες προς τις διεργασίες και όχι προς τα δεδομένα. Αν και οι περιγραφές των δεδομένων αποτελούν μέρος των μεθόδων, οι μεθοδολογίες εστιάζουν στον τρόπο μετασχηματισμού των δεδομένων αντί στα ίδια τα δεδομένα. Αυτές οι μεθοδολογίες είναι κατά το μεγαλύτερο μέρος γραμμικές - κάθε φάση πρέπει να ολοκληρωθεί πριν αρχίσει η επόμενη. Οι δομημένες μεθοδολογίες περιλαμβάνουν τη δομημένη ανάλυση, το δομημένο σχεδιασμό και τη χρήση διαγραμμάτων ροής.

4.2.1 Δομημένη ανάλυση

Η δομημένη ανάλυση χρησιμοποιείται σε μεγάλη κλίμακα για τον ορισμό των εισόδων, διεργασιών, και εξόδων ενός συστήματος, ενώ διαθέτει ένα λογικό μοντέλο γραφικών για απεικόνιση της ροής πληροφοριών, με το οποίο ένα σύστημα χωρίζεται σε λειτουργικές μονάδες εύλογου βαθμού λεπτομέρειας. Προδιαγράφει επακριβώς τις διεργασίες ή τους μετασχηματισμούς που συμβαίνουν μέσα σε κάθε λειτουργική μονάδα, καθώς και τις διασυνδέσεις που υπάρχουν μεταξύ τους. Το κύριο εργαλείο της είναι το διάγραμμα ροής δεδομένων (DFD), μια γραφική παράσταση των διεργασιών που αποτελούν το σύστημα και της ροής των δεδομένων μεταξύ τους.



Εικόνα 9. Διάγραμμα ροής δεδομένων δομημένης ανάλυσης.

Στην παραπάνω εικόνα φαίνεται ένα απλό διάγραμμα ροής δεδομένων ενός συστήματος ταχυδρομικής εγγραφής σε κύκλους μαθημάτων ενός πανεπιστημίου. Τα στρογγυλευμένα πλαίσια παριστάνουν διεργασίες στις οποίες γίνεται μετασχηματισμός δεδομένων. Το ορθογώνιο πλαίσιο είναι μια εξωτερική οντότητα δημιουργίας ή αποδοχής πληροφοριών που βρίσκεται έξω από τα όρια του συστήματος που σχεδιάζεται. Τα ανοικτά ορθογώνια παριστάνουν αποθήκες δεδομένων, χειρογραφικές ή αυτοματοποιημένες. Τα βέλη δείχνουν ροές πληροφοριών μεταξύ διεργασιών, εξωτερικών οντοτήτων, και αποθηκών δεδομένων. Αυτές οι ροές πληροφοριών περιέχουν πάντοτε πακέτα δεδομένων και το όνομα του περιεχομένου τους αναγράφεται δίπλα στο βέλος.

Αυτό το διάγραμμα ροής δείχνει ότι οι σπουδαστές στέλνουν έντυπες φόρμες αιτήσεων εγγραφής, με το όνομα τους, τον αριθμό της ταυτότητας τους, και τους αριθμούς των μαθημάτων που θέλουν να παρακολουθήσουν. Στη διεργασία 1.0, το σύστημα επαληθεύει την ύπαρξη κενών θέσεων ανατρέχοντας στο αρχείο μαθημάτων του πανεπιστημίου. Αυτό το αρχείο διακρίνει τα μαθήματα που έχουν κενές θέσεις από εκείνα που έχουν ακυρωθεί ή είναι πλήρη. Η

διεργασία 1.0 καθορίζει στην πορεία ποιες από τις επιλογές του σπουδαστή μπορούν να γίνουν αποδεκτές και ποιες θα απορριφθούν.

Η διεργασία 2.0 γράφει το σπουδαστή στα μαθήματα όπου έγινε δεκτός. Στη συνέχεια, ενημερώνει το αρχείο μαθημάτων με το όνομα και τον αριθμό ταυτότητας του σπουδαστή και επαναϋπολογίζει το μέγεθος της τάξης. Μόλις φτάσει στο μέγιστο αριθμό εγγραφών, το μάθημα επισημαίνεται ως πλήρες. Η διεργασία 2.0 ενημερώνει επίσης το πρωτεύον αρχείο σπουδαστών του πανεπιστημίου με πληροφορίες νέων σπουδαστών ή αλλαγές σε διευθύνσεις.

Η διεργασία 3.0 στέλνει σε κάθε σπουδαστή που έκανε αίτηση μια επιστολή επιβεβαίωσης της εγγραφής που αναφέρει τα μαθήματα στα οποία έγινε δεκτός και εκείνα για τα οποία η ικανοποίηση της επιλογής του ήταν αδύνατη.

Διαγράμματα μπορούν επίσης να χρησιμοποιηθούν για την παρουσίαση διεργασιών υψηλότερου επιπέδου και Λεπτομερειών χαμηλότερου επιπέδου. Με τα διαγράμματα ροής δεδομένων με επίπεδα, μια σύνθετη διεργασία μπορεί να αναλυθεί σε διαδοχικά επίπεδα λεπτομέρειας.

Ένα ολόκληρο σύστημα μπορεί να χωριστεί σε υποσυστήματα με τη βοήθεια ενός διαγράμματος ροής δεδομένων υψηλού επιπέδου. Κάθε υποσύστημα, με τη σειρά του, μπορεί να αναλυθεί σε άλλα υποσυστήματα με διαγράμματα ροής δεδομένων δεύτερου επιπέδου, και τα υποσυστήματα κατώτερου επιπέδου μπορούν να αναλυθούν περισσότερο μέχρι το κατώτερο επίπεδο λεπτομέρειας.

Ένα απλό εργαλείο της δομημένης ανάλυσης είναι το λεξικό δεδομένων (data dictionary), το οποίο περιέχει πληροφορίες για τα επιμέρους στοιχεία δεδομένων και ομάδες δεδομένων ενός συστήματος. Το Λεξικό δεδομένων ορίζει το περιεχόμενο των ροών δεδομένων και τις αποθήκες δεδομένων, έτσι ώστε οι κατασκευαστές του συστήματος να γνωρίζουν ακριβώς τα δεδομένα που περιέχουν. Οι προδιαγραφές των διεργασιών περιγράφουν τους μετασχηματισμούς που συμβαίνουν στα κατώτερα επίπεδα των διαγραμμάτων ροής δεδομένων και εκφράζουν τη Λογική κάθε διεργασίας.

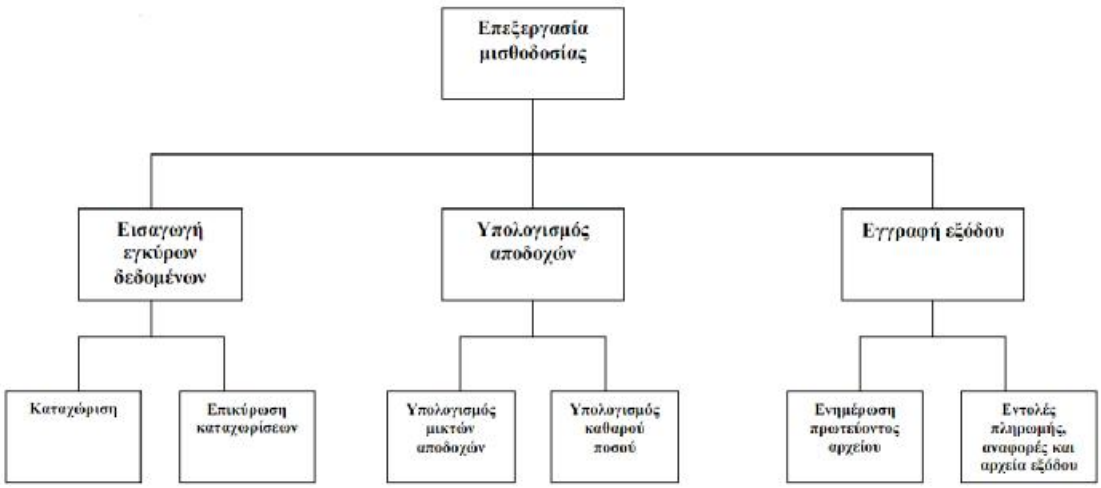
4.2.2 Δομημένος σχεδιασμός

Ο δομημένος σχεδιασμός περιλαμβάνει ένα σύνολο κανόνων και τεχνικών σχεδιασμού που συμβάλουν στη σαφήνεια και την απλότητα του προγράμματος και, με αυτόν τον τρόπο, μειώνουν το χρόνο και την προσπάθεια που απαιτείται για την κωδικοποίηση, την αποσφαλμάτωση, και τη συντήρηση.

Η κύρια αρχή του δομημένου σχεδιασμού είναι ότι ένα σύστημα θα πρέπει να σχεδιάζεται αναλυτικά με ιεραρχικό τρόπο και να αναλύεται σε μεγαλύτερο βαθμό λεπτομέρειας. Ο σχεδιασμός θα πρέπει να ασχολείται πρώτα με την κύρια λειτουργία του προγράμματος ή του συστήματος και κατόπιν να χωρίζει αυτήν τη λειτουργία σε υπολειτουργίες και να αναλύει κάθε υπολειτουργία περισσότερο μέχρι το κατώτερο επίπεδο Λεπτομέρειας. Οι λειτουργικές μονάδες του τελευταίου επιπέδου περιγράφουν την επεξεργασία που θα γίνεται στην πραγματικότητα.

Με αυτόν τον τρόπο, ολόκληρη η λογική υψηλού επιπέδου και το μοντέλο σχεδιασμού αναπτύσσονται πριν αρχίσει να γράφεται ο λεπτομερής κώδικας του προγράμματος. Αν έχει προηγηθεί η δομημένη ανάλυση, η δομημένη τεκμηρίωση των προδιαγραφών μπορεί να χρησιμεύσει ως είσοδος στη διαδικασία σχεδιασμού. Η προηγούμενη αναλυτική περιγραφή του συστήματος ανθρώπινων πόρων είναι ένα καλό παράδειγμα δομημένου σχεδιασμού. Καθώς προχωράει ο σχεδιασμός, η τεκμηρίωση του γίνεται σε ένα διάγραμμα δομής.

Το διάγραμμα δομής είναι ένα ιεραρχικό διάγραμμα το οποίο παρουσιάζει κάθε επίπεδο σχεδιασμού, τις σχέσεις του με τα άλλα επίπεδα, και τη θέση του στο συνολικό δομημένο σχεδιασμό.



Εικόνα 10. Διάγραμμα ροής δεδομένων δομημένου σχεδιασμού.

Η παραπάνω εικόνα δείχνει ένα διάγραμμα δομής υψηλού επιπέδου ενός συστήματος μισθοδοσίας. Αν ένα σχέδιο συστήματος έχει τόσα πολλά επίπεδα, ώστε να μη χωράει σε ένα διάγραμμα δομής, είναι δυνατό αυτό να αναλυθεί σε πιο λεπτομερή διαγράμματα δομής. Ένα διάγραμμα δομής μπορεί να τεκμηριώνει ένα πρόγραμμα, ένα σύστημα (σύνολο προγραμμάτων) ή ένα τμήμα προγράμματος.

4.2.3 Ενναλακτικές μέθοδοι ανάπτυξης συστημάτων

Οι εναλλακτικές μέθοδοι ανάπτυξης συστημάτων μπορούν να αντιμετωπίσουν μερικούς από τους περιορισμούς του παραδοσιακού κύκλου ζωής συστημάτων. Σε αυτές τις μεθόδους περιλαμβάνονται η δημιουργία πρωτοτύπων, τα πακέτα λογισμικού εφαρμογών, η ανάπτυξη από τους τελικούς χρήστες, και η εξωτερική ανάθεση. Ενδεικτικά θα αναφέρουμε τη μεθοδολογία του πρωτοτύπου.

4.3 Η μέθοδος της ταχείας ανάπτυξης εφαρμογών (RAD)

Είναι η μέθοδος κύκλου ζωής με έμφαση σε έναν πολύ μικρό κύκλο ανάπτυξης. Είναι δηλαδή μια υψηλής ταχύτητας προσαρμογή του μοντέλου του καταρράκτη.

Προσανατολίζεται δε στις παρακάτω φάσεις:

- Επιχειρησιακή διαμόρφωση
- Διαμόρφωση στοιχείων
- Διαμόρφωση διαδικασίας
- Υλοποίηση εφαρμογής
- Δοκιμές και επαλήθευση.

Η μέθοδος της ταχείας ανάπτυξης αναλύεται και σχεδιάζεται με ειδικές τεχνικές και εργαλεία, έτσι ώστε να επιταχυνθεί όλη η διαδικασία. Χρησιμοποιεί εργαλεία case, γλώσσες οπτικού προγραμματισμού, όπως η Visual basic, ενώ έχει γεννήτριες κώδικα, οι οποίες παράγουν τον κώδικα από το σχέδιο.

Θα πρέπει να σημειωθεί ότι η μέθοδος αυτή προϋποθέτει την δημιουργία αποτελεσματικών ομάδων εργασίας καθώς και την δέσμευση των δύο πλευρών για άμεσες ενέργειες αναγκαίες για την συμπλήρωση του συστήματος.

4.3.1 Agile

Στις 11 έως 13 Φεβρουαρίου 2001, σε ένα χειμερινό θέρετρο της οροσειράς Wasatch της Utah, συγκεντρώθηκαν εκπρόσωποι διάφορων νέων μεθοδολογιών, ώστε να συζητήσουν την ανάγκη νέων ελαφρότερων μεθοδολογιών ως εναλλακτικές των παραδοσιακών επιβαρυμένων.

Οι συζητήσεις κατέληξαν σε ένα μανιφέστο βασικών αρχών, οι οποίες αποτέλεσαν τα θεμέλια της ευέλικτης ανάπτυξης λογισμικού. Πιθανότατα, η πιο γνωστή ευέλικτη μέθοδος είναι ο "ακραίος προγραμματισμός" (extreme programming) , ο οποίος περιγράφεται παρακάτω. Ωστόσο, υπάρχουν κι άλλες μέθοδοι όπως: Scrum, Crystal, Adaptive Software Development, DSDM και Feature Driven Development.

Η επιτυχία αυτών των μεθόδων οδήγησε στην ένταξή τους σε πιο παραδοσιακές μεθόδους ανάπτυξης λογισμικού που βασίζονται στην μοντελοποίηση του συστήματος.

Οι ευέλικτες μέθοδοι εισάγουν μία διαφορετική προσέγγιση στον τρόπο ανάπτυξης του λογισμικού. Η προσέγγιση αυτή αφορά την στροφή από τον παραδοσιακό τρόπο, τον προσανατολισμένο στις διαδικασίες και την πλήρη τεκμηρίωση, στον ευέλικτο τρόπο, που είναι προσανατολισμένος στον άνθρωπο και την λιγότερη τεκμηρίωση (Beck, 2000-2006).

Ακολουθούν τον επαναληπτικό και αυξητικό τρόπο ανάπτυξης και επιτρέπουν την προσαρμοστικότητα και τη μεταβλητότητα, εστιάζοντας πάντα στα νέα στοιχεία, που προκύπτουν κατά την διαδικασία της ανάπτυξης. Η ικανότητα για γρήγορες και συχνές αλλαγές και για γρήγορη παράδοση λογισμικού σημαίνει και την άμεση προσαρμογή της ομάδας (Abrahamsson et al, 2002).

Μια προσαρμόσιμη ομάδα μπορεί να προγραμματίσει επακριβώς τις εργασίες που πρέπει να γίνουν την επόμενη εβδομάδα ή το πολύ μέσα στον τρέχοντα μήνα. Από την άλλη πλευρά οι παραδοσιακές μέθοδοι, σαν προφητικές μέθοδοι, εστιάζουν στο μέλλον σχεδιάζοντας με κάθε λεπτομέρεια τις μελλοντικές εξελίξεις. Μία ομάδα που ακολουθεί αυτή τη μεθοδολογία, μπορεί να αναφέρει με κάθε λεπτομέρεια ποια χαρακτηριστικά και ποιες εργασίες έχουν σχεδιαστεί για ολόκληρη την διαδικασία ανάπτυξης. Οι ομάδες αυτές δύσκολα αλλάζουν κατεύθυνση. Αν

συμβεί κάποια αλλαγή, τότε το όλο πλάνο του έργου θα πρέπει να αλλάξει. Οπότε όση δουλειά έγινε μέχρι το χρονικό εκείνο σημείο, θα πρέπει να επαναληφθεί.

Αυτό το μεγάλο μειονέκτημα των παραδοσιακών μεθόδων είναι η αιτία της κυριαρχίας των ευέλικτων μεθόδων, ιδιαίτερα στα έργα χαμηλού ρίσκου.

Ο ευέλικτος τρόπος ανάπτυξης λογισμικού θεωρείται καλύτερος, όταν οι ομάδες εργασίας είναι μικρές. Οι περισσότερες ομάδες είναι μέχρι 12 άτομα ή και σε κάποιες περιπτώσεις μέχρι 25. Όσο μεγαλώνει ο αριθμός των ατόμων της ομάδας, δυσκολεύει ο συντονισμός και η επικοινωνία των ατόμων. Η διαπροσωπική επικοινωνία γίνεται δύσκολη και περίπλοκη σε ομάδες που υπερβαίνουν τα 20-40 μέλη. Σε αυτή την περίπτωση είναι απαραίτητη η εφαρμογή κάποιων στρατηγικών κατάτμησης και διαμοίρασης των εργασιών σε περισσότερες υπό-ομάδες.

Ένας άλλος παράγοντας, που επηρεάζει την επιτυχή εφαρμογή των ευέλικτων μεθόδων, είναι η επιλογή των ατόμων της ομάδας. Η ομάδα θα πρέπει να διαθέτει έναν αριθμό από ικανά και πεπειραμένα στελέχη. Ικανά στελέχη σημαίνει να είναι άτομα με εμπειρία από τον πραγματικό κόσμο σε τεχνολογικά θέματα, να έχουν φτιάξει παρόμοια συστήματα στο παρελθόν και να διαθέτουν προσόντα καλής επικοινωνίας. Η ομάδα πρέπει να είναι αποτελεσματική ως προς την ποιότητα των ατόμων αλλά και ως προς τον τρόπο συνεργασίας και επικοινωνίας (Cockburn, 2002).

Οι οργανισμοί, που εφαρμόζουν ευέλικτες μεθόδους, θα πρέπει να διαθέτουν ένα περιβάλλον, το οποίο να διευκολύνει την γρήγορη επικοινωνία μεταξύ των μελών της ομάδας. Στην ομάδα συμπεριλαμβάνεται και ο πελάτης, που παρέχει άμεση ανατροφοδότηση. Ουσιαστικά οι ευέλικτες μέθοδοι βασίζονται στην στενή συνεργασία με τον πελάτη, ο οποίος πρέπει να είναι πάντα σε ετοιμότητα. Ο αριθμός των στελεχών, οι ικανότητές τους, οι ρόλοι τους καθώς και ο ρόλος του πελάτη στην ομάδα αποτελούν θέματα συζητήσεων και έρευνας τα τελευταία χρόνια.

Η νέα θεώρηση στον τρόπο ανάπτυξης δεν αφορά τόσο τις νέες πρακτικές, όσο την αναγνώριση του ανθρώπινου παράγοντα ως τον κύριο παράγοντα της επιτυχίας. Οι διευθυντές των έργων πρέπει να είναι ικανοί να καταλαβαίνουν το προσωπικό τους όχι σαν άτομα, αλλά σαν μέλη της ομάδας. Αν δεν λαμβάνονται υπόψη οι ικανότητες και οι περιορισμοί των προγραμματιστών,

τότε δεν είναι δυνατόν να εφαρμοστούν σωστά και με αποτελεσματικό τρόπο οι πρακτικές των ευέλικτων μεθόδων.

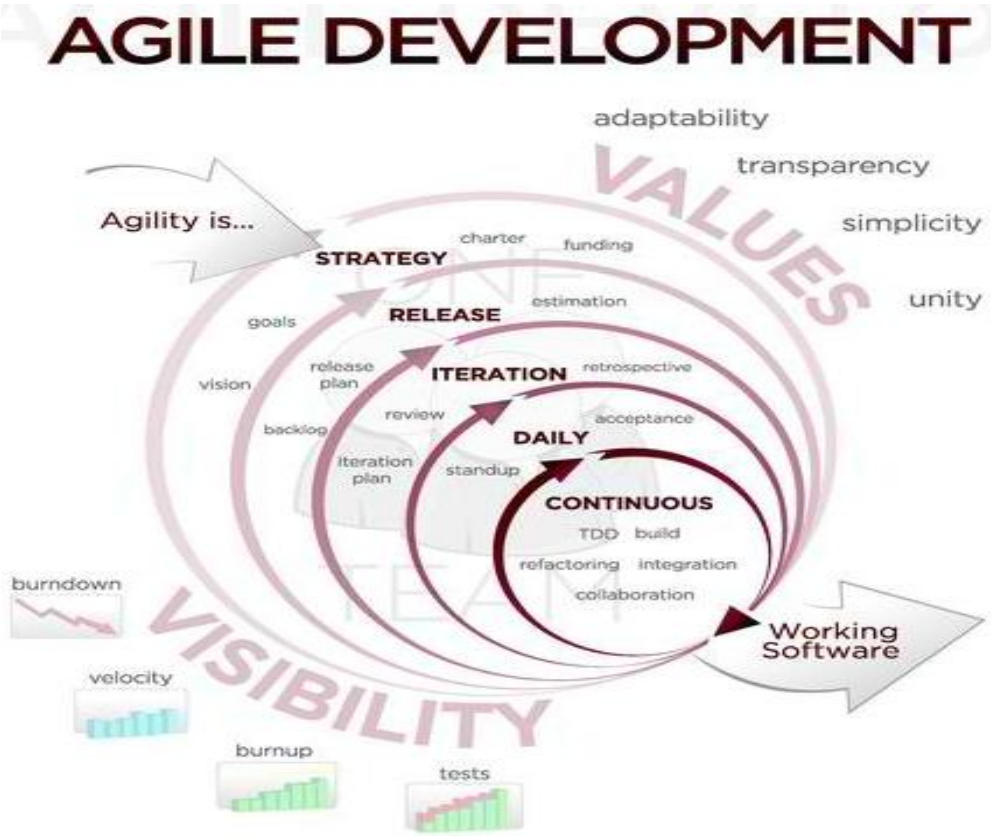
Στη σωστή διοίκηση ανάγεται και η έγκαιρη αναγνώριση προειδοποιητικών σημάτων, που υποδεικνύουν την ύπαρξη προβλημάτων στην διαδικασία ανάπτυξης. Το ερώτημα που τίθεται είναι πώς θα γνωρίζει η διοίκηση πότε να λάβει διορθωτικά μέτρα για την ελαχιστοποίηση των κινδύνων.

Μια καλή τακτική είναι οι καθημερινές συνεδριάσεις (stand-up meetings), οι οποίες παρέχουν έναν χρήσιμο τρόπο μέτρησης των προβλημάτων, μέσα από την συζήτηση που γίνεται μεταξύ των μελών της ομάδας. Επίσης ένα προειδοποιητικό σήμα για τον διαχειριστή του έργου είναι όταν η ομάδα μένει πίσω στις προγραμματισμένες επαναλήψεις. Γι' αυτό είναι πολύ σημαντικό να υπάρχουν συχνές επαναλήψεις, ώστε να μπορεί να ελέγχεται αυτό το προειδοποιητικό σήμα.

Υπάρχει μία μερίδα επιστημόνων, που αντιμετωπίζει τις ευέλικτες μεθόδους με σκεπτικισμό και πολλές φορές παρερμηνεύει τις βασικές αρχές και πρακτικές που εισάγουν. Η ανταπόκριση στις αλλαγές αντί ακολουθούμενου σχεδίου, το οποίο είναι μία από τις τέσσερεις προτροπές των ευέλικτων μεθόδων, παρερμηνεύεται σαν έλλειψη σχεδιασμού στην ανάπτυξη και συντελεί στην αύξηση των κινδύνων.

Αυτό όμως δεν ισχύει, καθότι ο σχεδιασμός για κάθε επαναληπτικό κύκλο, είναι πλήρης και απόλυτα ακριβής, επιτρέποντας την διασπορά των κινδύνων αλλά και την άμεση ανα-τροφοδότηση, που είναι ένας μηχανισμός μείωσης των κινδύνων. Επίσης αμφισβητείται κατά πόσο όλοι οι κατασκευαστές μπορούν να εφαρμόζουν σωστά τις πρακτικές τις οποίες εισάγουν οι ευέλικτες μέθοδοι, επειδή απαιτείται μια τελείως διαφορετική κουλτούρα πάνω στον τρόπο εργασίας και σκέψης, καθώς και για έλλειψη τεκμηρίωσης.

Όλες αυτές οι παρερμηνείες και αμφισβητήσεις αποτελούν στόχο έρευνας, η οποία έχει ξεκινήσει τα τελευταία χρόνια, ώστε να δοθούν τεκμηριωμένες απαντήσεις στον τρόπο χρήσης και στα πλεονεκτήματα που ευαγγελίζονται οι ευέλικτες μέθοδοι. Σε αυτό το πλαίσιο εντάξαμε την ερευνητική μας προσπάθεια (AgileAlliance, 2001).



Εικόνα 11. Η Agile μεθοδολογία, σχηματικά.

Ευελιξία (agility): είναι η ικανότητα της προσαρμογής και επαναπροσδιορισμού ενός αναπτυσσόμενου και συνεχώς εξελισσόμενου συστήματος, στην περίπτωση που εμφανίζονται αλλαγές στις αρχικές θεωρήσεις και παραδοχές. Οι οργανισμοί, που χρησιμοποιούν ευέλικτες μεθόδους, βλέπουν την αλλαγή σαν ευκαιρία βελτίωσης και προόδου και όχι σαν απειλή. Οι ευέλικτες μέθοδοι είναι:

Επαναληπτικές (iterative): Αρχικά παραδίδεται ένα πλήρες σύστημα και στη συνέχεια γίνονται αλλαγές στη λειτουργία κάθε υποσυστήματος σε κάθε νέα έκδοση.

Αυξητικές (incremental): Το σύστημα, όπως ορίζεται στις απαιτήσεις, χωρίζεται σε υποσυστήματα με βάση τη λειτουργία τους. Νέες λειτουργίες προστίθενται σε κάθε νέα έκδοση.

Αυτο-διοργανωμένες (self-organizing): Η ομάδα έχει την αυτονομία να οργανώνεται έτσι, ώστε να επιτύχει την βέλτιστη ολοκλήρωση των υποσυστημάτων με τον καλύτερο δυνατό τρόπο.

Προκύπτουσες (emergent): Οι απαιτήσεις και η τεχνολογία, που θα χρησιμοποιηθεί, προκύπτουν κατά τη διάρκεια του κύκλου ανάπτυξης

Οι ευέλικτες μέθοδοι ορίζονται από τις αρχές που παρουσιάζονται στον παρακάτω πίνακα:

Αρχή	Περιγραφή
Ανάμειξη πελάτη	Οι πελάτες θα πρέπει να είναι άμεσα αναμεμειγμένοι με τη διεργασία της ανάπτυξης λογισμικού. Ο ρόλος τους είναι να παρέχουν νέες απαιτήσεις, να τους δίνουν προτεραιότητες και να αξιολογούν τις επαναληπτικές εκδόσεις του συστήματος.
Αυξανόμενη παράδοση	Το λογισμικό αναπτύσσεται σε μέρη και οι πελάτες ορίζουν τις απαιτήσεις που θα περιληφθούν σε κάθε μέρος.
Άνθρωποι, όχι διεργασίες	Οι ικανότητες της ομάδας ανάπτυξης θα πρέπει να αναγνωρίζονται και να εκμεταλλεύονται. Τα μέλη της ομάδας θα πρέπει να έχουν τη δυνατότητα να εργάζονται με τον δικό τους τρόπο και όχι στα πλαίσια μιας ορισμένης διεργασίας.
Αποδοχή αλλαγών	Θα πρέπει να αναμένονται αλλαγές στις απαιτήσεις του συστήματος, για αυτό το λόγο το σύστημα πρέπει να σχεδιαστεί ώστε να δέχεται

Αρχή	Περιγραφή
	τέτοιες αλλαγές.
Διατήρηση απλότητας	Θα πρέπει να γίνεται εστίαση στην απλότητα τόσο του λογισμικού που αναπτύσσεται όσο και της διαδικασίας της ανάπτυξης. Όποτε είναι δυνατό, εκτελούνται εργασίες για την εξάλειψη πολυπλοκότητας από το σύστημα.

Πίνακας 3. Αρχές ευέλικτων μεθόδων

Στην πραγματικότητα, οι αρχές στις οποίες βασίζονται οι ευέλικτες μέθοδοι είναι πολλές φορές δύσκολο να πραγματοποιηθούν. Παρόλο που η ιδέα της ανάμειξης του πελάτη στη διαδικασία ανάπτυξης του λογισμικού είναι ελκυστική, η επιτυχία της εξαρτάται από την επιθυμία και τη δυνατότητα του πελάτη να διαθέσει χρόνο στην ομάδα ανάπτυξης.

Συχνά, οι πελάτες έχουν άλλες υποχρεώσεις και δεν μπορούν να έχουν πλήρη συμμετοχή στην ανάπτυξη του λογισμικού. Κάποια άτομα, μέλη μιας ομάδας, μπορεί να μην έχουν τον κατάλληλο χαρακτήρα και προσωπικότητα για συμμετέχουν εντατικά σε μία ομάδα, κάτι που είναι τυπικό στις ευέλικτες μεθόδους. Για αυτό το λόγο, μπορεί να μην συνεργάζονται σωστά με τα άλλα μέλη της ομάδας.

Το να θέτεις προτεραιότητες στις αλλαγές του συστήματος μπορεί να γίνει δύσκολο, ειδικά για συστήματα που υπάρχουν περισσότερα του ενός άτομα που θέτουν τις προτεραιότητες για τις αλλαγές του συστήματος. Συχνά, κάθε άτομο θέτει διαφορετικές προτεραιότητες σε διαφορετικές αλλαγές.

Η διατήρηση της απλότητας απαιτεί περισσότερη εργασία. Υπό την πίεση των προθεσμιών παράδοσης του συστήματος, η ομάδα μπορεί να μην έχει το χρόνο να κάνει τις επιθυμητές απλοποιήσεις στο σύστημα.

Ένα άλλο πρόβλημα, μη τεχνικό, το οποίο είναι γενικό πρόβλημα των μεθόδων της αυξανόμενης ανάπτυξης και παράδοσης, προκύπτει όταν ο πελάτης χρησιμοποιεί έναν εξωτερικό οργανισμό για την ανάπτυξη του συστήματος. Το κείμενο των προδιαγραφών του συστήματος συνήθως αποτελεί μέρος του συμβολαίου μεταξύ του πελάτη και του οργανισμού που το αναπτύσσει.

Λόγω του ότι ο αυξανόμενος ορισμός των απαιτήσεων αποτελεί μέρος των ευέλικτων μεθόδων, η συγγραφή συμβολαίων για αυτό τον τύπο ανάπτυξης μπορεί να είναι δύσκολη.

Συνεπώς, οι ευέλικτες μέθοδοι βασίζονται στα συμβόλαια, στα οποία ο πελάτης πληρώνει για το χρόνο που απαιτείται για την ανάπτυξη του λογισμικό και όχι για ένα συγκεκριμένο σύνολο απαιτήσεων. Εφόσον όλα πάνε καλά, αυτό ωφελεί και τον πελάτη και τον δημιουργό του συστήματος.

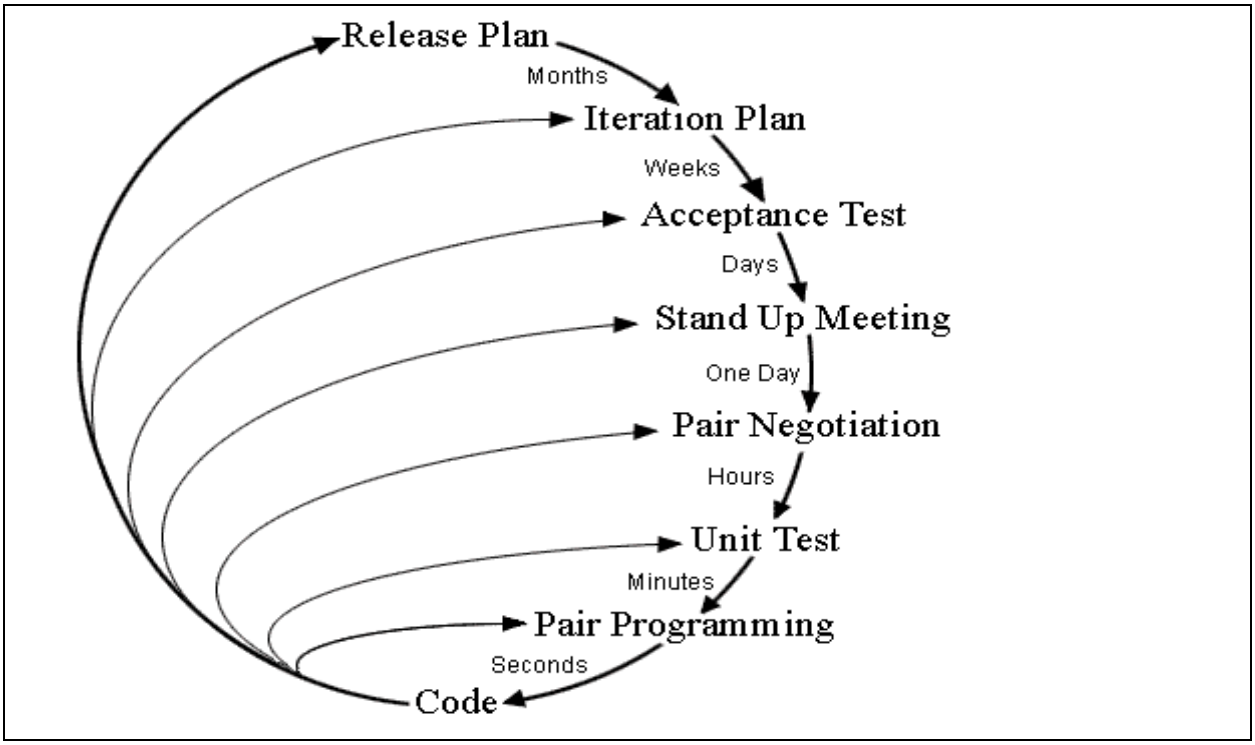
Ωστόσο, αν προκύψουν προβλήματα, θα προκύψουν διαφωνίες, που δύσκολα επιλύονται, για το ποιος ευθύνεται και ποιος πρέπει να πληρώσει για τον επιπλέον χρόνο και τους πόρους που χρειάστηκαν για να επιλυθούν τα προβλήματα.

4.3.1.1 Extreme Programming

Ο ακραίος προγραμματισμός (Extreme programming --- XP*)* είναι ίσως η πιο γνωστή και η πιο διαδεδομένη από τις ευέλικτες μεθόδους (Sommerville, 2001). Βαφτίστηκε έτσι από τον Beck γιατί αυτή η μέθοδος δημιουργήθηκε χρησιμοποιώντας καλές και αναγνωρισμένες πρακτικές, όπως η επαναλαμβανόμενη ανάπτυξη, και η ανάμειξη του πελάτη σε *υπερβολικό* (extreme) βαθμό.

Στην μέθοδο XP, όλες οι απαιτήσεις εκφράζονται ως σενάρια, τα οποία υλοποιούνται άμεσα ως μία σειρά εργασιών.

Οι προγραμματιστές εργάζονται ανά δύο και δοκιμάζουν κάθε εργασία πριν γράψουν τον κώδικα. Όλες οι δοκιμές πρέπει να ολοκληρωθούν με επιτυχία πριν ενσωματωθεί ο κώδικάς τους στο σύστημα. Μεσολαβεί μικρό χρονικό διάστημα μεταξύ των διαφορετικών εκδόσεων του συστήματος.



Εικόνα 12. Το Extreme Programming σχηματικά.

Η μέθοδος XP περιλαμβάνει ένα αριθμό πρακτικών που αντιστοιχούν στις αρχές των ευέλικτων μεθόδων ανάπτυξης:

- Η αυξανόμενη ανάπτυξη στηρίζεται στις μικρές και συχνές εκδόσεις του συστήματος, καθώς και σε μία προσέγγιση όπου η περιγραφή των απαιτήσεων βασίζεται στα σενάρια που παρέχει ο πελάτης και αποτελούν τη βάση για τον σχεδιασμό της διεργασίας.

- Η ανάμειξη του πελάτη στηρίζεται στην πλήρη δέσμευση του πελάτη στην ομάδα ανάπτυξης. Ο εκπρόσωπος του πελάτη ή ο ίδιος ο πελάτης παίρνει μέρος στην ανάπτυξη του συστήματος και είναι υπεύθυνος στο να καθορίζει αποδεκτές δοκιμές του συστήματος.

- Η άποψη άνθρωποι, όχι διεργασίες στηρίζεται στον προγραμματισμό ανά ζεύγη, τη συλλογική ιδιοκτησία του κώδικα του συστήματος και τον υποφερτό ρυθμό ανάπτυξης που δεν περιλαμβάνει υπερβολικές ώρες εργασίας.

- Η αλλαγή στηρίζεται στις συχνές εκδόσεις του συστήματος, την ανάπτυξη μετά τον έλεγχο και τη συνεχή συνένωση.

- Η διατήρηση της απλότητας στηρίζεται στις συνεχείς αλλαγές του κώδικα χάριν της βελτίωσης της ποιότητάς του και στη χρήση απλών σχεδιασμών που δεν προσδοκούν μελλοντικές αλλαγές στο σύστημα.

Η μέθοδος XP αποτελεί μια ακραία προσέγγιση της επαναλαμβανόμενης ανάπτυξης. Νέες εκδόσεις του συστήματος μπορεί να δημιουργούνται πολλές φορές κάθε μέρα και πρόσθετες λειτουργίες παραδίδονται στον πελάτη κάθε δύο εβδομάδες

Όταν ο προγραμματιστής κατασκευάζει το σύστημα το οποίο προορίζεται για την κυκλοφορία μιας νέας έκδοσης, θα πρέπει να το ελέγξει χρησιμοποιώντας τις υπάρχουσες αυτοματοποιημένες δοκιμές καθώς και τις δοκιμές για τον έλεγχο των νέων λειτουργιών. Η νέα έκδοση του συστήματος είναι αποδεκτή μόνο όταν όλες οι δοκιμές είναι επιτυχείς.

Μια βασική αρχή της παραδοσιακής τεχνολογίας λογισμικού είναι ότι θα πρέπει τα υπό ανάπτυξη συστήματα να σχεδιάζονται και με βάση τις μελλοντικές ανάγκες και συνεπώς θα

πρέπει να επιδέχονται αλλαγές και προσθήκες. Οι αλλαγές αυτές θα πρέπει να πραγματοποιούνται με το μικρότερο δυνατό κόστος σε πόρους, στους οποίους συγκαταλέγονται ο εργάσιμος χρόνος και οι υπολογιστικοί πόροι.

Ωστόσο, στην μέθοδο XP αυτή η αρχή έχει απορριφθεί με τη λογική ότι ο σχεδιασμός για αλλαγές είναι συχνά σπατάλη προσπάθεια. Οι αλλαγές που αναμένονται συχνά δεν πραγματοποιούνται ποτέ και οι αλλαγές που γίνονται είναι εντελώς διαφορετικές από αυτές που αναμένονταν.

Αρχή	Περιγραφή
Αυξανόμενος προγραμματισμός	Οι απαιτήσεις καταγράφονται σε story cards και αποφασίζεται σε ποια έκδοση θα συμπεριληφθούν βάσει του χρόνου που θα είναι διαθέσιμος ο ορισμός τους και της σχετικής τους προτεραιότητας. Οι προγραμματιστές μετατρέπουν αυτά τα σενάρια σε εργασίες.
Μικρές εκδόσεις	Το μικρότερο δυνατό σύνολο λειτουργιών είναι το πρώτο που αναπτύσσεται. Είναι συχνή η έκδοση επόμενων εκδόσεων καθώς και η αυξανόμενη προσθήκη λειτουργιών από την πρώτη έκδοση του συστήματος.
Απλός σχεδιασμός	Ο σχεδιασμός που γίνεται είναι αρκετός ώστε να καλύπτει μόνο τις παρούσες απαιτήσεις.
Προγραμματισμός με δοκιμή πρώτα	Ένα αυτοματοποιημένο πλαίσιο εργασίας για έλεγχο μονάδας χρησιμοποιείται για την συγγραφή δοκιμών για

Αρχή	Περιγραφή
	κάθε νέο κομμάτι λειτουργικότητας, πριν αυτή η λειτουργικότητα υλοποιηθεί.
Αλλαγή κώδικα	Όλοι οι συγγραφείς κώδικα αναμένεται να αλλάζουν συνεχώς μέρη του κώδικα για να τον βελτιώσουν όσο μπορούν. Αυτό-διατηρεί τον κώδικα απλό και εύκολο προς συντήρηση.
Προγραμματισμός ανα ζεύγη	Οι προγραμματιστές δουλεύουν ανά ζεύγη, ελέγχοντας ο ένας την εργασία του άλλου και προσφέροντας βοήθεια για να πετύχουν το καλύτερο δυνατό αποτέλεσμα.
Συλλογική ιδιοκτησία	Τα ζεύγη των προγραμματιστών δουλεύουν σε όλα τα μέρη του συστήματος, ώστε να μην αναπτυχθεί μεμονωμένη γνώση σε ένα κομμάτι του κώδικα και όλοι οι προγραμματιστές να γνωρίζουν καλά όλο τον κώδικα. Οποιοσδήποτε έχει την ικανότητα να αλλάζει οτιδήποτε.
Συνεχής συνένωση	Μόλις τελειώσει μια εργασία ενσωματώνεται σε όλο το σύστημα. Αφού ενσωματωθεί, πρέπει όλες οι δοκιμές μονάδων να είναι επιτυχείς.
Υποφερτός ρυθμός	Πολλές υπερωρίες δεν είναι αποδεκτές καθώς το φαινόμενο του δικτύου συχνά μειώνει την ποιότητα του

Αρχή	Περιγραφή
	κώδικα και την αποδοτικότητα.
Διαθέσιμος πελάτης	Ένας αντιπρόσωπος των τελικών χρηστών θα πρέπει πάντα να είναι διαθέσιμος στην ομάδα του XP. Σε μία διεργασία του XP, ο πελάτης είναι μέλος την ομάδας ανάπτυξης και είναι υπεύθυνος να αποδίδει σε αυτή σωστά τις απαιτήσεις του συστήματος.

Πίνακας 4. Χαρακτηριστικά της τεχνικής XP.

Το πρόβλημα με το σχεδιασμό χωρίς να λαμβάνουμε υπόψη τις αλλαγές σε λειτουργικότητα που μπορεί να ζητηθούν, είναι ότι όταν απαιτούνται αλλαγές, υποβαθμίζουν την δομή του κώδικα και η πραγματοποίηση των αλλαγών με τον καιρό γίνεται δυσκολότερη.

Η μέθοδος XP παρακάμπτει αυτό το πρόβλημα προτείνοντας ότι ο κώδικας πρέπει να αλλάζει συνεχώς. Αυτό σημαίνει ότι η ομάδα προγραμματισμού συνεχώς ψάχνει για πιθανές βελτιώσεις του κώδικα και τις υλοποιεί αμέσως μόλις τις εντοπίσει. Για αυτό το λόγο, ο κώδικας πρέπει να είναι κατανοητός και να μπορούν να γίνουν εύκολα αλλαγές.

Η μεθοδολογία δεν είναι κατάλληλη για:

- Επιχειρηματικά περιβάλλοντα όπου η διοίκηση θέλει να έχει τον πλήρη έλεγχο του έργου
- Έργα με αυστηρές προδιαγραφές
- Έργα που βασίζονται σε σύμβαση με βάση τις προδιαγραφές
- Συστήματα των οποίων τα αποτελέσματα αργούν να εμφανιστούν

- Περιβάλλοντα στα οποία οι έλεγχοι είναι ακριβοί

4.4 Επιλογή της καταλληλότερης μεθοδολογίας

Τα έργα λογισμικού διαφέρουν αρκετά μεταξύ τους, για το λόγο αυτό, διαφέρει και ο τρόπος προσέγγισης της ανάπτυξης λογισμικού, ο οποίος εξαρτάται από πολλούς παράγοντες, όπως, το είδος του συστήματος, η τεχνολογία την οποία θα χρησιμοποιούμε, το μέγεθος της ομάδας που έχουμε, τη φύση των κινδύνων, τις συνέπειες που θα έχουμε στην περίπτωση αποτυχίας, τις εργασιακές συνήθειες της ομάδας, καθώς και την κουλτούρα του οργανισμού.

Όλοι οι παραπάνω τρόποι προσέγγισης, μας δημιουργούν ένα μεγάλο ερώτημα, ποιά μεθοδολογία είναι καλύτερη ή ποια μεθοδολογία ταιριάζει περισσότερο στο project μας;

Ο αριθμός των μεθοδολογιών είναι μεγάλος, ενώ η κάθε μία τους έχει όχι μόνο πλεονεκτήματα, αλλά και μειονεκτήματα, τα οποία μπορούν να μας δημιουργήσουν προβλήματα, σε σημείο που να μην είναι δυνατό να ανταπεξέλθουμε στις ανάγκες του πελάτη και να μην υλοποιηθεί πλήρως το λογισμικό. Για το λόγο αυτό, ανάλογα με τον τύπο Λογισμικού που θέλουμε να αναπτύξουμε θέτουμε κάποια κριτήρια τα οποία μπορούν να βοηθήσουν.

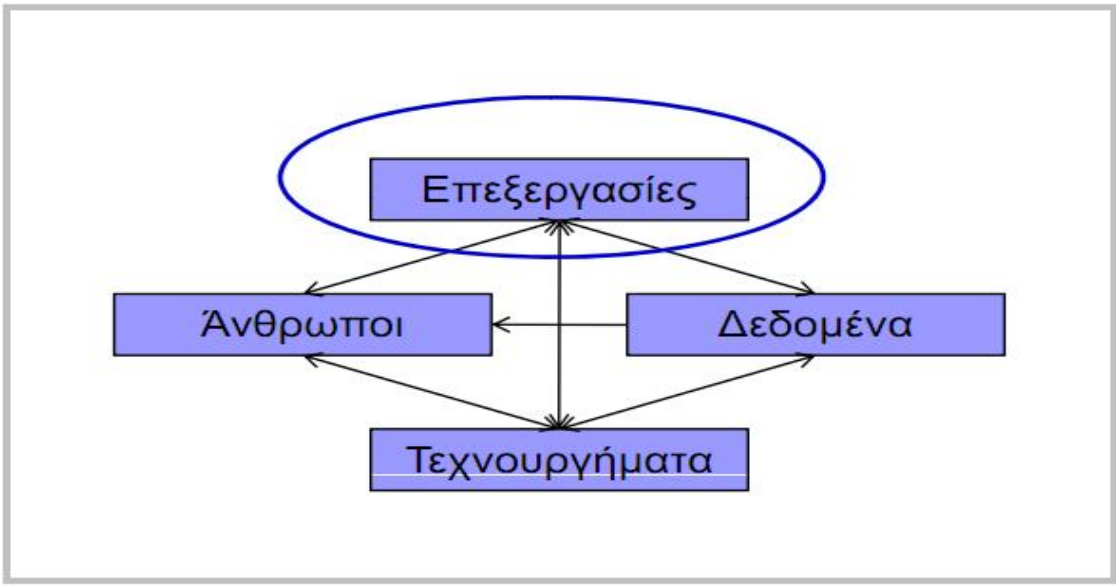
Τα κριτήρια μπορούν να οριστούν αρχικά με τέσσερεις απλές ερωτήσεις για να έχουμε μια πρώτη άποψη πριν κατευθυνθούμε προς την επιλογή της καταλληλότερης μεθοδολογίας.

- Υπάρχει σαφήνεια των απαιτήσεων που έχουν οι χρήστες;
- Οι χρήστες είναι εξοικειωμένοι με την Τεχνολογία;
- Πόσο πολύπλοκο είναι το Σύστημα;
- Τί βαθμό αξιοπιστίας καλείτε το λογισμικό μας να έχει;

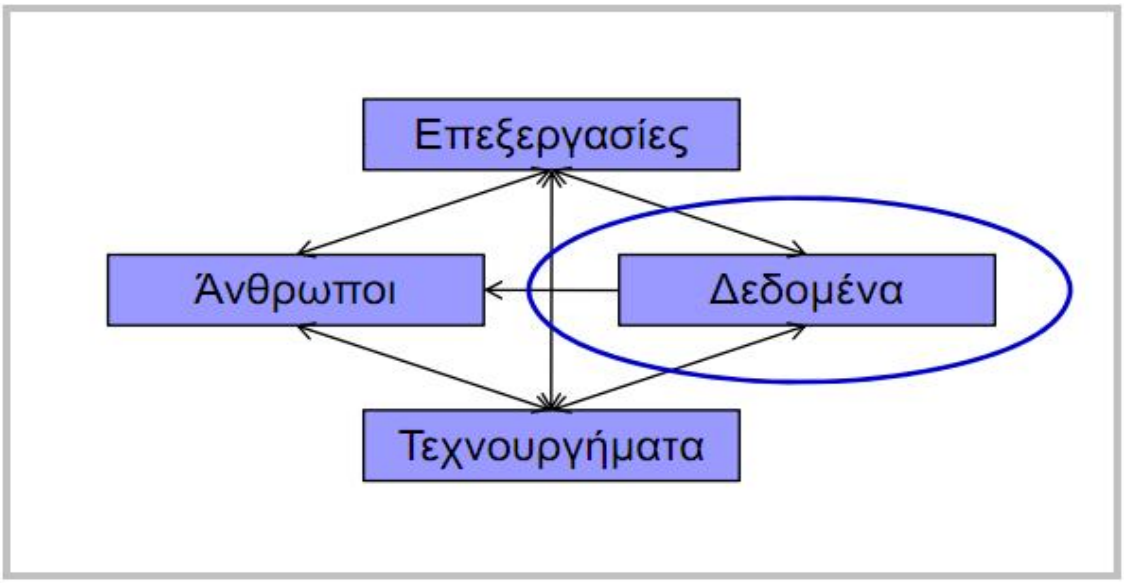
Η μεθοδολογία που επιλέγουμε πρέπει να ταιριάζει στην περίπτωση που την χρειαζόμαστε, ή τουλάχιστον να μπορεί να προσαρμοστεί κατάλληλα στο project που έχουμε τεθεί να φέρουμε

εις πέρας. Πρέπει να εξασφαλίζει, όχι μόνο την ποιότητα των επιμέρους προϊόντων, αλλά και του τελικού συστήματος, ενώ ταυτόχρονα πρέπει να ανακαλύπτει έγκαιρα τις πραγματικές απαιτήσεις των χρηστών πριν δαπανηθεί αρκετός χρόνος, αλλά και σημαντικό χρηματικό ποσό για τις εργατικές ώρες. Πρέπει να μας διαβεβαιώνει ότι το παραδοτέο σύστημα βοηθάει πράγματι το χρήστη και να επιτρέπει να πραγματοποιηθούν αλλαγές στο σύστημα και μετά την υλοποίησή του διότι, σκοπός της μεθοδολογίας είναι να ελαχιστοποιεί τη γραφειοκρατία, να μην εμποδίζει τη δημιουργικότητα, ώστε το management της ανάπτυξης και της λειτουργίας να γίνεται χωρίς εμπόδια, έχοντας πάντα βοήθεια από αυτοματοποιημένα εργαλεία.

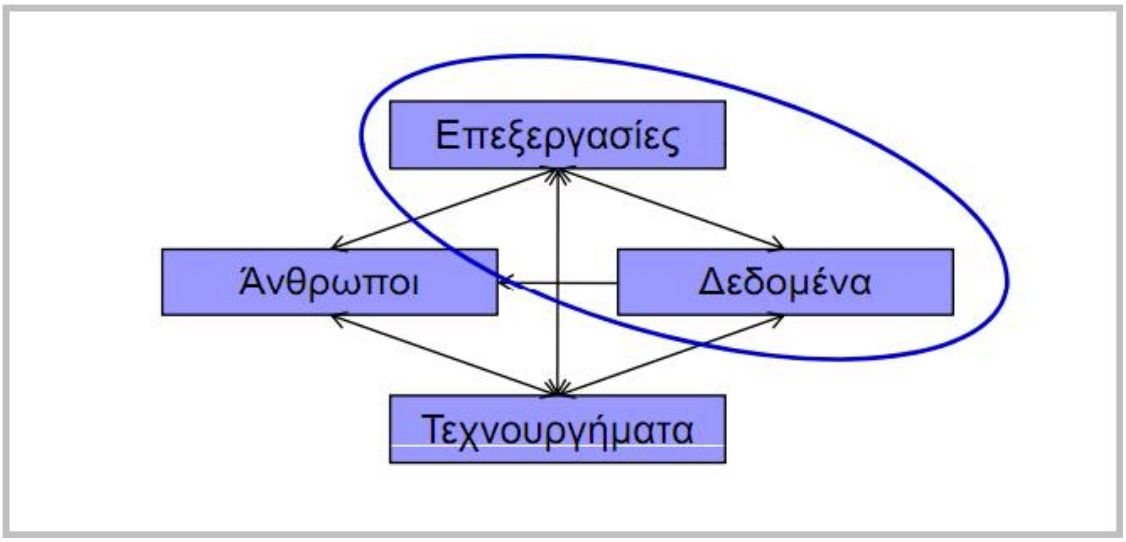
Με σκοπό να καταφέρουμε να κάνουμε την καλύτερη επιλογή της καταλληλότερης μεθοδολογίας, κατηγοριοποιούμε τις μεθοδολογίες με βάση τα σημεία που δίνεται έμφαση. Κατά την ανάλυση των μεθοδολογιών, βλέπουμε πως έχουμε τρείς διαφορετικές κατηγορίες, επεξεργασιο-κεντρικές (process centered), δεδομενο-κεντρικές (data-centered) και τις αντικειμενο-στρεφείς (object-oriented).



Εικόνα 13. Process-centered methodologies



Εικόνα 14. Data-centered methodologies



Εικόνα 15. Object-oriented methodologies

4.5 Αδυναμίες δομημένης ανάλυσης, σχεδίασης και ανάπτυξης

Παρατηρώντας τη μεθοδολογία της δομημένης ανάλυσης, σχεδίασης και ανάπτυξης, γίνεται αντιληπτή η εγγενής αδυναμία στην απεικόνιση των οντοτήτων του πραγματικού κόσμου σε συστατικά λογισμικού ανεξάρτητα από τα δεδομένα. Η πειθαρχία και η αυστηρή υλοποίηση

πρωτοτύπων είναι μία πολύ δύσκολη διεργασία, ενώ δεν μπορούμε να αγνοήσουμε ότι η επιχειρησιακή λογική δεν είναι εύκολο να μοντελοποιηθεί και να αντιστοιχιστεί σε δεδομένα και λειτουργίες , οι οποίες είναι εντελώς ανεξάρτητες μεταξύ τους. Επίσης είναι πολύ δύσκολη η επαναχρησιμοποίηση συστατικών του λογισμικού σε νέες εφαρμογές

Η δομημένη προσέγγιση μας κατευθύνει στην αντικειμενοστρεφή μοντελοποίηση, διότι η ανάγκη για γρήγορη πρόοδο είναι πολύ μεγάλη. Οι δομημένες μεθοδολογίες ανάλυσης και σχεδιασμού στηρίζονται σε διαδοχικούς μετασχηματισμούς που αλλάζουν δραματικά τη δομή και τα στοιχεία κάθε φάσης, καθώς προχωράμε από την ανάλυση στον προγραμματισμό μέσω του σχεδιασμού.

Τα βασικότερα μειονεκτήματα είναι η αλλαγή από τη μια φάση στην άλλη και ο έλεγχος της ορθότητας κάθε μετασχηματισμού είναι πολύ δύσκολα, ακόμα και αν οι ομάδες ανάπτυξης καταφέρουν να κάνουν τους μετασχηματισμούς με επιτυχία, αυτός ο σωστός μετασχηματισμός σπάνια διαρκεί. Όταν παρουσιασθούν νέες απαιτήσεις, συχνά, κωδικοποιούνται αμέσως με αλλαγές στον φυσικό κώδικα, οπότε, ο φυσικός σχεδιασμός δεν αντιστοιχεί στο υπάρχον πρόγραμμα λογισμικού. Έτσι, όπως ο δομημένος προγραμματισμός οδήγησε στη δομημένη μεθοδολογία σχεδίασης, έτσι και ο αντικειμενοστρεφής προγραμματισμός οδήγησε στην αντικειμενοστρεφή μεθοδολογία.

4.6 Αντικειμενοστρεφείς Μεθοδολογίες

Σύμφωνα με την αντικειμενοστρεφή θεώρηση, μπορούμε να θεωρήσουμε τον πραγματικό κόσμο ως ένα σύνολο από αντικείμενα τα οποία επικοινωνούν μεταξύ τους. Στόχος μας είναι η μοντελοποίηση αντικειμένων από τον πραγματικό κόσμο και της επικοινωνίας τους και ο μετασχηματισμός των μοντέλων αυτών σε αντικειμενοστρεφές λογισμικό.

Για να πετύχουμε τον στόχο μας χρειάζεται να χρησιμοποιήσουμε Αντικειμενοστρεφή Γλώσσα Μοντελοποίησης (πχ.UML), με αντικειμενοστρεφή μοντέλα, Αντικειμενοστρεφή Μεθοδολογία Ανάπτυξης (πχ.ICONIX) και αντικειμενοστρεφή εργαλεία ανάπτυξης, δηλαδή case tools (πχ. Star UM) και γλώσσες προγραμματισμού (πχ.Java).

Η αντικειμενοστρεφής προσέγγιση επιταχύνει τη διαδικασία ανάπτυξης, είναι πολύ ευέλικτη διευκολύνοντας την αλλαγή των απαιτήσεων, τη σταδιακή βελτίωση και επέκταση του λογισμικού, τον έλεγχο και την συντήρησή του. Δίνει τη δυνατότητα της επιστροφής σε προηγούμενες φάσεις, παραμερίζοντας το εμπόδιο του ριζικού μετασχηματισμού, ενώ η μετάβαση από φάση σε φάση είναι ευκολότερη διότι σε κάθε φάση, χρησιμοποιείται η ίδια γλώσσα.

Οι βασικότερες έννοιες της αντικειμενοστρεφούς προσέγγισης είναι οι εξής:

- Αντικείμενο (object): Λογική ενότητα η οποία συνδυάζει δεδομένα και επεξεργασίες για να εκπληρώσει το ρόλο της μέσα στο σύστημα.

Σύμφωνα με τον Booch (1991) για να οριστεί ένα αντικείμενο θα πρέπει να έχει:

- Κατάσταση (state), η οποία καθορίζεται από τις ιδιοκτησίες του και τις τιμές σε κάθε μια από αυτές
- Συμπεριφορά (behavior), η οποία αναφέρεται στο πως δρα το αντικείμενο μέσα στο περιβάλλον του, δηλαδή πως αντιδρά σε κάθε ενεργοποίηση των μεθόδων του
- Ταυτότητα, δηλαδή το μοναδικό χαρακτηριστικό το οποίο αντιστοιχεί σε μια οντότητα του πραγματικού κόσμου
- Μέθοδοι (methods): Είναι οι πράξεις που εκτελεί το αντικείμενο όταν λαμβάνει ένα μήνυμα. Οι πράξεις μπορούν να εκτελούνται μόνο αν είναι γνωστές στο αντικείμενο.
- Μηνύματα (messages): Είναι οι αιτήσεις που στέλνονται από ένα αντικείμενο σε ένα άλλο έτσι ώστε το τελευταίο να εκτελέσει μια λειτουργία

Οι αντικειμενοστρεφείς μεθοδολογίες εκμεταλλεύονται τις αρχές και τις τεχνικές του αντικειμενοστρεφούς προγραμματισμού.

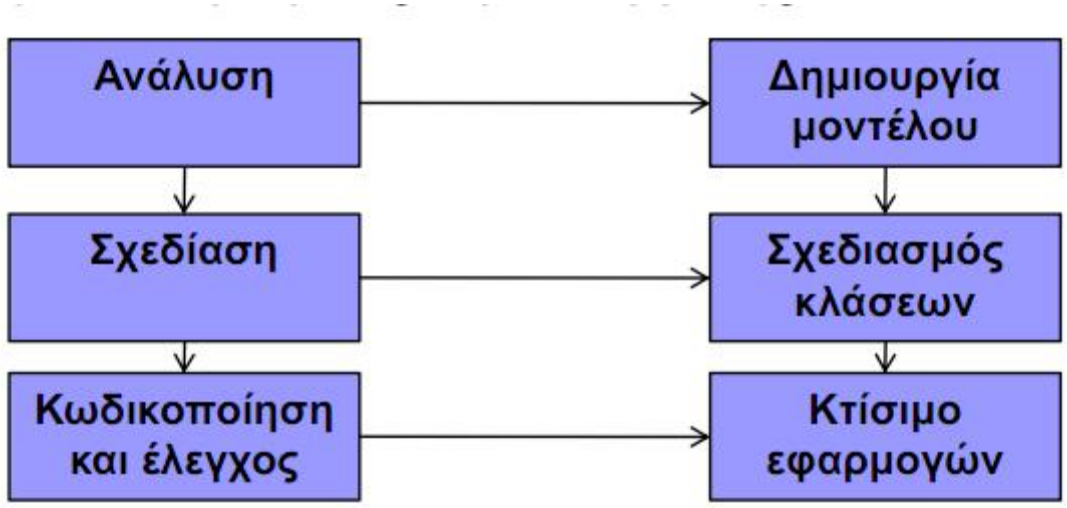
Ενθυλάκωση (encaspulation): Η συγκέντρωση, ομαδοποίηση δεδομένων και πράξεων στην ίδια δομή. Με αυτόν τον τρόπο «κρύβονται» οι εσωτερικές λεπτομέρειες της υλοποίησης και

αυτό αποτελεί σημαντικό πλεονέκτημα και για την επαναχρησιμοποίηση. Οι εσωτερικές λεπτομέρειες περιλαμβάνουν δεδομένα για την κατάσταση των ενοτήτων και τις διαδικασίες που καθορίζουν τη συμπεριφορά τους.

Τμηματοποίηση (modularity): Ο χωρισμός πολύπλοκων συστημάτων σε ενότητες (modules).

Αφαίρεση (abstraction): Η απλοποιημένη περιγραφή και καθορισμός του συστήματος ώστε να δίδεται έμφαση σε ορισμένες λεπτομέρειες και να αγνοούνται άλλες.

Πολυμορφισμός (polymorphism): Η ιδιότητα που επιτρέπει να σταλεί το ίδιο μήνυμα σε διαφορετικά αντικείμενα και το κάθε αντικείμενο να εκτελέσει μια λειτουργία που αντιστοιχεί στην κλάση στην οποία ανήκει.



Εικόνα 16. Αντιστοίχιση φάσεων μοντέλου κύκλου ζωής και αντικειμενοστρεφούς προσέγγισης

Χαρακτηριστικά της αντικειμενοστρεφούς προσέγγισης είναι η διαδικασία της ανάπτυξης, που γίνεται με βάση τις περιπτώσεις χρήσεων και την αρχιτεκτονική ενώ είναι επαναληπτική και αυξητική.

Αναλυτικότερα, η διαδικασία ανάπτυξης ξεκινάει με περιπτώσεις χρήσης (use cases), δηλαδή, σενάρια από τα οποία μπορούμε να συνάγουμε τις λειτουργικές και τις μη λειτουργικές απαιτήσεις, όπως επίσης περιγράφουν επιθυμητές και ανεπιθύμητες ακολουθίες γεγονότων.

Οι περιπτώσεις χρήσης χρησιμοποιούνται ως βασικά στοιχεία, για τον καθορισμό της συμπεριφοράς του συστήματος, την επαλήθευση της αρχιτεκτονικής του συστήματος και τη δοκιμή του, όπως επίσης και για την επικοινωνία μεταξύ αυτών που εμπλέκονται στην ανάπτυξη του συστήματος.

Η διαδικασία ανάπτυξης με βάση την αρχιτεκτονική χρησιμοποιεί ως βασικό στοιχείο την κατανόηση, την κατασκευή, τη διαχείριση και την εξέλιξη του υπό κατασκευή συστήματος, ενώ πρέπει να οδηγεί στις προδιαγραφές (specifications), την κατασκευή και την τεκμηρίωση του συστήματος.

Η επαναληπτική και αυξητική διαδικασία ανάπτυξης αφορά τη διαχείριση και τη συνεχή ολοκλήρωση της αρχιτεκτονικής για την παραγωγή των εκτελέσιμων εκδόσεων ενός συστήματος. Το λογισμικό δεν πρέπει να εκδίδεται μονομιάς στο τέλος του έργου, αλλά πραγματοποιούνται αρκετές επαναλήψεις, και κάθε επανάληψη φέρνει το σύστημα όλο και πιο κοντά στις πραγματικές ανάγκες του χρήστη, ενώ κάθε έκδοση αποτελεί βελτιωμένη έκδοση της προηγούμενης.

4.6.1 Unified Process (UP)

Η Ενοποιημένη Διαδικασία (UP) επιτρέπει σε νέες εκδόσεις του έργου να υλοποιούν και νέες απαιτήσεις που εξελίσσονται όσο το σύστημα υλοποιείται. Καθοδηγείται από μελέτες χρήσης (use cases) και χρησιμοποιεί την UML σαν γλώσσα μοντελοποίησης. Το μοντέλο προσφέρει πλούσιο πλαίσιο υποστήριξης της διαδικασίας. Το μοντέλο UP έχει τέσσερις φάσεις όπου κάθε φάση έχει προκαθορισμένο σκοπό και ολοκληρώνεται τη χρονική στιγμή, στην οποία πρέπει να παρθεί κρίσιμη απόφαση και συνεπώς, σημαντικοί στόχοι πρέπει να έχουν επιτευχθεί (Sommerville, 2001).

- Έναρξη ή Σύλληψη (Inception) – Ορισμός του έργου και της έκτασής του. Παρουσιάζεται η αρχική ιδέα του συστήματος μέχρι του σημείου που είναι πολύ καλά θεμελιωμένη για να επιτρέψει την είσοδο στη φάση επεξεργασίας.

- Επεξεργασία (Elaboration) – Κατάστρωση μεθόδου υλοποίησης του έργου, μοντελοποίηση χαρακτηριστικών του έργου, ορισμός της αρχιτεκτονικής του συστήματος.

- Κατασκευή (Construction) – Υλοποίηση του έργου .

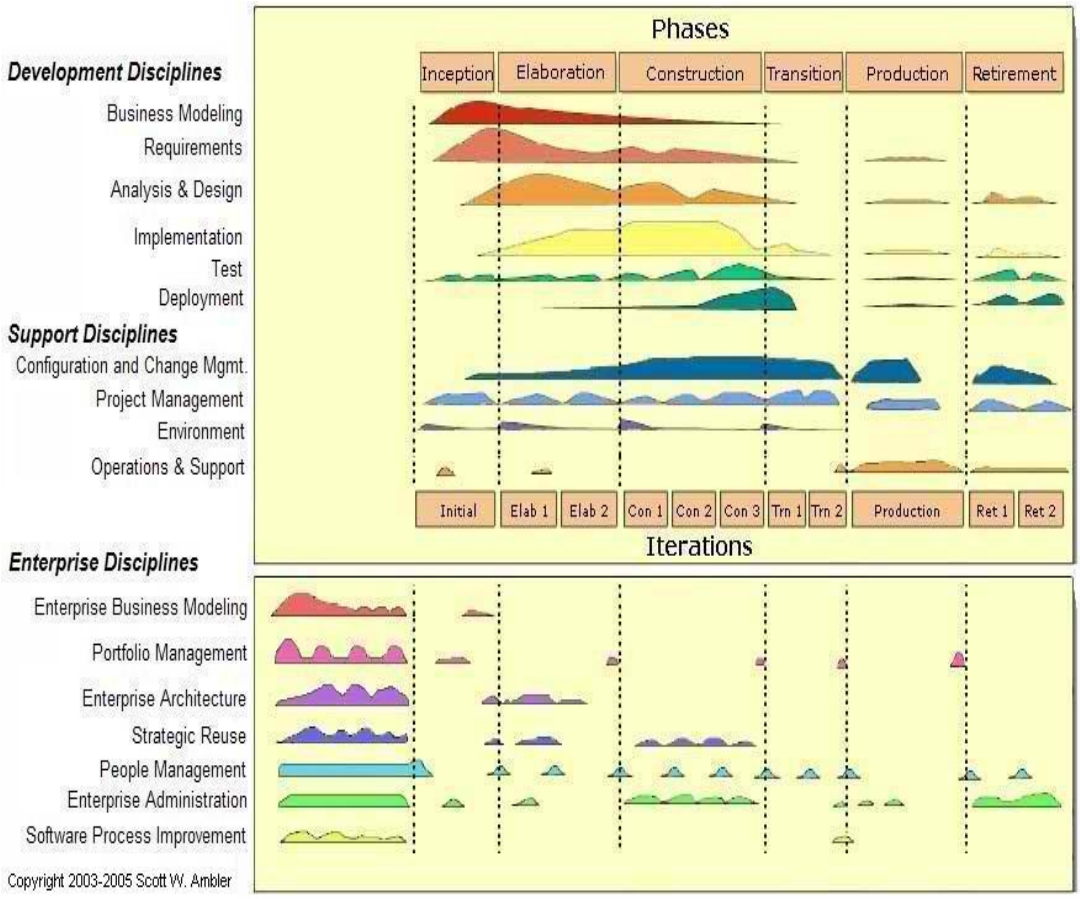
- Μετάβαση (Transition) – Ανάπτυξη του συστήματος στο περιβάλλον χρήσης του.

Γίνεται η εγκατάσταση συστήματος στο περιβάλλον χρήσης του, πραγματοποιείται η εκπαίδευση χρηστών, ενώ ελέγχεται το σύστημα από τους χρήστες.

Το σημείο της μετάβασης αποτελεί έναρξη της φάσης συντήρησης του λογισμικού και όχι το τέλος της διαδικασίας ανάπτυξης.

- Παραγωγή (Production) – Η χρήση του συστήματος σε πραγματικές συνθήκες.

- Απόσυρση (Retirement) – Η φάση όπου το σύστημα δεν επιδέχεται άλλες βελτιώσεις και δεν ικανοποιεί τις ανάγκες των χρηστών .



Εικόνα 17. Στάδια της διαδικασίας UP.

4.6.2 ICONIX

Είναι κοινά αποδεκτό, ότι το αντικειμενοστρεφές υπόδειγμα προγραμματισμού παρέχει πολλά πλεονεκτήματα, σε σχέση με την ευκολία κατανόησης, συντήρησης, ελέγχου κι επαναχρησιμοποίησης του λογισμικού, έναντι των άλλων υποδειγμάτων, (π.χ. διαδικαστικού). Όμως τα πλεονεκτήματα αυτά είναι εμφανή σε έργα λογισμικού, τα οποία εστιάζουν το πρόβλημα στην δυσκολία επικοινωνίας και συντονισμού των μονάδων του έργου, δηλαδή σε έργα μεγάλης κλίμακας.

Μονάδες του έργου θεωρούνται οι μονάδες του λογισμικού (τα φυσικά ή λογικά συστατικά του) που αναπτύσσονται χωριστά και πρέπει να συνεργαστούν για την επίτευξη της ζητούμενης λειτουργικότητας, αλλά και οι εμπλεκόμενοι στη διαδικασία ανάπτυξης, δηλαδή οι πελάτες, οι

τελικοί χρήστες, οι αναλυτές, οι σχεδιαστές, οι προγραμματιστές και τα μέλη της διοίκησης της εταιρείας ανάπτυξης. Τα άτομα αυτά, επειδή έχουν διαφορετική άποψη για τη διαχείριση του έργου, πρέπει να συντονιστούν και να επικοινωνήσουν αποτελεσματικά, ώστε η ανάπτυξη ενός έργου λογισμικού να είναι επιτυχής.

Επιπρόσθετα, τα έργα μεγάλης κλίμακας χαρακτηρίζονται από την εξελισσόμενη φύση τους. Είναι κοινά αποδεκτό ότι οι απαιτήσεις των πελατών συνεχώς αλλάζουν, για αυτό και προκύπτει η αναγκαιότητα για συντήρηση του λογισμικού με την πάροδο του χρόνου.

Με τον όρο συντήρηση εννοούμε τον εμπλουτισμό του λογισμικού με νέες δυνατότητες που ανταποκρίνονται στα ζητούμενα των πελατών (**τροποποιητική συντήρηση**), όσο και στη διόρθωση σφαλμάτων ή ατελειών του λογισμικού, καθώς αυτά εντοπίζονται από την εκτεταμένη χρήση του λογισμικού (**διορθωτική συντήρηση**).

Το αντικειμενοστρεφές υπόδειγμα και η χρήση της Ενοποιημένης Γλώσσας Μοντελοποίησης στα πλαίσια κάποιας μεθοδολογίας ανάπτυξης, όπως η ICONIX, παρέχουν τα εργαλεία και τις τεχνικές για την αποτελεσματική διαχείριση των προβλημάτων αυτών σε έργα λογισμικού μεγάλης κλίμακας.

Αξίζει να σημειωθεί ότι η χρήση αντικειμενοστρεφούς προσέγγισης σε έργα πολύ μικρής κλίμακας ενδέχεται να μην προσφέρει πλεονεκτήματα, αλλά αντίθετα να συνεισφέρει στην πολυπλοκότητα του έργου.

4.6.2.1 Ορισμός και φιλοσοφία ICONIX

Η ICONIX είναι μια μεθοδολογία ανάπτυξης λογισμικού που επιτρέπει τη συστηματική μετάβαση από τις αρχικές απαιτήσεις όπως αυτές διατυπώνονται από τον πελάτη, στον κώδικα που τελικά υλοποιεί τις απαιτήσεις αυτές. Είναι μια απλούστερη εκδοχή της ευρέως διαδεδομένης Ενοποιημένης Προσέγγισης (Unified Process).

Το μείζον χαρακτηριστικό της μεθοδολογίας ICONIX είναι η επαναληπτικότητα. Αφενός, η διαδικασία είναι επαναληπτική διότι επιτρέπει την παραγωγή λειτουργικού κώδικα για κάθε μια περίπτωση χρήσης του συστήματος ξεχωριστά. Σε κάθε επανάληψη, εξετάζεται μια νέα περίπτωση χρήσης που καταλήγει στην προσθήκη λειτουργικότητας στο τελικό προϊόν. Αφετέρου, η διαδικασία είναι επαναληπτική διότι επιτρέπει (και υποβοηθά) την επιστροφή από ένα στάδιο της διαδικασίας ανάπτυξης (π.χ. το σχεδιασμό) σε προηγούμενα (π.χ. την ανάλυση απαιτήσεων), με σκοπό την αποσαφήνιση, βελτίωση και διόρθωση προηγούμενων ενεργειών. Η επαναστατικότητα βρίσκεται στο κέντρο του αντικειμενοστραφούς υποδείγματος προγραμματισμού, καθώς αναγνωρίζει ότι ένα μεγάλο σύστημα λογισμικού δεν μπορεί να αναπτυχθεί με μιας και ότι οι αλλαγές σε προηγούμενες επιλογές είναι αναπόφευκτες.

Η φιλοσοφία της μεθοδολογίας ICONIX αποτυπώνεται στο διάγραμμα της παρακάτω εικόνας. Το ζητούμενο είναι από τις απαιτήσεις του χρήστη να παραχθεί το τελικό προϊόν, δηλαδή ο λειτουργικός κώδικας. Ο κώδικας παράγεται με τη διαδοχική εκλέπτυνση και βελτίωση δύο μοντέλων:

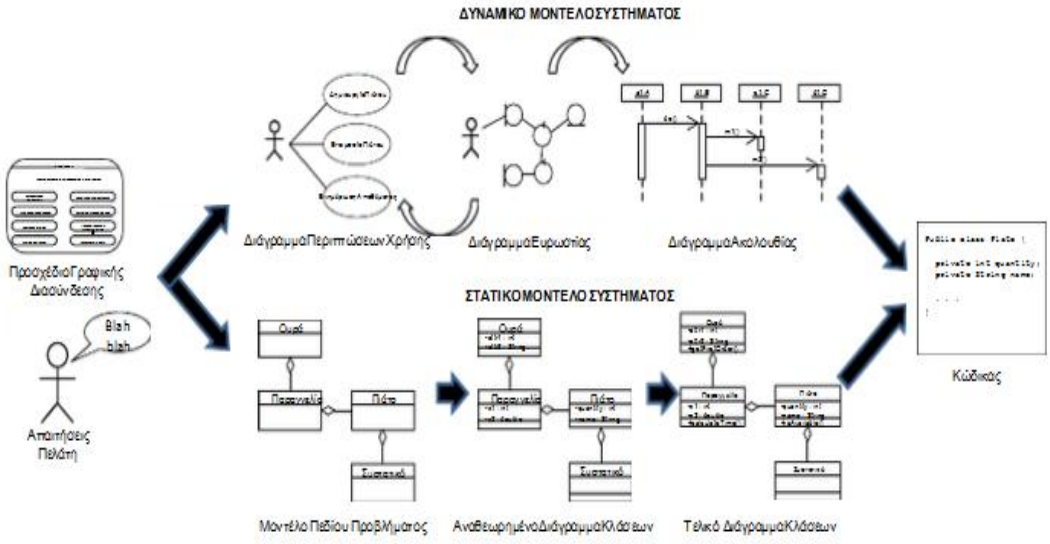
- Του στατικού μοντέλου που περιγράφει την αρχιτεκτονική του συστήματος, δηλαδή τις δομικές του μονάδες και τις σχέσεις μεταξύ τους και
- Του δυναμικού μοντέλου που περιγράφει τον τρόπο με τον οποίο οι μονάδες αλληλεπιδρούν για την επίτευξη της επιθυμητής λειτουργικότητας.

Οι απαιτήσεις του πελάτη/χρήστη του συστήματος θεωρείται ότι διατυπώνονται σε κάποιο αρχικό κείμενο (γνωστό ως απαιτήσεις υψηλού επιπέδου) και ενδεχομένως σε κάποια σκίτσα της επιθυμητής γραφικής διασύνδεσης χρήστη.

Η μεθοδολογία ICONIX βασίζεται στην αξιοποίηση ενός υποσυνόλου της UML για τη δημιουργία διαγραμμάτων ως ενδιάμεσα προϊόντα κατά την εξέλιξη του δυναμικού και στατικού μοντέλου του συστήματος που αναπτύσσεται.

Συγκεκριμένα, από το σύνολο των διαγραμμάτων της UML, χρησιμοποιούνται τα διαγράμματα περιπτώσεων χρήσης, τα διαγράμματα κλάσεων, τα διαγράμματα ευρωστίας (ειδική περίπτωση των διαγραμμάτων συνεργασίας) και τα διαγράμματα ακολουθίας.

Το βασικό χαρακτηριστικό της ICONIX είναι η επαναληπτική μέθοδος ανάπτυξης λογισμικού, κατά την οποία ¨σπάμε¨ το έργο με βάση τη λειτουργικότητά του. Γι' αυτόν ακριβώς το λόγο επιλέχθηκε από τους αναλυτές- προγραμματιστές ως το καταλληλότερο εργαλείο για την ανάπτυξη του πληροφοριακού συστήματος.



Εικόνα 18. Μεθοδολογία ICONIX

Από τη στιγμή που οι απαιτήσεις του πελάτη δοθούν στην εταιρεία ανάπτυξης λογισμικού (και η εταιρεία αναλάβει το έργο) η διαδικασία ακολουθεί τις εξής φάσεις:

4.6.2.2 Ανάλυση Απαιτήσεων

Πρώτο βήμα στην ανάλυση των απαιτήσεων είναι η καταγραφή των οντοτήτων του πεδίου που διαπραγματεύεται το σύστημα που πρόκειται να αναπτυχθεί (πεδίο προβλήματος) και των σχέσεων μεταξύ τους. Οι οντότητες αυτές αποτελούν τη βάση του στατικού αντικειμενοστρεφούς μοντέλου καθώς η λειτουργία του λογισμικού βασίζεται στην αλληλεπίδραση μεταξύ τους. Έχοντας το μοντέλο του πεδίου προβλήματος στη διάθεσή μας, επόμενο βήμα στην ανάλυση απαιτήσεων είναι η λεπτομερής και σαφής περιγραφή των απαιτήσεων του πελάτη υπό μορφή περιπτώσεων χρήσης

4.6.2.3 Ανάλυση– Αρχική Σχεδίαση

Σε επίπεδο δυναμικού μοντέλου κύριο εργαλείο στη φάση αυτή είναι τα διαγράμματα ευρωστίας για τη διερεύνηση των ενεργειών που υπαγορεύονται από τις περιπτώσεις χρήσης και κυρίως την βελτίωση του κειμένου των ίδιων των περιπτώσεων χρήσης. Με το πέρας της ανάλυσης ευρωστίας προκύπτει συνήθως ένα αναθεωρημένο και εμπλουτισμένο διάγραμμα κλάσεων (ως μετεξέλιξη του μοντέλου πεδίου προβλήματος) που περιλαμβάνει επιπρόσθετες κλάσεις καθώς και ορισμένες από τις ιδιότητες των κλάσεων.

4.6.2.4 Σχεδίαση

Στη φάση της σχεδίασης επιχειρείται η ακριβής διερεύνηση της δυναμικής συμπεριφοράς του συστήματος και η κατανομή της λειτουργικότητας στις κλάσεις. Κύριο εργαλείο για το σκοπό αυτό είναι τα διαγράμματα ακολουθίας.

Με το πέρας της σχεδίασης η ομάδα ανάπτυξης παράγει το τελικό διάγραμμα κλάσεων, που αποτελεί την είσοδο για τη φάση της υλοποίησης του λογισμικού.

4.6.2.5 Υλοποίηση

Το σημαντικότερο προϊόν της διαδικασίας ανάπτυξης δεν είναι φυσικά τα διαγράμματα, αλλά ο ίδιος ο κώδικας που ικανοποιεί τις απαιτήσεις του πελάτη. Στη φάση της υλοποίησης αναπτύσσεται ο κώδικας βάσει της σχεδίασης που προηγήθηκε σε μεγάλο βαθμό υπάρχει η δυνατότητα παραγωγής κώδικα από τα διαγράμματα κλάσεων και ακολουθίας). Σημαντικό μέρος της υλοποίησης είναι και ο έλεγχος μονάδων (unit testing), δηλαδή η εξασφάλιση της ορθής λειτουργίας των κλάσεων που δημιουργήθηκαν.

Σύμφωνα με τη μεθοδολογία ICONIX αλλά και τις γενικότερες επιταγές της Τεχνολογίας Λογισμικού, στο τέλος κάθε φάσης θα πρέπει να πραγματοποιείται μια συστηματική επισκόπηση (inspection/review).

Στις επισκοπήσεις αυτές τα μέλη της ομάδας ανάπτυξης (αλλά ενδεχομένως και εκπρόσωποι του πελάτη ή της διοίκησης) ελέγχουν την ορθότητα των μοντέλων που δημιουργήθηκαν και τη συνέπεια προς τις απαιτήσεις. Στη συνέχεια παρουσιάζονται οι φάσεις ανάπτυξης για ένα έργο λογισμικού που αφορά σε ένα σύστημα διαχείρισης παραγγελιών εστιατορίου .Όπως αναφέρθηκε, η διαδικασία ξεκινά από την αρχική διατύπωση των απαιτήσεων από πλευράς του πελάτη.

Σημειώνεται ότι οι προδιαγραφές που τίθενται από το χρήστη (αναφέρονται ως απαιτήσεις υψηλού επιπέδου – high level requirements specification) είναι συχνά περιληπτικές, ασαφείς και όχι πλήρεις.

Κεφάλαιο 5 Μοντελοποίηση-Γλώσσες μοντελοποίησης λογισμικού

5.1 Μοντέλο πληροφοριακού συστήματος

Ένα μοντέλο πληροφοριακού συστήματος είναι η απεικόνιση ενός πληροφοριακού συστήματος, με τέτοιο βαθμό πληροφορίας που απαιτείται ανάλογα με το χρησιμοποιούμενο διάγραμμα.

Στην Τεχνολογία Λογισμικού, μοντέλο είναι μία προσεγγιστική λύση ενός προβλήματος, το οποίο σχετίζεται με μια συγκεκριμένη οντότητα. Αυτή η προσεγγιστική λύση προκύπτει αφαιρώντας από την πλήρη λύση αυτού του προβλήματος ορισμένα στοιχειά θεωρώντας τα επουσιώδη και κρατώντας ορισμένα άλλα θεωρώντας τα ουσιώδη.

Όταν το πρόβλημα είναι σύνθετο, για να προσεγγιστεί επαρκώς η λύση του, πρέπει να ληφθούν υπόψη τα κύρια υπό-προβλήματά του. Για κάθε κύριο υπό-πρόβλημα αυτού του σύνθετου προβλήματος απαιτείται και ένα ξεχωριστό (υπό) μοντέλο που ονομάζεται όψη, και να προσεγγίζει τη λύση του υπό-προβλήματος. Έτσι η όψη είναι (υπό)μοντέλο, ένα από πολλά που απαιτούνται για να προσεγγιστεί η λύση ενός σύνθετου προβλήματος μιας οντότητας. Το κύριο υπό-πρόβλημα ονομάζεται σκοπιά.

Το μοντέλο λοιπόν, ως προσεγγιστική λύση ενός προβλήματος μιας οντότητας αποτελείται από όψεις που κάθε μία αναφέρεται σε μια τριάδα που είναι (οντότητα, πρόβλημα, υπό-πρόβλημα).

Για μια οντότητα μπορούμε να έχουμε πολλά διαφορετικά προβλήματα και για το κάθε ένα από αυτά να έχουμε μπορούμε να έχουμε πολλές όψεις. Μπορούμε βέβαια για μια οντότητα να έχουμε μόνο ένα πρόβλημα και για αυτό μια μόνο σκοπιά, οπότε έχουμε ένα μόνο μοντέλο με μία μόνο όψη.

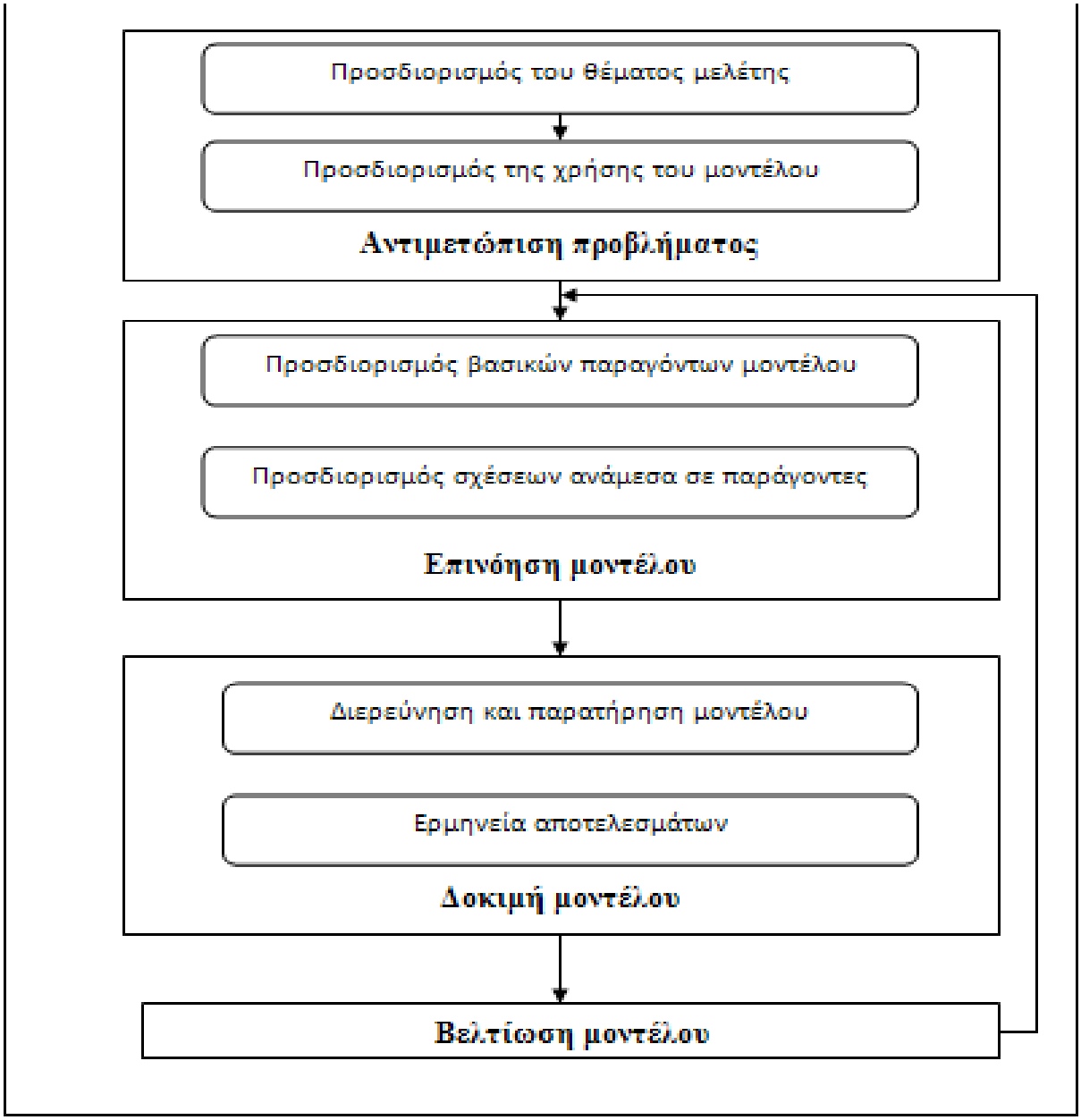
Το μοντέλο μιας οντότητας αναφέρεται σε ένα πρόβλημά της και αποτελείται από όψεις, που κάθε μία αποτελείται από διαγράμματα και το κάθε διάγραμμα αποτελείται από κόμβους και ακμές που σχετίζονται με την προσεγγιστική λύση την οποία παριστάνουν, τα σημαντικότερα χαρακτηριστικά που πρέπει να έχει ένα καλώς ορισμένο μοντέλο πληροφοριακού συστήματος είναι τα εξής:

- Ακρίβεια - περιγράφουν με σωστό τρόπο το σύστημα.
- Συνέπεια – διαφορετικές όψεις δεν έρχονται σε σύγκρουση μεταξύ τους.
- Διευκόλυνση της επικοινωνίας.
- Ευμετάβλητα.
- Κατανοητά.

Λαμβάνοντας υπόψην τα παραπάνω χαρακτηριστικά, θα έχουμε επιτύχει ένα καλά ορισμένο μοντέλο πληροφοριακού συστήματος, πριν από τη σχεδίαση.

5.2 Μοντελοποίηση

Μοντελοποίηση είναι η διαδικασία της επινόησης και της δοκιμής ενός μοντέλου. Η διάθεση του ανθρώπου να κατασκευάζει μοντέλα είναι πολύ παλιά, αρχίζει από τη στιγμή που οι πρωτόγονοι άνθρωποι δημιούργησαν μαγικά ομοιώματα για να εξευμενίζουν τους θεούς τους και φτάνει μέχρι στις μέρες μας, τόσο με το παιχνίδι των παιδιών στην πρώτη κιόλας παιδική ηλικία, όσο και με τα μοντέλα που επινοούνται στα επιστημονικά εργαστήρια.



Εικόνα 19. Διαδικασία μοντελοποίησης Πληροφοριακού Συστήματος.

Η διαδικασία μοντελοποίησης, ανεξάρτητα από το είδος των μοντέλων τα οποία επινοεί, είναι σχεδόν συγκεκριμένη και περιλαμβάνει στη πλήρη της μορφή ορισμένα στάδια, που είναι λιγότερο ή περισσότερο διακριτά:

1. Αντιμετώπιση προβλήματος, όπου πρέπει να γίνει προσδιορισμός του θέματος, της κατάστασης ή του προβλήματος που θέλουμε να μοντελοποιήσουμε, καθώς και του σκοπού για τον οποίο επινοούμε το μοντέλο (για να εξηγήσουμε κάτι, να προβλέψουμε, ή να αναπαραστήσουμε).

2. Επινόηση του μοντέλου για την οποία απαιτείται:

- Αρχικά, η μελέτη και η απόφαση για το ποιοι είναι οι βασικοί παράγοντες του μοντέλου και πως θα απλοποιήσουμε την πραγματικότητα, και

- Στη συνέχεια, ο καθορισμός των σχέσεων ανάμεσα στις συνιστώσες του μοντέλου.

3. Δοκιμή της καταλληλότητας του μοντέλου (η οποία γίνεται είτε μελετώντας τη συμπεριφορά του μέσω προσομοίωσης είτε εφαρμόζοντας άλλες μεθόδους) και ερμηνεία των αποτελεσμάτων.

4. Βελτίωση του μοντέλου, που είναι σχεδόν πάντα απαραίτητη, και συνήθως γίνεται με επανάληψη των δύο προηγούμενων βασικών σταδίων.

Παρουσιάζονται στη συνέχεια, τα βασικά χαρακτηριστικά κάθε ενός από τα στάδια της μοντελοποίησης, καθώς και ορισμένα ερωτήματα που είναι χρήσιμο να τα θέτουμε κατά τη διάρκεια της διαδικασίας αυτής.

5.2.1 Αντιμετώπιση προβλήματος - Προσδιορισμός θέματος μελέτης

Η διαδικασία μοντελοποίησης που αναφέρουμε παρακάτω πραγματοποιείται για τη μελέτη ενός θέματος, μιας κατάστασης ή ειδικότερα ενός πιο συγκεκριμένου προβλήματος.

Αρχικά, θα πρέπει λοιπόν να προσδιορίσουμε με ακρίβεια, ποιο είναι το θέμα της μελέτης ή το συγκεκριμένο πρόβλημα, το οποίο θα κληθούμε να επιλύσουμε. Ύστερα, θα χρειαστεί να αναφέρουμε την κατάσταση που πρόκειται να μοντελοποιήσουμε, καθώς κι αν χρειαστεί να θέσουμε ορισμένα πιο σαφή ερωτήματα αναφορικά με τη συγκεκριμένη κατάσταση.

Επινόηση του μοντέλου. Επινοούμε ένα μοντέλο, άλλοτε για να αναπαραστήσουμε απλά κάτι, άλλοτε για να εξηγήσουμε ή να προβλέψουμε ένα φαινόμενο. Ανάλογα με το σκοπό που θα εξυπηρετήσει το μοντέλο, θα σκεφτούμε διαφορετικά για το πώς θα το οικοδομήσουμε.

Προσδιορισμός των βασικών παραγόντων του μοντέλου. Όταν μοντελοποιούμε ένα φαινόμενο ή μια κατάσταση, χρειάζεται να σκεφτούμε πάνω στους παράγοντες που είναι σημαντικοί για το μοντέλο, έτσι ώστε να απλοποιήσουμε την αρχική συνήθως πολύπλοκη εικόνα.

Επειδή όμως εύκολα στη συνέχεια, ξεχνάμε ποιες αρχικές απλουστεύσεις κάναμε, είναι σημαντικό να κρατήσουμε σημειώσεις, κάνοντας μια αναλυτική καταγραφή των παραγόντων που λάβαμε υπόψη και των απλοποιήσεων που έγιναν.

Προσδιορισμός των σχέσεων ανάμεσα στους παράγοντες του μοντέλου. Εφόσον έχουμε αποφασίσει ποιοι θα είναι οι σημαντικοί παράγοντες από την προηγούμενη παράγραφο, έρχονται στο προσκήνιο διάφορα άλλα θέματα που έχουν να κάνουν με την αναπαράστασή τους. Δηλαδή θα πρέπει να σκεφτούμε τον τρόπο σύμφωνα με τον οποίο θα τους αναπαραστήσουμε μέσα στο μοντέλο, και κυρίως ποιες θα είναι οι σχέσεις που συνδέουν αυτούς τους παράγοντες.

Δοκιμή μοντέλου - Διερεύνηση και παρατήρηση. Για ένα δυναμικό μοντέλο, όταν δηλαδή μπορούμε να δούμε την εξέλιξη του φαινομένου που μοντελοποιούμε, η δοκιμή του μοντέλου γίνεται μέσω προσομοίωσης. Προσομοιώνοντας την κατάσταση, παρατηρούμε πώς συμπεριφέρεται το μοντέλο. Χρήσιμα δεδομένα μπορούμε επίσης να πάρουμε από γραφικές αναπαραστάσεις (όταν αυτές είναι διαθέσιμες), όπως πίνακες τιμών και γραφικές παραστάσεις.

Τα αποτελέσματα της δοκιμής του μοντέλου είναι τα αποτελέσματα της πειραματικής διαδικασίας. Είναι σημαντικό λοιπόν να κρατούνται τα απαραίτητα στοιχεία για τη συμπεριφορά του μοντέλου. Όταν το μοντέλο είναι στατικό, δηλαδή περιγράφει μια κατάσταση σε μια δεδομένη χρονική στιγμή, μπορούμε να το ελέγξουμε συγκρίνοντας το με άλλα μοντέλα της ίδιας κατάστασης.

Ο έλεγχος της αναπαράστασης γίνεται αν συγκρίνουμε το μοντέλο που έχουμε κατασκευάσει με μοντέλα της ίδιας κατάστασης που έχουν παράγει άλλοι και αν δώσουμε το μοντέλο σε κάποιον άλλο και ζητήσουμε να μας περιγράψει τι έχει καταλάβει, για να ελέγξουμε αν είναι σωστό, κατανοητό και πλήρες (αν περιέχει τις σωστές έννοιες και τις μεταξύ τους σχέσεις, αν αποδίδει ορθά και πιστά την πραγματικότητα).

Ερμηνεία αποτελεσμάτων. Η ερμηνεία των αποτελεσμάτων είναι ένα από τα πιο κρίσιμα στάδια. Μας επιτρέπει είτε να θεωρήσουμε ότι το μοντέλο που επινοήσαμε είναι κατάλληλο, είτε, κάτι που συμβαίνει πιο συχνά, να ξεκινήσουμε τη διαδικασία της βελτίωσης του μοντέλου. Χρήσιμα ερωτήματα σε αυτό το σημείο είναι τα εξής:

- Πώς λειτουργεί το μοντέλο και γιατί λειτουργεί έτσι;
- Τι συμβαίνει στον πραγματικό κόσμο και γιατί συμβαίνει έτσι;
- Τι μπορεί να εξηγήσει και τι μπορεί να προβλέψει;
- Ποια είναι η περιοχή της ισχύος του; Σε ποιες περιπτώσεις το μοντέλο λειτουργεί κατάλληλα και σε ποιες όχι;

Βελτίωση του μοντέλου. Σχεδόν πάντα βρισκόμαστε στην κατάσταση να βελτιώσουμε ή να τροποποιήσουμε ριζικά το μοντέλο που επινοήσαμε. Για τη βελτίωση του μοντέλου που επινοήσαμε είναι συχνά χρήσιμο να θέτουμε μια σειρά ερωτήματα. Χρήσιμα ερωτήματα σε αυτό το σημείο είναι τα εξής:

- Το μοντέλο συμπεριφέρεται όπως ο πραγματικός κόσμος;
- Το μοντέλο μας βοηθά να εξηγήσουμε τι συμβαίνει στον πραγματικό κόσμο ή να προβλέψουμε την εξέλιξή του;
- Μήπως το μοντέλο είναι πολύ απλό; Μήπως δεν έχει συμπεριληφθεί κάποιος παράγοντας που είναι σημαντικός;
- Μήπως είναι πολύ σύνθετο;

- Οι σχέσεις που έχουν προσδιοριστεί είναι οι κατάλληλες;

5.3 Γλώσσα Ενιαίας Μοντελοποίησης (unified modeling language)

Μία γλώσσα μοντελοποίησης είναι μια διαγραμματική τεχνική, συνδυάζει απλά διαγράμματα που συνοδεύονται από επεξηγηματικό κείμενο, το οποίο μπορεί να είναι σε φυσική γλώσσα πχ. Ελληνικά, είτε σε κάποια πιο τυπική γλώσσα πχ. μια γλώσσα βασισμένη στη μαθηματική λογική.

Το πρόβλημα που υπάρχει στο πεδίο του λογισμικού είναι η ύπαρξη σύγχυσης, λόγω της ύπαρξης πολλών και διαφορετικών γλωσσών μοντελοποίησης, παρά το γεγονός ότι οι βασικές ιδέες είναι κοινές.

Σημαντική εξέλιξη, η οποία έδωσε λύση στο πρόβλημα είναι η δημιουργία της Ενοποιημένης Γλώσσας Μοντελοποίησης (Unified Modeling Language), η οποία είναι κατάλληλη για αντικειμενοστρεφή μοντελοποίηση.

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modelling Language) αποτελεί πρότυπη και την πλέον δημοφιλή γλώσσα για την αστικοποίηση, προσδιορισμό, ανάπτυξη και τεκμηρίωση συστημάτων λογισμικού και όχι μόνο (J. Warmer A. K., 1999).

Η πλούσια γραφική σημειολογία της UML σε συνδυασμό με τις δυνατότητες μοντελοποίησης που παρέχει, την καθιστούν ικανή να χρησιμοποιηθεί ακόμα και στην ανάπτυξη ενσωματωμένων συστημάτων.

5.3.1 Ιστορικά

Ως πρώτη αντικειμενοστρεφής γλώσσα προγραμματισμού αναγνωρίζεται η Simula (1967), αλλά η ευρεία διάδοση της αντικειμενοστρεφούς ανάπτυξης λογισμικού άρχισε στις αρχές της δεκαετίας του 1980 με γλώσσες όπως η Smalltalk, C++ και στη συνέχεια με τη Java.

Στη συνέχεια δημοσιεύτηκαν και οι πρώτες μεθοδολογίες ανάπτυξης λογισμικού βασισμένες στο αντικειμενοστρεφές μοντέλο, καταλήγοντας στις αρχές της δεκαετίας του ’90 σε μια πληθώρα τεχνικών, ορισμών, συμβολισμών και ορολογίας (Eriksson E.).

Η πρώτη επιτυχής προσπάθεια ενοποίησης των διαφόρων μεθοδολογιών ξεκίνησε από την Rational Software Corporation το 1994 με επικεφαλής τους Grady Booch και James Rumbaugh (Booch G., 1999).

Στόχος τους ήταν η δημιουργία μιας νέας μεθόδου, της «Ενοποιημένης Μεθόδου», η οποία θα ένωνε τη μέθοδο του Booch και την OMT 2 μέθοδο στην ανάπτυξη της οποίας επικεφαλής ήταν ο James Rumbaugh. Το 1995 ο Ivar Jacobson, ο άνθρωπος πίσω από τις μεθόδους OOSE και Objectory προστέθηκε στην ομάδα.

Οι μελλοντικοί σχεδιαστές της UML συνειδητοποίησαν ότι η δουλειά τους δε στόχευε στη δημιουργία μιας πρότυπης μεθόδου, αλλά μιας πρότυπης γλώσσας μοντελοποίησης και μετονόμασαν τη δουλειά τους σε «Unified Modeling Language».

Οι στόχοι της UML, όπως τέθηκαν από τους σχεδιαστές είναι οι εξής: η μοντελοποίηση συστημάτων (όχι μόνο λογισμικού) χρησιμοποιώντας αντικειμενοστρεφείς έννοιες. Η καθιέρωση μιας ρητής σύνδεσης σε εννοιολογικά καθώς και εκτελέσιμα κατασκευάσματα (artifacts).

Η αντιμετώπιση θεμάτων κλιμάκωσης, τα οποία είναι εγγενή σε σύνθετα, κρίσιμα συστήματα. Και τέλος, η δημιουργία μιας γλώσσας μοντελοποίησης χρησιμοποιήσιμης τόσο από ανθρώπους, όσο και από υπολογιστικές μηχανές (Selic, June 2000).

Η πρώτη έκδοση της UML (UML 1.1) δημοσιεύτηκε τον Ιανουάριο του 1997. Ακολούθησαν μερικές ήσσονος σημασίας εκδόσεις (UML 1.3, UML 1.4, UML 1.5), οι οποίες διόρθωσαν κάποιες ελλείψεις και σφάλματα της πρώτης έκδοσης.

Στη συνέχεια ακολούθησε η κύρια αναθεώρηση με τη UML 2.0, η οποία υιοθετήθηκε και καθιερώθηκε ως πρότυπο από την OMG (Object Management Group) το 2005, ενώ η τελευταία έκδοση είναι η UML 2.2.

5.3.2 Πεδίο εφαρμογής

Κατ' αρχάς πρέπει να τονιστεί ότι η UML συνδυάζει τις έννοιες των τεχνικών Booch, OMT και OOSE. Το αποτέλεσμα είναι μία, κοινή και ευρέως χρησιμοποιούμενη γλώσσα μοντελοποίησης για τους χρήστες αυτών και άλλων μεθόδων. Δεύτερον, η UML προχωράει πιο πέρα το ζήτημα του τι μπορεί να κάνει κανείς με τις μεθόδους αυτές.

Για παράδειγμα, οι συγγραφείς της UML έδωσαν σημασία στη μοντελοποίηση κατανεμημένων συστημάτων, ώστε να επιβεβαιώσουν ότι η UML θα μπορεί να αντιμετωπίσει με ευκολία τα συγκεκριμένα πεδία εφαρμογών. Τρίτον, η UML εστιάζει σε μία πρότυπη γλώσσα μοντελοποίησης, όχι σε μία πρότυπη διαδικασία μοντελοποίησης.

Αν και η UML πρέπει να εφαρμόζεται στα πλαίσια μίας διαδικασίας, από πείρα μπορούμε να καταλάβουμε ότι διαφορετικοί οργανισμοί και πεδία εφαρμογής απαιτούν διαφορετικές διαδικασίες. Για παράδειγμα, η διαδικασία ανάπτυξης ενός παιχνιδιού για υπολογιστές είναι ενδιαφέρουσα, αλλά διαφέρει από τη διαδικασία ανάπτυξης ενός συστήματος πραγματικού χρόνου για την αεροπορία που από την καλή λειτουργία του οποίου εξαρτώνται ανθρώπινες ζωές.

Για τους λόγους αυτούς, οι προσπάθειες έχουν επικεντρωθεί πρώτα στη δημιουργία ενός κοινού μετα-μοντέλου (που θα ενοποιεί τη σημασιολογία) και στη συνέχεια στη δημιουργία ενός κοινού συμβολισμού (που θα παρέχει την άποψη των ατόμων για τη σημασιολογία).

Η UML προδιαγράφει μία γλώσσα μοντελοποίησης που περιλαμβάνει τη συμφωνία της αντικειμενοστρεφούς κοινότητας σχετικά με βασικές έννοιες μοντελοποίησης. Επιτρέπει να εκφράζονται οι αποκλίσεις με χρήση των μηχανισμών επέκτασης και παρέχει τα ακόλουθα:

- Σημασιολογία και συμβολισμό για να αντιμετωπίζεται ένα ευρύ φάσμα σύγχρονων θεμάτων μοντελοποίησης με άμεσο και οικονομικό τρόπο.

- Σημασιολογία για να αντιμετωπιστούν ορισμένα από τα αναμενόμενα μελλοντικά ζητήματα μοντελοποίησης, τα οποία σχετίζονται με τους κατανεμημένους υπολογισμούς καθώς και την εκτελεσιμότητα.

- Μηχανισμούς επέκτασης ώστε συγκεκριμένα έργα να μπορούν να επεκτείνουν το μετα-μοντέλο για την εφαρμογή τους με χαμηλό κόστος. Δε θέλουμε να αλλάζουν οι χρήστες απευθείας το μετα-μοντέλο της UML.

- Μηχανισμούς επέκτασης, ώστε να είναι δυνατό να ενσωματωθούν οι μελλοντικές προσεγγίσεις μοντελοποίησης μέσα στη UML.

- Σημασιολογία για να διευκολύνεται η ανταλλαγή μοντέλων μεταξύ διαφορετικών εργαλείων.

- Σημασιολογία που να καθορίζει τη διασύνδεση με αποθηκευτικούς χώρους, για διαμοιρασμό και αποθήκευση των στοιχείων μοντελοποίησης.

5.3.3 Βασικά χαρακτηριστικά

Σύμφωνα με το επίσημο εγχειρίδιο αναφοράς, η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language ή UML) είναι μια γραφική γλώσσα γενικού σκοπού, η οποία χρησιμοποιείται για τον προσδιορισμό, οπτικοποίηση, ανάπτυξη και τεκμηρίωση των κατασκευασμάτων (artifacts) ενός συστήματος λογισμικού (J. Warmer A. K., 1999).

Η UML αποτελεί βιομηχανικό πρότυπο για τη μοντελοποίηση συστημάτων λογισμικού και χρησιμοποιείται στη μοντελοποίηση συστημάτων βασισμένων σε αντικείμενα (αντικειμενοστρεφή συστήματα).

Βασικό χαρακτηριστικό της είναι, ότι αποτελεί μια γλώσσα μοντελοποίησης ανεξάρτητη από τις μεθοδολογίες που χρησιμοποιούνται κατά την ανάπτυξη συστημάτων λογισμικού. Επιπλέον, δεν πρόκειται μόνο για μια τυποποίηση και ανακάλυψη μιας κοινής σημειολογίας, αλλά περιέχει νέες και ενδιαφέρουσες έννοιες που δεν υπάρχουν εν γένει στο πεδίο της αντικειμενοστρεφούς ανάπτυξης, όπως για παράδειγμα η περιγραφή και χρησιμοποίηση προτύπων (patterns) σε μια γλώσσα μοντελοποίησης, η χρησιμοποίηση της έννοιας του στερεοτύπου (stereotype) για την επέκταση της γλώσσας, η παροχή πλήρους ιχνηλασιμότητας (traceability) από τα εννοιολογικά μοντέλα ενός συστήματος στα εκτελέσιμα συστατικά της

φυσικής αρχιτεκτονικής. Συνεπώς, η κατανόηση της UML δεν περιορίζεται στην εκμάθηση των συμβόλων και της σημασίας τους, αλλά εκτείνεται σε ευρύτερο πλαίσιο στη μάθηση της αντικειμενοστρεφούς μοντελοποίησης (Chen R., 2002).

Η UML χρησιμοποιείται για τη μοντελοποίηση μεγάλου εύρους συστημάτων. Ο στόχος της UML είναι να περιγράφει κάθε τύπο συστήματος, μέσα από αντικειμενοστρεφή διαγράμματα. Η πιο συνήθης χρήση της είναι η παραγωγή μοντέλων συστημάτων λογισμικού, πέρα από αυτό όμως χρησιμοποιείται για την περιγραφή συστημάτων που δεν αφορούν λογισμικό, όπως για παράδειγμα μηχανικών συστημάτων.

Οι κυριότερες κατηγορίες συστημάτων στα οποία χρησιμοποιείται η UML είναι οι εξής: πληροφοριακά συστήματα, τεχνολογικά συστήματα, συστήματα λογισμικού, ενσωματωμένα συστήματα πραγματικού χρόνου, κατανεμημένα συστήματα, καθώς και συστήματα επιχειρήσεων.

5.3.4 UML σε σχέση με άλλες γλώσσες μοντελοποίησης

Η UML είναι πιο εκφραστική, αλλά και πιο ξεκάθαρη και ενοποιημένη από γλώσσες όπως η Booch, OMT και OOSE. Αυτό σημαίνει ότι όταν μεταφερόμαστε στη UML κερδίζουμε γιατί μπορούμε να μοντελοποιήσουμε έργα που δεν μπορούσαμε πριν. Επίσης υπάρχει κέρδος γιατί αφαιρούνται οι άχρηστες διαφορές στο συμβολισμό και την ορολογία που δεν αφήνουν να φανούν οι ομοιότητες των προσεγγίσεων αυτών.

Σε σχέση με άλλες οπτικές γλώσσες μοντελοποίησης, συμπεριλαμβανομένων της μοντελοποίησης οντοτήτων-συσχετίσεων, της Business Reengineering Process (BRP), των διαγραμμάτων ροής και των γλωσσών που προσανατολίζονται στις καταστάσεις, η UML προσφέρει επιπλέον εκφραστικότητα καθώς και ολιστική ακεραιότητα.

Οι χρήστες άλλων γλωσσών μοντελοποίησης θα αντιμετωπίσουν μικρο-αλλαγές στους συμβολισμούς, αλλά αυτό δε σημαίνει τόσο ότι πρέπει να μάθουν εκ νέου κάποια πράγματα, όσο να ξεκαθαρίσουν κάποιες έννοιες στην υποκείμενη σημασιολογία. Αν έχουν επιτευχθεί οι

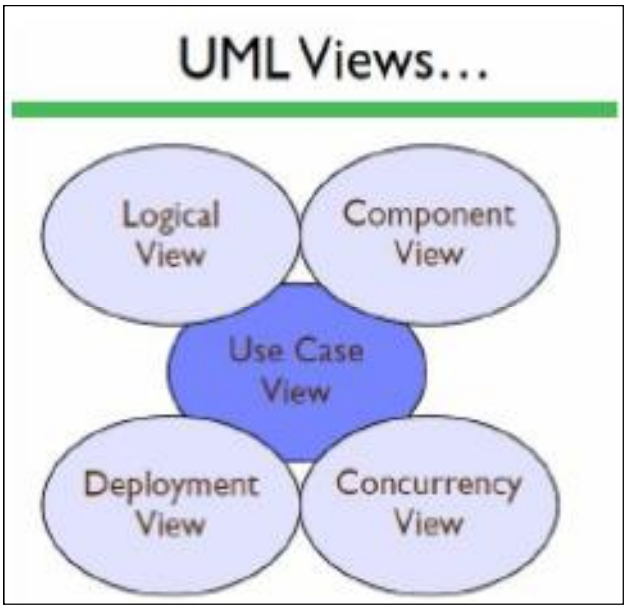
στόχοι της ενοποίησης, η UML αποτελεί την προφανή επιλογή για να ξεκινήσει κάποιος ένα νέο έργο, ειδικά όσο υπάρχουν όλο και περισσότερα εργαλεία, βιβλία και παροχή κατάρτισης.

Πολλά από τα εργαλεία οπτικής μοντελοποίησης υποστηρίζουν υπάρχοντες συμβολισμούς, όπως Booch, OMT, OOSE και άλλους, ως όψεις του υποκείμενου μοντέλου. Όταν τα εργαλεία αυτά προσφέρουν και την υποστήριξη στη UML, οι χρήστες θα μπορούν να μεταφέρουν τα υπάρχοντα μοντέλα τους στο συμβολισμό της UML, χωρίς να χάσουν πληροφορία.

5.3.5 UML Views

Ένα σύστημα αναπαρίσταται με τη βοήθεια των μοντέλων της UML. Το κάθε μοντέλο περιγράφει το σύστημα από μια ευδιάκριτα διαφορετική οπτική γωνία, ή αλλιώς όψη (view). Η όψη δεν αποτελεί γράφημα, αλλά μια αφαιρετική έννοια με την οποία συνδέεται ένας αριθμός διαγραμμάτων.

Οι όψεις επίσης συνδέουν τη γλώσσα μοντελοποίησης με τη μέθοδο που επιλέγεται για την ανάπτυξη του συστήματος. Κάθε όψη περιγράφεται από έναν αριθμό διαγραμμάτων, τα οποία περιέχουν πληροφορία που αφορά σε συγκεκριμένη οπτική γωνία του συστήματος.



Εικόνα 20. Οι όψεις της UML.

Η όψη περιπτώσεων χρήσης (Use case view): Περιγράφει τη λειτουργικότητα του συστήματος, έτσι όπως αυτή γίνεται αντιληπτή από τους εξωτερικούς χειριστές (actors). Ο χειριστής ο οποίος αλληλεπιδρά με το σύστημα μπορεί να είναι ένας χρήστης ή ένα άλλο υποσύστημα. Η όψη περιπτώσεων χρήσης περιγράφεται μέσα από διαγράμματα περιπτώσεων χρήσης (use case diagrams) και ενίοτε από διαγράμματα δραστηριοτήτων (activity diagrams).

Η επιθυμητή λειτουργικότητα του συστήματος περιγράφεται από έναν αριθμό περιπτώσεων χρήσης, όπου περίπτωση χρήσης είναι η περιγραφή μιας συγκεκριμένης λειτουργίας που απαιτείται από το σύστημα. Η όψη αυτή είναι σημαντική διότι επηρεάζει τις υπόλοιπες όψεις, μιας και ο τελικός στόχος είναι το σύστημα να παρέχει τη λειτουργικότητα που περιγράφεται σε αυτήν.

Ένα επιπλέον χαρακτηριστικό της όψης αυτής είναι ότι μπορεί να ελεγχθεί εύκολα η ορθότητά της μέσω των πελατών και κατ' επέκταση να επικυρωθεί το σύστημα.

Η Λογική όψη (Logical view): Περιγράφει το πώς παρέχεται η λειτουργικότητα του συστήματος, από την άποψη της στατικής δομής και δυναμικής συμπεριφοράς του. Σε αντίθεση με την όψη περιπτώσεων χρήσης, επικεντρώνεται στο εσωτερικό του συστήματος.

Η στατική δομή περιγράφεται μέσα από το διάγραμμα κλάσεων, ενώ η δυναμική συμπεριφορά μέσα από τα διαγράμματα καταστάσεων, ακολουθίας, συνεργασίας και δραστηριοτήτων.

Η όψη συστατικών (Component view): Περιγράφει την οργάνωση και υλοποίηση των εκτελέσιμων συστατικών και των εξαρτήσεών τους. Περιγράφεται μέσα από το διάγραμμα συστατικών.

Η όψη αυτή μπορεί να περιέχει επίσης επιπλέον πληροφορία σχετικά με τα συστατικά, όπως κατανομή των πόρων, ή διοικητική πληροφορία, όπως μια αναφορά προόδου της πορείας της εργασίας.

Η όψη ταυτοχρονισμού (Concurrency view): Περιγράφει τον ταυτοχρονισμό του συστήματος. Αυτή η όψη, που αποτελεί μια μη λειτουργική ιδιότητα του συστήματος, εκφράζει την αποδοτική χρησιμοποίηση πόρων, την παράλληλη εκτέλεση, και το χειρισμό των ασύγχρονων γεγονότων που προέρχονται από το περιβάλλον.

Πέρα από τη διαίρεση του συστήματος σε ταυτοχρόνως εκτελέσιμα νήματα ελέγχου, πρέπει επίσης να χειριστούν θέματα επικοινωνίας και συγχρονισμού των νημάτων. Η όψη αυτή περιγράφεται μέσα από δυναμικά διαγράμματα (διαγράμματα καταστάσεων, ακολουθίας,

συνεργασίας και δραστηριοτήτων), καθώς και διαγράμματα υλοποίησης (διαγράμματα συστατικών και ανάπτυξης).

Η όψη ανάπτυξης (Deployment view): Περιγράφει την ανάπτυξη του συστήματος σε φυσικό επίπεδο, δηλαδή τους κόμβους (υπολογιστές) στους οποίους τρέχουν τα συστατικά της εφαρμογής, καθώς και το πώς συνδέονται μεταξύ τους. Περιγράφεται από το διάγραμμα ανάπτυξης.

5.3.6 Διαγράμματα της UML

Τα κυριότερα διαγράμματα της UML είναι τα εξής:

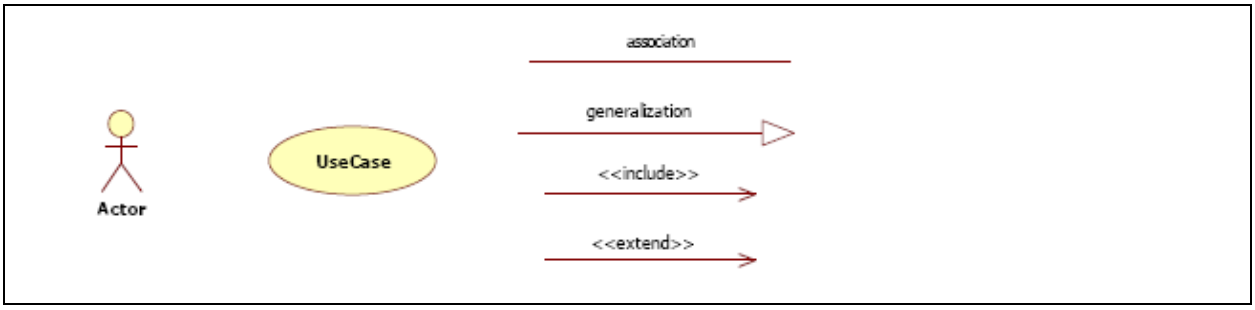
- Διάγραμμα περιπτώσεων χρήσης (Use case diagram).
- Διάγραμμα κλάσεων (Class diagram).
- Διάγραμμα ακολουθίας (Sequence diagram).
- Διάγραμμα συνεργασίας (Collaboration diagram).
- Διάγραμμα καταστάσεων (Statechart diagram).
- Διάγραμμα δραστηριότητας (Activity diagram).
- Διάγραμμα συστατικών (Component diagram).
- Διάγραμμα ανάπτυξης (Deployment diagram).
- Διάγραμμα πακέτων (Package diagram).

Παρακάτω περιγράφονται αναλυτικά τα παραπάνω διαγράμματα:

5.3.6.1 Διάγραμμα περιπτώσεων χρήσης

Το διάγραμμα περιπτώσεων χρήσης στη UML χρησιμοποιείται για την μοντελοποίηση της λειτουργικότητας ενός συστήματος, όπως αυτή γίνεται αντιληπτή από τον εξωτερικό χρήστη. Στην ανάλυση απαιτήσεων απεικονίζουν την οπτική χρήση στην συμπεριφορά του συστήματος. Τα διαγράμματα αυτά διαμερίζουν τη λειτουργικότητα του συστήματος σε συναλλαγές που έχουν νόημα για τους χρήστες του συστήματος ή αλλιώς χειριστές (actors).

Τα επιμέρους τμήματα της λειτουργικότητας ονομάζονται περιπτώσεις χρήσης (use cases). Το σύνολο των περιπτώσεων χρήσης συνιστούν τη συμπεριφορά του συστήματος. Τα βασικά διαγραμματικά στοιχεία του διαγράμματος περιπτώσεων χρήσης είναι το σύστημα, ο χειριστής, η περίπτωση χρήσης και οι σχέσεις μεταξύ τους. Τα στοιχεία αυτά φαίνονται παρακάτω, στο σχήμα που ακολουθεί.



Εικόνα 21. Βασικά στοιχεία διαγραμμάτων περίπτωσης χρήσης.

Η αξία του διαγράμματος περιπτώσεων χρήσης είναι ιδιαίτερα σημαντική, διότι καθορίζει τις λειτουργικές απαιτήσεις, οι οποίες θα αποτελέσουν σημείο αναφοράς καθ' όλη τη διάρκεια ανάπτυξης του συστήματος. Ο σημαντικότερος ρόλος του συγκεκριμένου διαγράμματος είναι ότι αποτελεί ένα μέσο επικοινωνίας μεταξύ πελατών και σχεδιαστών, όσον αφορά στη λειτουργικότητα του συστήματος. Η απλότητα των συμβολισμών το καθιστά ιδανικό για αυτό το σκοπό, παρέχοντας τη δυνατότητα εύκολης αντίληψης του συνόλου των λειτουργιών καθώς και εύκολης τροποποίησής τους.

Ο **χειριστής** αντιπροσωπεύει μια εξωτερική οντότητα, άνθρωπο ή σύστημα, η οποία αλληλεπιδρά με το σύστημα. Ο χειριστής αναπαριστά ένα ρόλο, όχι έναν μεμονωμένο χρήστη

του συστήματος, μιας και ο ίδιος χρήστης μπορεί να αλληλεπιδρά με το σύστημα με πολλαπλούς ρόλους. Οι χειριστές είναι κλάσεις με το στερεότυπο «actor», όπου το όνομα της κλάσης γενικά αναπαριστά το ρόλο του χειριστή. Το σύμβολο του χειριστή φαίνεται στο παραπάνω σχήμα.

Στο διάγραμμα περιπτώσεων χρήσης χρησιμοποιείται μόνο η σχέση **γενίκευσης** ανάμεσα σε χειριστές, προκειμένου να περιγραφεί η κοινή συμπεριφορά ανάμεσα τους, την οποία και κληρονομούν από μια πρόγονο κλάση χειριστή.

Ο τυπικός ορισμός μιας **περίπτωσης χρήσης** είναι *μια ακολουθία ενεργειών που πραγματοποιείται από το σύστημα για την παραγωγή μετρήσιμων αποτελεσμάτων που έχουν νόημα για τον χρήστη*. Η περίπτωση χρήσης ορίζει ένα συγκεκριμένο τρόπο χρησιμοποίησης του συστήματος, προσδιορίζοντας την αλληλεπίδραση ανάμεσα σε έναν ή περισσότερους χειριστές και το σύστημα. Το στιγμιότυπο μιας περίπτωσης χρήσης ονομάζεται σενάριο (scenario), και αναπαριστά ένα συγκεκριμένο μονοπάτι εκτέλεσης (execution path) μέσα στο σύστημα. Η περίπτωση χρήσης έχει τα ακόλουθα χαρακτηριστικά:

- Ξεκινάει πάντα από ένα χειριστή.
- Πρέπει να επιστρέφει κάποιου είδους απτή πληροφορία στο χρήστη.
- Μια περίπτωση χρήσης είναι πλήρης, με την έννοια ότι αποτελεί μια πλήρη περιγραφή.

Μια περίπτωση χρήσης δε θεωρείται ότι έχει ολοκληρωθεί μέχρις ότου η τελική πληροφορία παραχθεί, ακόμη κι αν απαιτούνται γι’ αυτό πολλαπλές αλληλεπιδράσεις μεταξύ των αντικειμένων. Ένα σύνηθες λάθος είναι η διαίρεση μίας περίπτωσης χρήσης σε μικρότερες, οι οποίες παράγουν ενδιάμεσα αποτελέσματα.

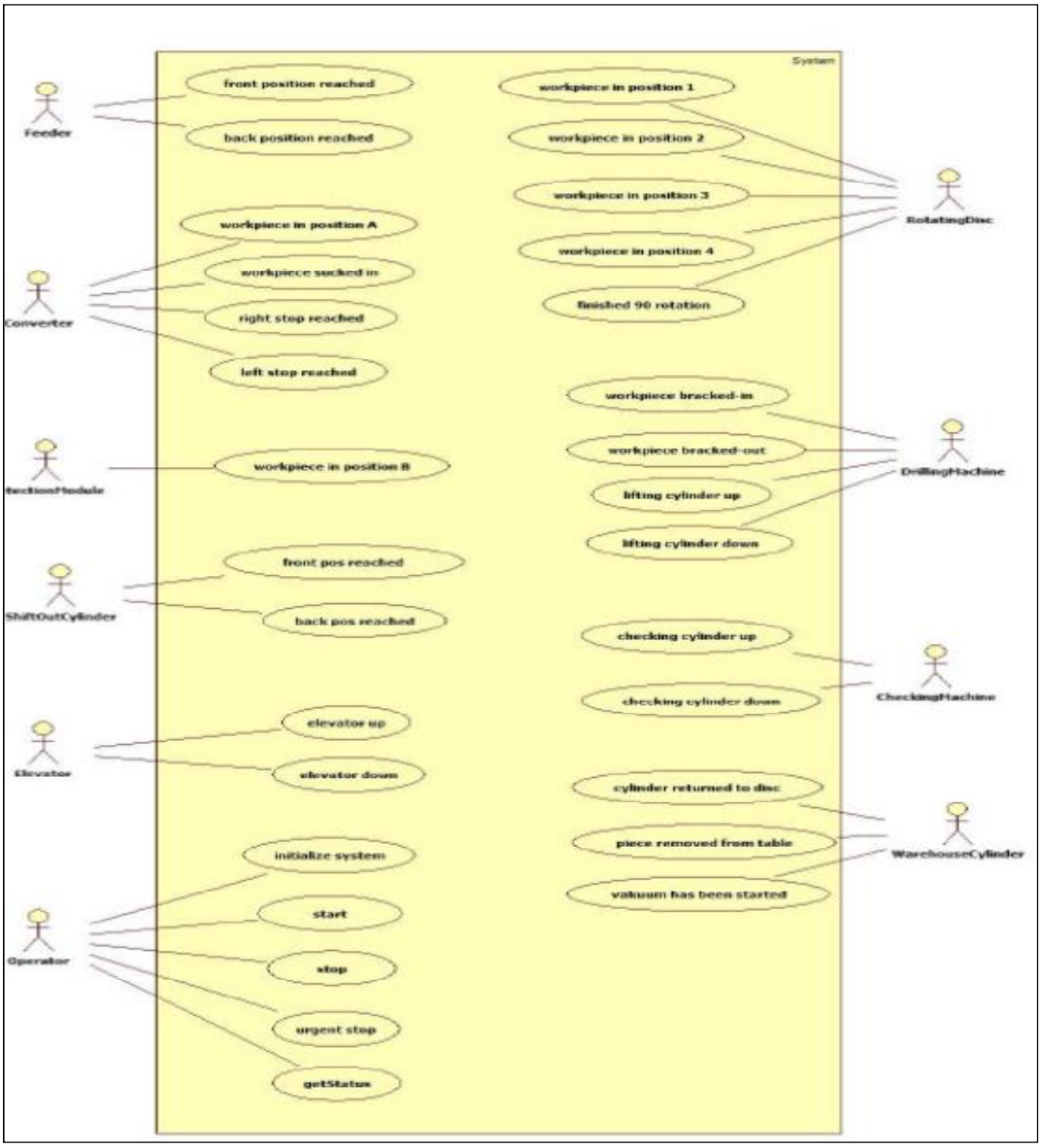
Ανάμεσα στις περιπτώσεις χρήσης υπάρχουν τρία είδη σχέσεων: η επέκταση (extends), η συμπερίληψη (uses ή includes) και η ομαδοποίηση (grouping). Η σχέση της **επέκτασης** είναι μια σχέση γενίκευσης που χρησιμοποιείται στην περίπτωση όπου μια περίπτωση χρήσης συμπεριλαμβάνει ένα τμήμα, όχι απαραίτητα ολόκληρη, την συμπεριφορά της περίπτωσης

χρήσης που επεκτείνει. Τέτοιου είδους περιπτώσεις χρήσης χρησιμοποιούνται στο χειρισμό εξαιρέσεων.

Η σχέση της **συμπερίληψης** είναι και αυτή μια σχέση γενίκευσης που χρησιμοποιείται στην περίπτωση όπου μια περίπτωση χρήσης συμπεριλαμβάνει την πλήρη λειτουργικότητα μιας άλλης. Όταν ένα σύνολο περιπτώσεων χρήσης παρουσιάζουν σε κάποια τμήματα κοινή συμπεριφορά, η σχέση αυτή χρησιμοποιείται για τη μοντελοποίηση αυτής της κοινής συμπεριφοράς σε μια περίπτωση χρήσης που χρησιμοποιείται από τις υπόλοιπες.

Τέλος με τη σχέση της **ομαδοποίησης**, περιπτώσεις χρήσης, οι οποίες διαθέτουν παρόμοια συμπεριφορά ή σχετίζονται με κάποιο τρόπο μεταξύ τους, οργανώνονται σε πακέτα. Ωστόσο, για λόγους απλότητας των διαγραμμάτων περιπτώσεων χρήσης η τελευταία σχέση συνήθως δε χρησιμοποιείται.

Στο επόμενο σχήμα, μπορούμε να δούμε ένα παράδειγμα διαγράμματος χρήσης ενός πληροφοριακού συστήματος με πολλές κατηγορίες χρηστών.



Εικόνα 22. Παράδειγμα διαγράμματος χρήσης πληροφοριακού συστήματος.

Η περιγραφή των περιπτώσεων χρήσης γίνεται με τη μορφή κειμένου στην ορολογία του χρήστη και αποτελεί μια απλή και συνεπή τεκμηρίωση. Η τεκμηρίωση κάθε περίπτωσης χρήσης, για να είναι πλήρης, θα πρέπει να περιλαμβάνει μια ακολουθία γεγονότων που λαμβάνουν χώρα για την υλοποίηση της επιθυμητής συμπεριφοράς. Επικεντρώνεται στην

εξωτερική συμπεριφορά του συστήματος, αγνοώντας τον τρόπο υλοποίησης και την εσωτερική δομή του.

Μερικά σημεία τα οποία θα πρέπει να συμπεριλαμβάνονται στην περιγραφή είναι ο στόχος της περίπτωσης χρήσης, από ποιόν χειριστή ξεκινάει, η ακολουθία των μηνυμάτων μεταξύ χειριστή και συστήματος, εναλλακτική ροή γεγονότων σε περιπτώσεις εξαιρέσεων, και τέλος το πώς η περίπτωση χρήσης τερματίζεται επιστρέφοντας κάποια τιμή στο χειριστή (Booch G., 1999)..

5.3.6.2 Διάγραμμα κλάσεων

Τα διαγράμματα περιπτώσεων χρήσης που είδαμε προηγουμένως είναι διαγράμματα καταγραφής προδιαγραφών και είναι χρήσιμα για κάθε τύπο συστήματος. Στην ανάλυση απαιτήσεων απεικονίζει τις βασικές οντότητες συσχετίσεων και τις μεταξύ τους σχέσεις, ενώ στο σχεδιασμό απεικονίζουν την ενσωμάτωση της δομής του συστήματος.

Σε ένα αντικειμενοστρεφές σύστημα τα δομικά στοιχεία του είναι οι κλάσεις και οι σχέσεις μεταξύ των κλάσεων, οι οποίες επιτρέπουν τη συνεργασία αντικειμένων που δημιουργούνται ως στιγμιότυπα των κλάσεων.

Το διάγραμμα κλάσεων αποτελείται από τις κλάσεις του συστήματος και τις μεταξύ τους συσχετίσεις, περιγράφοντας με αυτό τον τρόπο τη στατική δομή του συστήματος.

Το διάγραμμα κλάσεων μπορεί να χρησιμοποιηθεί σε διάφορες φάσεις της ανάπτυξης του συστήματος. Στο αρχικό στάδιο της ανάλυσης απαιτήσεων οι κατασκευαστές αρχίζουν να αποκτούν γνώση για το πεδίο του προβλήματος του συστήματος.

Αυτή η αρχική κατανόηση των εννοιών του πεδίου του προβλήματος καταγράφεται σε ένα διάγραμμα κλάσεων, το οποίο ονομάζεται μοντέλο του πεδίου προβλήματος (problem domain model). Στο μοντέλο αυτό καταγράφονται ως κλάσεις οι έννοιες του πεδίου του προβλήματος και οι μεταξύ τους συσχετίσεις.

Έπειτα, στο στάδιο της ανάλυσης, με οδηγό το μοντέλο του πεδίου προβλήματος, κατασκευάζεται ένα διάγραμμα κλάσεων, το οποίο αναπαριστά τη βασική αρχιτεκτονική δομή

του συστήματος. Σε αυτό το στάδιο οι κλάσεις πρέπει να επιδιώκουν την αναπαράσταση του συστήματος που μοντελοποιείται με την ελάχιστη δυνατή πληροφορία, χωρίς να επιχειρείται αναφορά σε θέματα υλοποίησης.

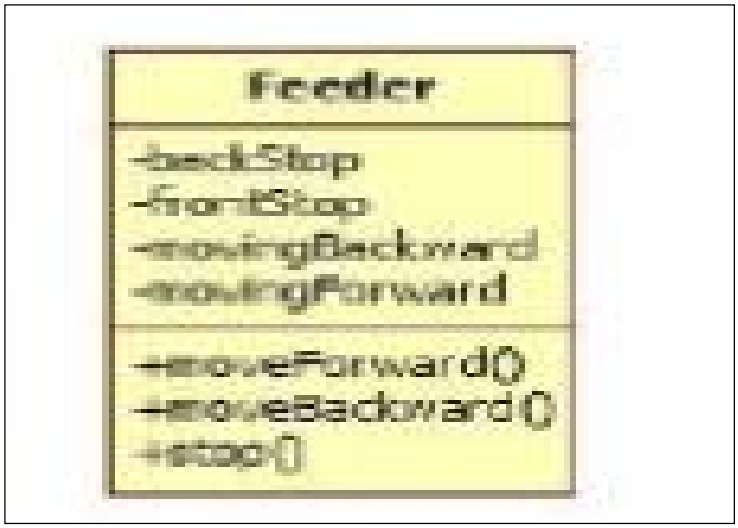
Στη συνέχεια, μεταβαίνοντας στο στάδιο της σχεδίασης, η περιγραφή των κλάσεων συμπληρώνεται με τις λειτουργίες που υλοποιούν τη συμπεριφορά των αντικειμένων και με επιπρόσθετες ιδιότητες ή συσχετίσεις, που επιβάλλονται από το περιβάλλον υλοποίησης.

Τέλος, κατά την υλοποίηση του συστήματος, είναι δυνατόν να επέλθουν τροποποιήσεις στη δομή των κλάσεων λόγω απαιτήσεων που σχετίζονται με απόκρυψη πληροφορίας, ορατότητα και άλλες μη λειτουργικές απαιτήσεις, όπως π.χ. απόδοση και ασφάλεια.

Σε μερικές περιπτώσεις, το διάγραμμα κλάσεων είναι το μόνο είδος διαγράμματος της UML που χρησιμοποιείται, λόγω των πληροφοριών που παρέχει σχετικά με τον πηγαίο κώδικα. Όπως θα αναφερθεί παρακάτω, υπάρχει η δυνατότητα αυτόματης παραγωγής τμημάτων κώδικα από το διάγραμμα κλάσεων, καθώς και η αυτόματη δημιουργία διαγραμμάτων κλάσεων λαμβάνοντας ως είσοδο τον πηγαίο κώδικα.

Για το λόγο αυτό, ο κάθε συμβολισμός είναι σημαντικός, ακόμα κι αν υποδηλώνεται με ένα στοιχειώδες σύμβολο στο διάγραμμα κλάσεων. Στη συνέχεια παρουσιάζονται τα βασικά διαγραμματικά στοιχεία ενός διαγράμματος κλάσεων.

Οι κλάσεις αποτελούν τη βάση της κατασκευής οποιουδήποτε αντικειμενοστρεφούς συστήματος. Ενσωματώνουν δεδομένα, καθώς και τις λειτουργίες που επενεργούν στα δεδομένα αυτά. Αν μια κλάση είναι αφηρημένη (abstract class) το όνομα της κλάσης σημειώνεται με πλάγιους χαρακτήρες. Στο παρακάτω σχήμα φαίνεται η κλάση που αναπαριστά το φυσικό αντικείμενο Feeder.



Εικόνα 23. Παράδειγμα κλάσης.

Το συντακτικό για τη δήλωση ιδιοτήτων στη UML είναι:

ορατότητα όνομα : Τύπος = Αρχική Τιμή

Ενώ το συντακτικό για τη δήλωση λειτουργιών είναι:

ορατότητα όνομα (λίστα παραμέτρων): Επιστρεφόμενος τύπος

Η ορατότητα απεικονίζεται με τα σύμβολα “-”, “+”, “#”,“~” τα οποία δηλώνουν ιδιωτική, δημόσια, προστατευμένη καθώς και πρόσβαση σε επίπεδο πακέτου αντίστοιχα. Μια στατική ιδιότητα ή λειτουργία, που ανήκει δηλαδή στην κλάση και όχι στα στιγμιότυπά της, υποδηλώνεται στη UML υπογραμμίζοντας το όνομα της ιδιότητας ή της μεθόδου αντίστοιχα.

Μια συσχέτιση μεταξύ δύο κλάσεων απεικονίζει μια στατική σχέση μεταξύ τους. Αν η σχέση αυτή μεταξύ των κλάσεων υφίσταται σε μόνιμη βάση, τότε χρησιμοποιούμε τη συσχέτιση, ενώ αν η σχέση είναι παροδική (π.χ. όταν τα αντικείμενα μιας κλάσης είναι παράμετροι σε μια μέθοδο μιας άλλης κλάσης) χρησιμοποιούμε την εξάρτηση.

Ενώ μια συσχέτιση στη UML συνδέει δύο κλάσεις ενός μοντέλου, ένα στιγμιότυπο μιας συσχέτισης συνδέει δύο συγκεκριμένα στιγμιότυπα κλάσεων και ονομάζεται σύνδεση (link). Προαιρετικά μπορούμε να έχουμε σε μία συσχέτιση τα εξής στοιχεία:

- Όνομα συσχέτισης, το οποίο θα πρέπει να υποδηλώνει με σαφήνεια το νόημα της συσχέτισης.

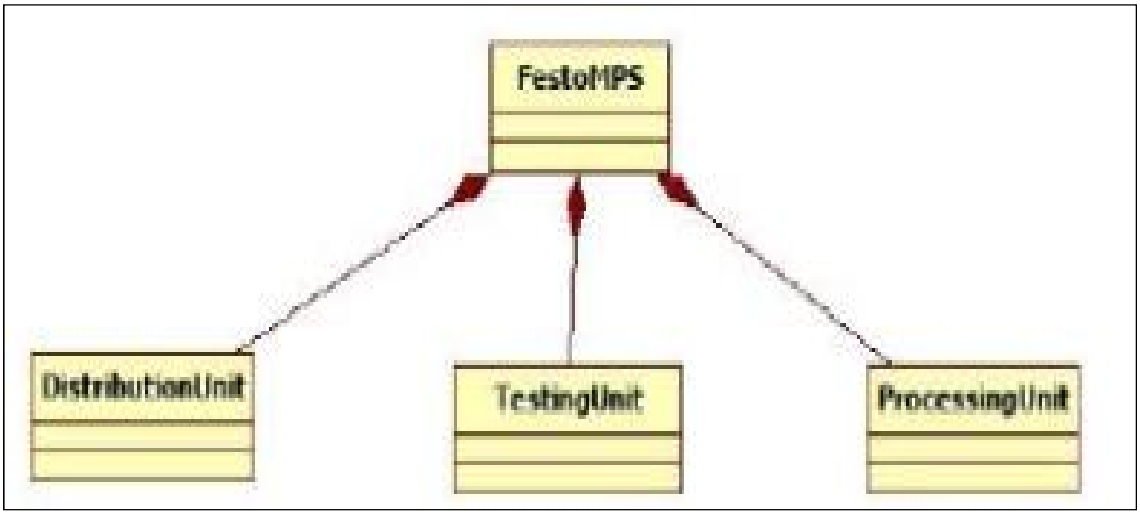
- Ονόματα άκρων συσχέτισης: τα οποία υποδηλώνουν το ρόλο αυτής της κλάσης στη συσχέτιση.

- Πολλαπλότητα: Η πολλαπλότητα αφορά σε ένα άκρο μιας συσχέτισης και είναι το πλήθος των αντικειμένων που μπορεί να σχετίζονται με ένα αντικείμενο της άλλης κλάσης.

Στη συνέχεια αναφέρονται διάφοροι ειδικότεροι τύποι συσχετίσεων.

Γενίκευση: Η γενίκευση είναι μια ειδική μορφή συσχέτισης, η οποία αποτελεί μια σχέση μεταξύ μιας γενικής περιγραφής και μιας ειδικότερης περιγραφής που την επεκτείνει. Η γενίκευση αξιοποιεί το μηχανισμό της κληρονομικότητας και επιτρέπει πολυμορφική συμπεριφορά.

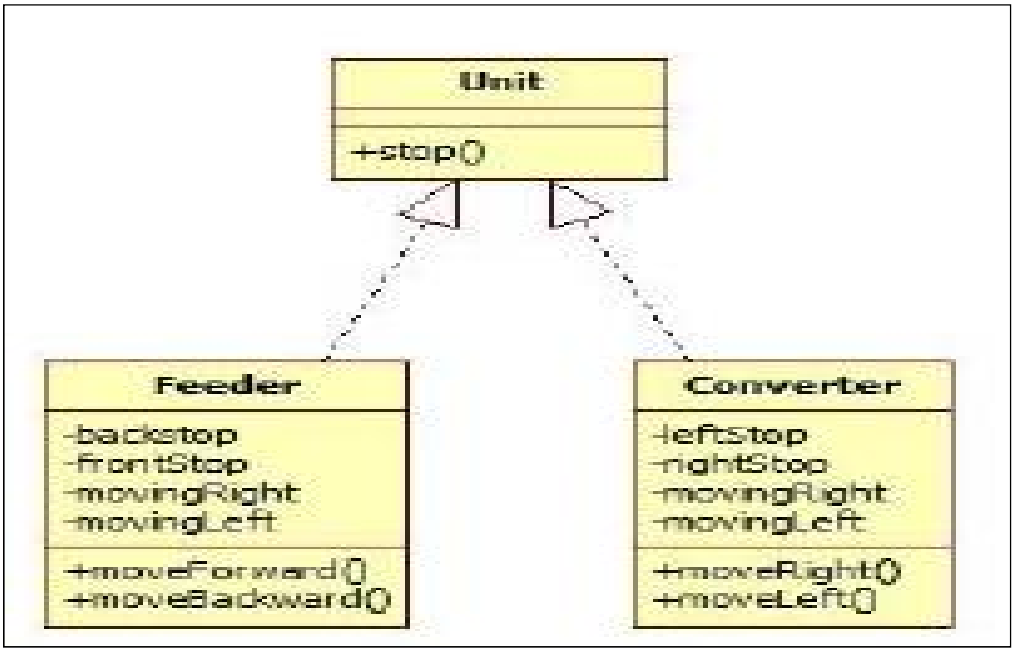
Συσσωμάτωση και σύνθεση: Η συσσωμάτωση είναι μια σχέση ειδικής μορφής που αναπαριστά μια σχέση συνόλου-τμήματος ή όλου-μέρους (whole-part). Η σύνθεση είναι μιας ισχυρότερης μορφής συσχέτιση στην οποία το σύνολο έχει την αποκλειστική ευθύνη διαχείρισης των τμημάτων, όπως τη δημιουργία και τη διαγραφή τους. Αν διαγραφεί για παράδειγμα η κλάση που αντιστοιχεί στο σύνολο, διαγράφονται και οι κλάσεις των τμημάτων.



Εικόνα 24. Παράδειγμα συσσωμάτωσης.

Εξάρτηση: Μια εξάρτηση υποδηλώνει σημασιολογική σχέση μεταξύ δύο ή περισσοτέρων στοιχείων ενός μοντέλου. Αν δυο κλάσεις Α και Β συνδέονται με μια σχέση εξάρτησης από την Α προς τη Β, υποδηλώνεται ότι, παρόλο που η κλάση Α δε δημιουργεί ούτε «έχει» τη Β, απαιτεί την ύπαρξη της Β για την αποστολή μηνυμάτων προς αυτή. Αν η κλάση Β τροποποιηθεί, ενδεχομένως να απαιτείται και η τροποποίηση της κλάσης Α.

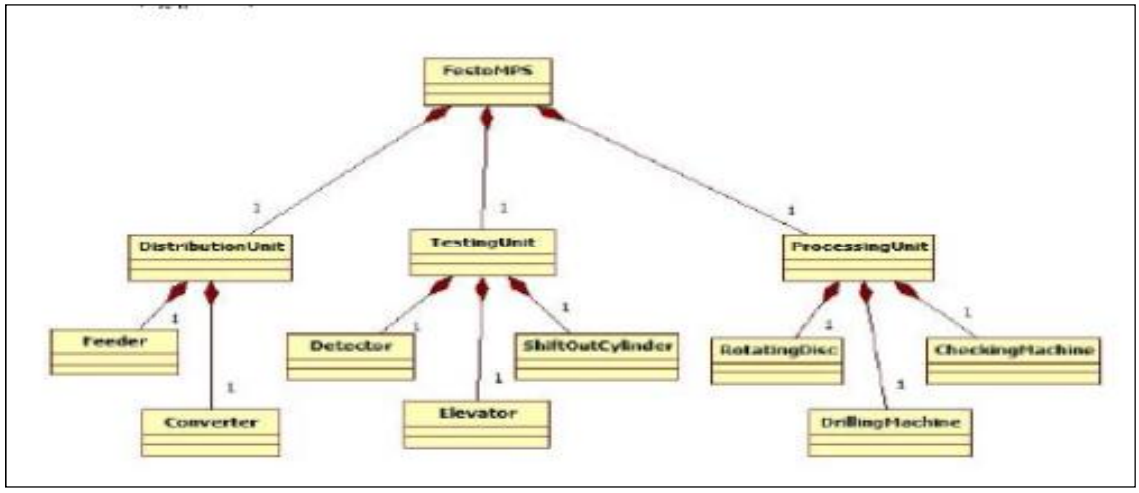
Διασύνδεση – σχέση πραγμάτωσης: Η διασύνδεση (interface) είναι ένας τύπος που παρέχει λειτουργίες που είναι στο σύνολό τους αφαιρετικές. Η κλάση η οποία πραγματώνει μια διασύνδεση συνδέεται μαζί της με τη σχέση πραγμάτωσης (realization). Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα σχέσης - πραγμάτωσης.



Εικόνα 25. Παράδειγμα σχέσης πραγμάτωσης.

Παρόλο που ένα μοντέλο στη UML αναπαρίσταται γραφικά, απαιτείται συχνά χρήση κειμένου για μέγιστη δυνατή διαφάνεια. Ένας περιορισμός είναι μια λογική συνθήκη (έκφραση Boole) που πρέπει να είναι αληθής για να λάβει χώρα μια ενέργεια ή για να υπάρξει μια συσχέτιση.

Η UML επιτρέπει τον καθορισμό περιορισμών με οποιοδήποτε τρόπο, αρκεί η περιγραφή να βρίσκεται μέσα σε άγκιστρα {}. Ωστόσο, η UML περιλαμβάνει τον ορισμό μιας τυπικής γλώσσας περιορισμών (Object Constraint Language – OCL) (Booch G., 1999).



Εικόνα 26. Παράδειγμα συσσωμάτωσης.

5.3.6.3 Διάγραμμα ακολουθίας

Το διάγραμμα ακολουθίας παρουσιάζει την αλληλεπίδραση μεταξύ αντικειμένων σε δύο διαστάσεις. Η κάθετη διάσταση αντιστοιχεί στην κλίμακα του χρόνου, ενώ στην οριζόντια διάσταση συμβολίζονται τα ανεξάρτητα αντικείμενα.

Τα αντικείμενα συμβολίζονται με παραλληλόγραμμα μέσα στα οποία μπορεί να σημειωθεί το όνομα του στιγμιότυπου του αντικειμένου που συμμετέχει στο σενάριο που απεικονίζεται και ακολουθεί μετά από άνω-κάτω τελεία το όνομα της κλάσης στην οποία ανήκει το αντικείμενο.

Σε κάθε αντικείμενο αντιστοιχεί μια κάθετη γραμμή που ονομάζεται γραμμή ζωής (lifeline). Τα αντικείμενα ανταλλάσουν μηνύματα, τα οποία στην επίσημη ορολογία της UML ονομάζονται **ερεθίσματα** (stimuli).

Ένα μήνυμα που αποστέλλεται μεταξύ των αντικειμένων συμβολίζεται ως ένα βέλος από τη γραμμή ζωής ενός αντικειμένου προς τη γραμμή ζωής ενός άλλου. Μήνυμα μπορεί να είναι οτιδήποτε από τα εξής:

Κλήση μιας λειτουργίας: όταν ένα αντικείμενο καλεί μια λειτουργία ενός άλλου αντικειμένου. Πρόκειται για σύγχρονο μήνυμα, δηλαδή ο αποστολέας του μηνύματος θα πρέπει να περιμένει

την ολοκλήρωση της λειτουργίας για να συνεχίσει. Η κεφαλή του βέλους είναι γεμισμένη με μαύρο χρώμα. Πάνω από το βέλος αναγράφεται το όνομα της λειτουργίας που καλείται, με τις ενδεχόμενες παραμέτρους σε παρενθέσεις. Ειδική περίπτωση κλήσης είναι η αυτοκλήση, η οποία ξεκινάει από το αντικείμενο και καταλήγει πάλι σε αυτό.

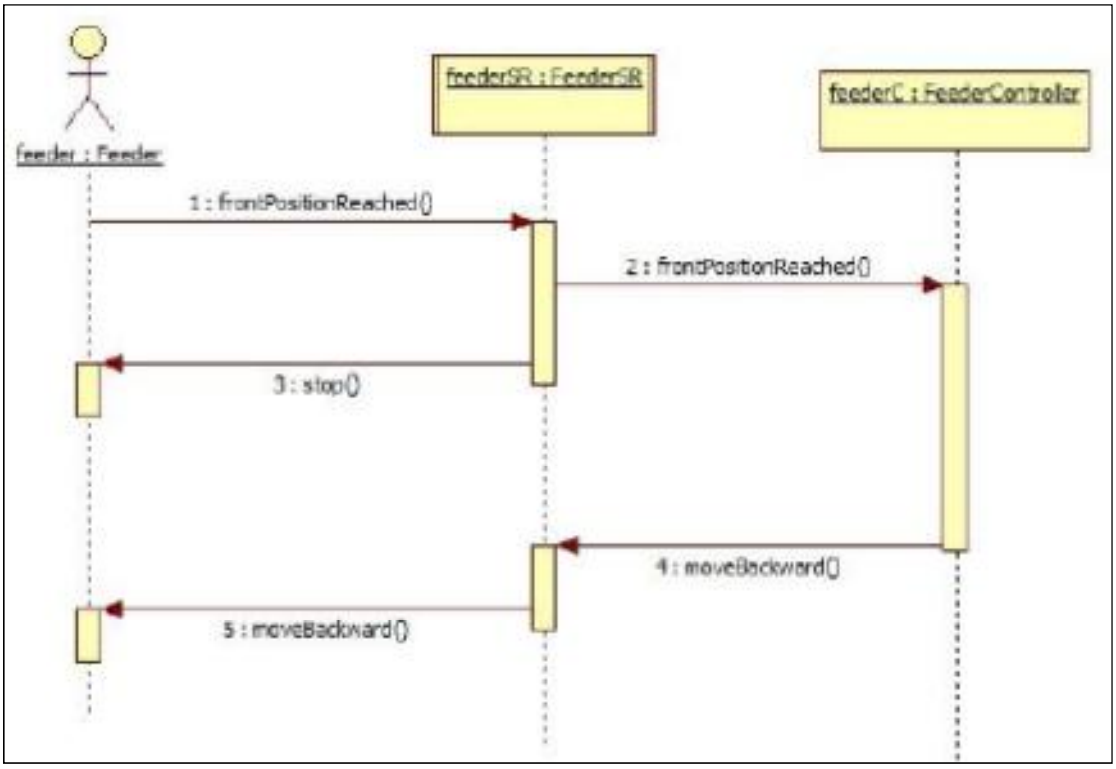
Σήμα: όταν ένα αντικείμενο αποστέλλει ένα ασύγχρονο μήνυμα σε ένα άλλο αντικείμενο. Τυπικά ασύγχρονα μηνύματα συναντάμε σε πολυνηματικές εφαρμογές, όπου ένα μήνυμα τοποθετείται σε κάποια ουρά ενός νήματος εκτέλεσης ενώ το ενεργό αντικείμενο-παραλήπτης θα επεξεργαστεί το μήνυμα σε κάποια επόμενη χρονική στιγμή. Η διαφορά με την κλήση λειτουργίας στον συμβολισμό είναι πως η κατάληξη είναι ένα ανοιχτό βέλος.

Επιστροφή κλήσης: είναι ένα διακεκομμένο βέλος το οποίο συμβολίζει την επιστροφή από μία κλήση λειτουργίας. Πάνω στο διακεκομμένο βέλος αναγράφεται συνήθως η τιμή επιστροφής, αν υπάρχει.

Μηνύματα υπό συνθήκη: Στα μηνύματα υπό συνθήκη τοποθετούνται αγκύλες μέσα στις οποίες αναγράφεται μία συνθήκη που μπορεί να είναι αληθής ή ψευδής. Η σημασία του συμβολισμού είναι ότι το μήνυμα θα αποσταλεί μόνο αν η συνθήκη είναι αληθής. Αν θέλουμε ταυτόχρονα να δείξουμε μια αποστολή εναλλακτικού μηνύματος στην περίπτωση που η συνθήκη είναι ψευδής, τότε δείχνουμε τα δύο αμοιβαία αποκλειόμενα μηνύματα σαν μηνύματα με το ίδιο σημείο εκκίνησης και γράφουμε στο πρώτο τη συνθήκη και στο δεύτερο τη φράση [else].

Σε ένα διάγραμμα ακολουθίας είναι δυνατόν να προστεθούν και περισσότεροι συμβολισμοί που υποδηλώνουν βρόχους επανάληψης, μηνύματα που αποστέλλονται πολλαπλές φορές, συγχρονισμό νημάτων και ούτω καθεξής.

Ωστόσο, ο σκοπός των διαγραμμάτων ακολουθίας δεν είναι να αποτυπώσουν τις λεπτομέρειες ενός αλγορίθμου αλλά να αναπαραστήσουν με απλό και κατανοητό τρόπο τα σενάρια συνεργασίας μεταξύ αντικειμένων (Booch G., 1999).



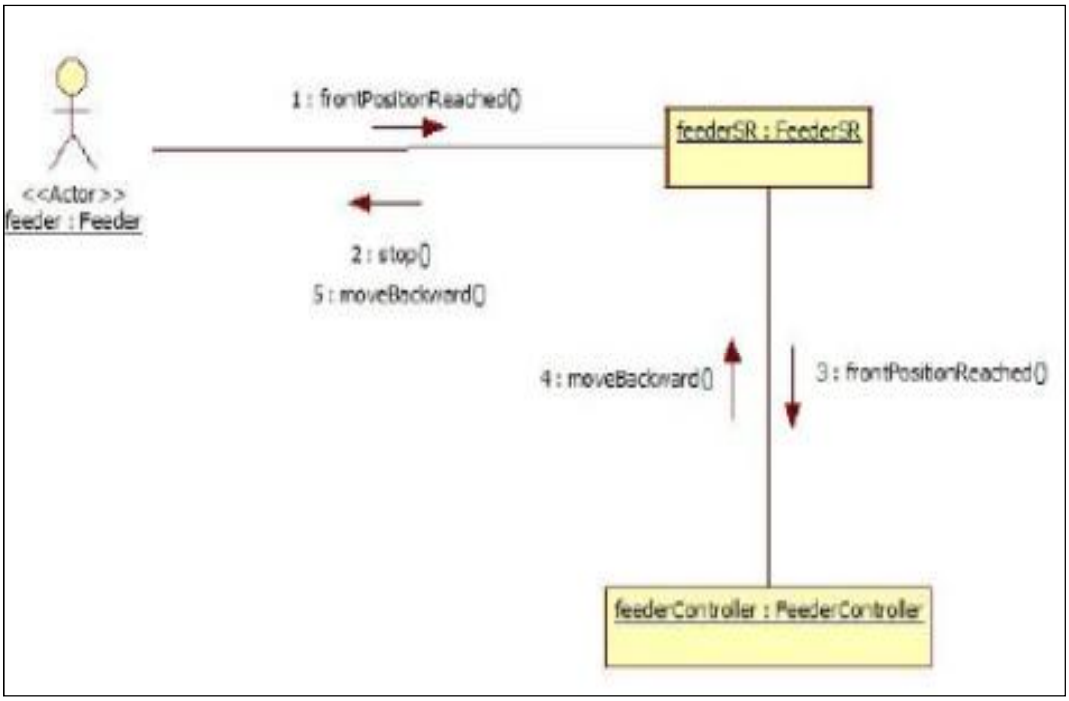
Εικόνα 27. Παράδειγμα διαγράμματος ακολουθίας.

5.3.6.4 Διάγραμμα συνεργασίας

Σε ένα διάγραμμα συνεργασίας απεικονίζονται τα συνεργαζόμενα αντικείμενα και οι συσχετίσεις μεταξύ τους. Ενώ τα διαγράμματα ακολουθίας απεικονίζουν κυρίως τη ροή των μηνυμάτων σε ένα σενάριο μιας περίπτωσης χρήσης, τα διαγράμματα συνεργασίας χρησιμοποιούνται για να παρουσιάσουν τις σχέσεις μεταξύ αντικειμένων.

Πλησίον των συνδέσεων εμφανίζονται ως μικρότερες ακμές τα μηνύματα που αποστέλλονται. Για να απεικονιστεί η ακολουθία των μηνυμάτων που ανταλλάσσονται χρησιμοποιείται αρίθμηση των μηνυμάτων.

Τα διαγράμματα ακολουθίας και συνεργασίας θεωρούνται συμπληρωματικά, καθώς περιέχουν την ίδια πληροφορία άλλα κάθε ένα δίνει μια διαφορετική οπτική γωνία (σε πολλά εργαλεία το ένα είδος διαγράμματος παράγεται αυτόματα από το άλλο).



Εικόνα 28. Παράδειγμα διαγράμματος συνεργασίας.

Από ένα τέτοιο διάγραμμα είναι εύκολο να αντιληφθεί κανείς την ομάδα των συνεργαζόμενων αντικειμένων και την ύπαρξη των συνδέσεων μεταξύ τους αλλά είναι δυσκολότερο να οπτικοποιηθεί η ροή των μηνυμάτων, η οποία φαίνεται καλύτερα στο διάγραμμα ακολουθίας (Booch G., 1999).

5.3.6.5 Διάγραμμα καταστάσεων

Το διάγραμμα καταστάσεων χρησιμοποιείται για την περιγραφή της ροής του ελέγχου σε ένα σύστημα εστιάζοντας στις αλλαγές κατάστασης που λαμβάνουν χώρα σε ένα αντικείμενο. Στην ανάλυση απαιτήσεων απεικονίζει τον κύκλο ζωής σημαντικών δραστηριοτήτων, ενώ στο σχεδιασμό απεικονίζει τον κύκλο ζωής βασικών κλάσεων. Πολύ συχνά οι προδιαγραφές ενός

συστήματος μπορούν να καθοριστούν βάσει μιας μηχανής πεπερασμένων καταστάσεων (finite state machine) ή απλά μηχανής καταστάσεων.

Συνήθως, μια μηχανή καταστάσεων περιγράφεται ως ένας γράφος όπου οι κόμβοι αντιστοιχούν σε καταστάσεις και τα βέλη υποδηλώνουν τη μετάβαση από μια κατάσταση σε μια άλλη. Εν γένει, οι μηχανές πεπερασμένων καταστάσεων είναι κατάλληλες για την περιγραφή σύγχρονων συστημάτων.

Συνήθως, ένα διάγραμμα καταστάσεων είναι προσαρτημένο σε μια κλάση και αποτελεί ένα μοντέλο όλων των δυνατών κύκλων ζωής ενός αντικειμένου της κλάσης. Κάθε αντικείμενο αντιμετωπίζεται ως ξεχωριστή οντότητα που επικοινωνεί με το περιβάλλον ανιχνεύοντας γεγονότα και αντιδρώντας σε αυτά.

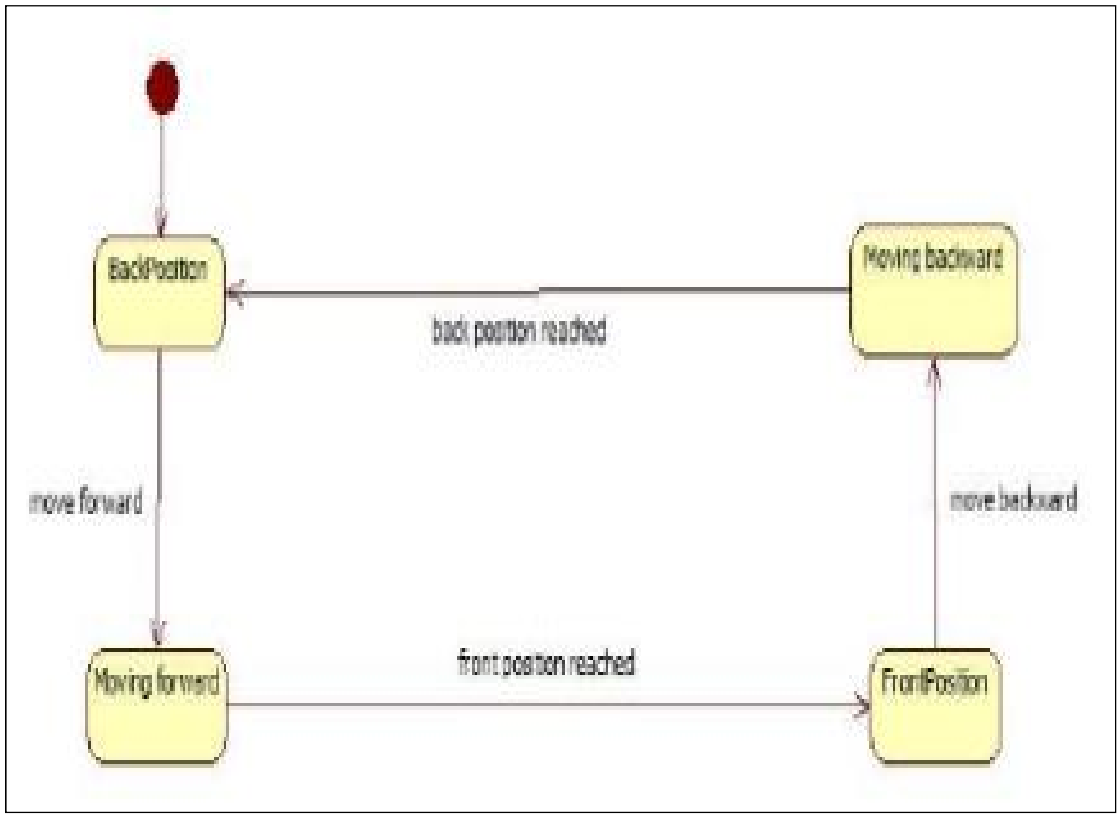
Όταν λαμβάνει χώρα ένα ανιχνεύσιμο γεγονός, το αντικείμενο αποκρίνεται με βάση την κατάσταση στην οποία βρίσκεται. Η εκτέλεση μιας ενέργειας μπορεί να οδηγήσει σε μετάβαση σε μια άλλη κατάσταση.

Σε ένα διάγραμμα καταστάσεων της UML απεικονίζονται γεγονότα, καταστάσεις και μεταβάσεις:

Ένα **γεγονός** (event) έχει χωρική και χρονική θέση στο σύστημα αλλά δεν έχει διάρκεια. Ένα γεγονός συμβολίζεται σημειώνοντας το όνομά του στις μεταβάσεις τις οποίες προκαλεί. Στην περίπτωση που κάποια ενέργεια πραγματοποιείται ταυτόχρονα με την εμφάνιση ενός γεγονότος, σημειώνεται μετά το όνομα του γεγονότος διαχωρισμένη με κάθετο “/”.

Μια **κατάσταση** (state) περιγράφει μια χρονική περίοδο κατά τη διάρκεια ζωής ενός αντικειμένου. Μπορεί να χαρακτηριστεί ως ένα σύνολο τιμών για τις ιδιότητες του αντικειμένου που είναι παρόμοιες από κάποια άποψη, ως μια περίοδος κατά την οποία ένα αντικείμενο αναμένει την εμφάνιση ενός γεγονότος, ή ως μία περίοδος κατά την οποία ένα αντικείμενο εκτελεί μια εργασία. Μια κατάσταση συμβολίζεται ως ένα ορθογώνιο με καμπύλες γωνίες. Ειδικά για το συμβολισμό της αρχικής κατάστασης ενός συστήματος χρησιμοποιείται ένας «γεμισμένος κύκλος».

Μια **μετάβαση** καθορίζει την απόκριση ενός αντικειμένου που βρίσκεται σε μια κατάσταση όταν λάβει χώρα ένα γεγονός. Εν γένει, μια μετάβαση περιλαμβάνει το γεγονός που την ενεργοποιεί, προαιρετικά μια συνθήκη ελέγχου έτσι ώστε η μετάβαση να πραγματοποιείται μόνο όταν η συνθήκη είναι αληθής, μια ενέργεια και μια τελική κατάσταση (Booch G., 1999).



Εικόνα 29. Παράδειγμα διαγράμματος καταστάσεων.

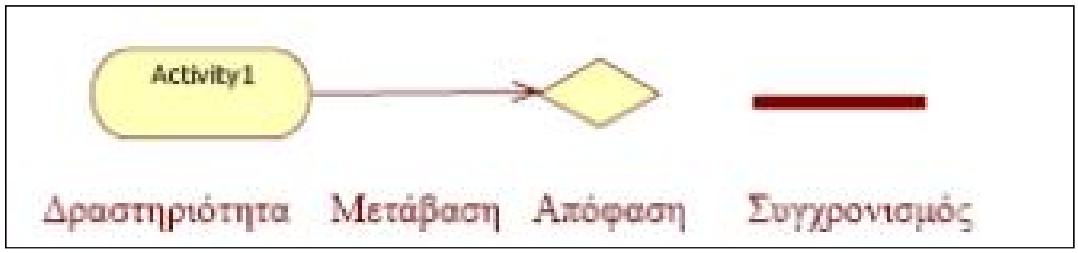
5.3.6.6 Διάγραμμα δραστηριότητας

Ένα **διάγραμμα δραστηριότητας** είναι μια ειδική μορφή **μηχανής καταστάσεων** που έχει ως στόχο τη μοντελοποίηση των υπολογισμών και της ροής της εργασίας. Στην ανάλυση απαιτήσεων απεικονίζει τις δραστηριότητες χρήστη-αλληλεπίδρασης με το σύστημα. Οι καταστάσεις του διαγράμματος δραστηριότητας αναπαριστούν τις καταστάσεις εκτέλεσης ενός υπολογισμού, όχι τις καταστάσεις των αντικειμένων που συμμετέχουν.

Υπό κανονικές συνθήκες, ένα διάγραμμα δραστηριότητας προϋποθέτει ότι οι υπολογισμοί πραγματοποιούνται χωρίς εξωτερικές διακοπές από γεγονότα, αλλιώς είναι προτιμότερο ένα διάγραμμα καταστάσεων.

Μια κατάσταση δραστηριότητας δεν αναμένει την εμφάνιση ενός γεγονότος, αλλά την ολοκλήρωση της διαδικασίας που περιγράφει, για τη μετάβαση στην επόμενη δραστηριότητα. Ένα διάγραμμα δραστηριότητας μπορεί να περιλαμβάνει διακλάδωση της δραστηριότητας σε ταυτόχρονα νήματα εκτέλεσης.

Ένα διάγραμμα δραστηριότητας περιγράφει τις συνθήκες που καθορίζουν ποιες δραστηριότητες θα εκτελεστούν σε κάθε σημείο του προγράμματος, ποιες δραστηριότητες μπορούν να λαμβάνουν χώρα παράλληλα καθώς και τυχόν επαναληπτικές δομές που περιλαμβάνονται.



Εικόνα 30. Παράδειγμα διαγράμματος δραστηριότητας.

Οι διακλαδώσεις συμβολίζονται είτε με συνθήκες «φρουρούς» επί των μεταβάσεων είτε με κόμβους απόφασης (ρόμβους) με πολλαπλές εξερχόμενες ακμές. Μια ένωση συμβολίζει συνένωση πολλών εισερχόμενων μεταβάσεων σε μία εξερχόμενη, ενώ μια διχάλα την ανάλυση μιας εισερχόμενης μετάβασης σε πολλές παράλληλες εξερχόμενες μεταβάσεις.

Τα διαγράμματα δραστηριότητας είναι χρήσιμα για την ανάλυση μιας περίπτωσης χρήσης (συνοδεύοντας την τεκμηρίωσή της) όταν πρέπει να γίνει κατανοητό ποιες ενέργειες πρέπει να πραγματοποιηθούν υπό διάφορες δυνατές συνθήκες. Επιπρόσθετα, τα διαγράμματα δραστηριότητας είναι χρήσιμα για την περιγραφή πολύπλοκων αλγορίθμων, οι οποίοι πρόκειται να υλοποιηθούν από μία ή και περισσότερες μεθόδους μιας κλάσης..

5.3.6.7 Διάγραμμα συστατικών

Ένα **συστατικό** (component) είναι μια φυσική μονάδα υλοποίησης κώδικα με σαφώς προσδιορισμένες διασυνδέσεις, η οποία αποτελεί επαναχρησιμοποιήσιμο τμήμα του συστήματος.

Σε ένα αντικειμενοστρεφές σύστημα ένα συστατικό ενσωματώνει την υλοποίηση μίας ή περισσοτέρων κλάσεων. Καλά σχεδιασμένα συστατικά δε θα πρέπει να εξαρτώνται άμεσα από άλλα συστατικά αλλά μόνο από διασυνδέσεις.

Οι εξαρτήσεις έχουν επίδραση στη συντήρηση ενός συστήματος λογισμικού. Αν κάποιο συστατικό Α εξαρτάται από κάποιο άλλο συστατικό Β, οποιαδήποτε αλλαγή στο Β μπορεί να επηρεάσει το Α. Υπό την ίδια έννοια, οι εξαρτήσεις καθορίζουν την ευκολία επαναχρησιμοποίησης ενός συστατικού. Στην περίπτωση όπου ένα συστατικό στο σύστημα μπορεί να αντικατασταθεί από κάποιο άλλο, που υποστηρίζει τις ίδιες διασυνδέσεις, δεν επιφέρονται αλλαγές στο υπόλοιπο σύστημα.

Ένα διάγραμμα συστατικών απεικονίζει το δίκτυο των εξαρτήσεων μεταξύ των συστατικών του συστήματος. Μια εξάρτηση μεταξύ δύο συστατικών υποδηλώνει ότι για την ορθή λειτουργία του ενός συστατικού απαιτείται η ύπαρξη ενός άλλου.

Το συστατικό συμβολίζεται ως ένα ορθογώνιο ενώ οι εξαρτήσεις συμβολίζονται ως διακεκομμένες ακμές με κατεύθυνση από το εξαρτώμενο συστατικό προς αυτό που παρέχει τις λειτουργίες.

Στα διαγράμματα συστατικών υπάρχει η δυνατότητα απεικόνισης λογικών τμημάτων ενός συστήματος με τη χρήση των πακέτων. Ένα πακέτο περιλαμβάνει ένα σύνολο από συστατικά τα οποία έχουν λειτουργική συνάφεια. Ο συμβολισμός ενός πακέτου στη UML επιτυγχάνεται με τη χρήση ενός φακέλου (Booch G., 1999).

5.3.6.8 Διάγραμμα ανάπτυξης

Το διάγραμμα ανάπτυξης περιγράφει την οργάνωση των επεξεργαστικών πόρων (κόμβων) του συστήματος και την αντιστοίχιση των συστατικών λογισμικού στους κόμβους αυτούς. Κατά κύριο λόγο απεικονίζουν την τοπολογία του υλικού επί του οποίου εκτελείται το σύστημα λογισμικού. Στον σχεδιασμό απεικονίζει τη φυσική απεικόνιση του λογισμικού.

Ένας **κόμβος** (node) είναι ένα φυσικό αντικείμενο που αναπαριστά έναν υπολογιστικό πόρο, ο οποίος στη γενική περίπτωση έχει τουλάχιστον μνήμη και δυνατότητα επεξεργασίας. Οι κόμβοι μπορούν να αντιστοιχίζονται σε στερεότυπα ώστε να διακρίνονται διαφορετικά είδη πόρων, όπως κεντρικές μονάδες επεξεργασίας, μνήμες, εξυπηρετητές για βάσεις δεδομένων και συσκευές διασύνδεσης με άλλα συστήματα.

Ένας κόμβος συμβολίζεται ως ένας τρισδιάστατος κύβος με το όνομα του κόμβου και ενδεχομένως ένα στερεότυπο που εκφράζει την κατηγορία στην οποία ανήκει. Σε κάθε κόμβο μπορούν προαιρετικά να αναφερθούν (υπό μορφή σημειώσεων) και τα συστατικά τα οποία εκτελούνται σε αυτόν.

Η τοπολογία του συστήματος απεικονίζεται συνδέοντας τους κόμβους με γραμμές συσχέτισης, οι οποίες μπορούν να υποδηλώνουν ρητά το πρωτόκολλο επικοινωνίας ή να χαρακτηρίζουν το σύστημα μεταφοράς δεδομένων με κάποιο τρόπο.

Το διάγραμμα ανάπτυξης δεν προσφέρει σημαντική πληροφορία για μια αυτόνομη αντικειμενοστρεφή εφαρμογή που εκτελείται αποκλειστικά σε έναν υπολογιστή.

Χρησιμοποιείται από μηχανικούς συστημάτων για τη μοντελοποίηση ενσωματωμένων συστημάτων, συστημάτων πελάτη/εξυπηρετητή, όπου υπάρχει σαφής διαχωρισμός μεταξύ των εφαρμογών που εκτελούνται στο σύστημα του πελάτη και των μονίμων δεδομένων που φιλοξενούνται στον εξυπηρετητή, καθώς και πλήρως κατανεμημένων συστημάτων που περιλαμβάνουν συνήθως πολλα απλά επίπεδα εξυπηρετητών και συνήθως φιλοξενούν πολλαπλές εκδόσεις των συστατικών λογισμικού στους κόμβους τους (Booch G., 1999).

5.3.7 Χρήση διαγραμμάτων

Τα διαγράμματα της UML επιτελούν διαφορετικούς ρόλους ανάλογα με το μοντέλο που μας ενδιαφέρει: (Booch G., 1999).

Διάγραμμα	Ιδεατό μοντέλο	Μοντέλο προδιαγραφών	Μοντέλο υλοποίησης
Διάγραμμα περιπτώσεων χρήσης	-	Αλληλεπιδράσεις με το λογισμικό	-
Διάγραμμα κλάσεων	Μοντέλα πληροφορίας	Δομές αντικειμένων	Δομές αντικειμένων
Διάγραμμα ακολουθίας ή συνεργασίας	-	Απαιτούμενες αλληλεπιδράσεις αντικειμένων	Υλοποιημένες αλληλεπιδράσεις αντικειμένων
Διάγραμμα δραστηριότητας	Επιχειρηματικές διεργασίες	-	-
Διάγραμμα καταστάσεων	Περιορισμοί αλληλουχίας γεγονότων	Περιορισμοί αλληλουχίας μηνυμάτων	Καθορισμός μηνυμάτων ή αποκρίσεων

Πίνακας 6. Μοντελοποίηση διαγραμμάτων UML, με βάση γνωστά μοντέλα.

Τα διαγράμματα περιπτώσεων χρήσης είναι παρόμοια στην εμφάνιση με αυτά του OOSE. Τα διαγράμματα κλάσεων είναι συνδυασμός αυτών του OMT, του Booch και των διαγραμμάτων κλάσεων άλλων μεθοδολογιών.

Για διάφορα διαγράμματα μπορούν να οριστούν στερεότυπα και τα αντίστοιχα εικονίδια, ώστε να υποστηρίζονται άλλα στυλ μοντελοποίησης. Τα στερεότυπα, οι περιορισμοί και οι χαρακτηρισμένες τιμές είναι χαρακτηριστικά που έχουν προστεθεί στη UML χωρίς να υπάρχουν σε άλλες μείζονες γλώσσες μοντελοποίησης.

Τα διαγράμματα καταστάσεων σε μεγάλο βαθμό βασίζονται στα διαγράμματα καταστάσεων του David Harel με μικρές αλλαγές. Τα διαγράμματα δραστηριοτήτων είναι παρόμοια με τα διαγράμματα ροής εργασίας που αναπτύχθηκαν από πολλές πηγές καθώς και από πολλές πηγές που προϋπήρχαν του αντικειμενοστρεφούς μοντέλου.

Τα διαγράμματα ακολουθίας εμφανίζονται σε πολλές αντικειμενοστρεφείς μεθοδολογίες με διάφορα ονόματα (αλληλεπίδρασης, παρακολούθησης μηνυμάτων και παρακολούθησης συμβάντων) και επίσης πηγαίνουν πίσω πριν την ύπαρξη του αντικειμενοστρεφούς μοντέλου. Τα διαγράμματα συνεργασίας προσαρμόστηκαν από τον Booch (διαγράμματα αντικειμένων), το Fusion (γράφος αλληλεπίδρασης διαγραμμάτων) και από πολλές άλλες πηγές.

Οι συνεργασίες αποτελούν πλέον οντότητες μοντελοποίησης πρώτης τάξης και συχνά αποτελούν τη βάση για τα patterns.

Τα διαγράμματα υλοποίησης (διαγράμματα συνιστωσών και deployment) προκύπτουν από τα διαγράμματα τμημάτων (module) και διεργασιών του Booch, αλλά πλέον επικεντρώνουν στις συνιστώσες, αντί στα τμήματα και είναι καλύτερα διασυνδεδεμένα..

Τα στερεότυπα είναι ένας από τους μηχανισμούς επέκτασης και επεκτείνουν τη σημασιολογία του μετα-μοντέλου. Εικονίδια που ορίζονται από το χρήστη μπορούν να χρησιμοποιηθούν με συγκεκριμένα στερεότυπα για προσαρμογή της UML σε συγκεκριμένες διαδικασίες.

Η γλώσσα περιορισμών αντικειμένων (Object Constraint Language – OCL) χρησιμοποιείται από τη UML για καθορισμό της σημασιολογίας και παρέχεται ως μία γλώσσα για εκφράσεις κατά τη μοντελοποίηση.

Η OCL είναι μία γλώσσα εκφράσεων που έχει τις ρίζες της στη μεθοδολογία Syntropy και έχει επηρεαστεί από γλώσσες εκφράσεων άλλων μεθοδολογιών όπως η Catalysis. Η μη τυπική

πλοήγηση στην OMT έχει τον ίδιο στόχο, αλλά η OCL είναι τυπική και περισσότερο εκτεταμένη.

Κεφάλαιο 6 Μελέτη περίπτωσης για ένα ηλεκτρονικό κατάστημα

Στο κεφάλαιο αυτό θα μελετήσουμε ένα ηλεκτρονικό κατάστημα, το οποίο θα ασχολείται με την πώληση ηλεκτρονικών υπολογιστών, από την ανάλυση μέχρι και τη σχεδίασή του. Θα αναλύσουμε τη μεθοδολογία που χρησιμοποιήθηκε για την ανάπτυξη αυτού του πληροφοριακού συστήματος, θα ορίσουμε τις προδιαγραφές του και θα το παρουσιάσουμε διαγραμματικά.

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα πραγματοποιηθεί η παρουσίαση των προδιαγραφών ενός πληροφοριακού συστήματος, με τη μορφή διαγραμμάτων, που θα παρουσιάζουν τις βασικές του λειτουργίες.

Αναλυτικότερα, θα εξετάσουμε την ανάλυση, τον σχεδιασμό και την υλοποίηση ενός διαδικτυακού καταστήματος, το οποίο εξειδικεύεται στην πώληση φορητών ηλεκτρονικών υπολογιστών.

Οι βασικές λειτουργίες του διαδικτυακού καταστήματος ταξινομούνται με βάση τα κυριότερα είδη χρηστών που μπορεί να αλληλεπιδράσουν με το σύστημα. Γι' αυτό τον λόγο, θα προσδιοριστούν οι προδιαγραφές του διαδικτυακού καταστήματος όσον αφορά:

- Τους Διαχειριστές του ηλεκτρονικού καταστήματος, οι οποίοι θα εποπτεύονται του πληροφοριακού αυτού συστήματος.

- Τους τελικούς χρήστες-πελάτες που θα πραγματοποιήσουν την αγορά.

Σημειώνεται ότι η εφαρμογή έχει σχεδιαστεί με τέτοιο τρόπο, ώστε στο μέλλον να μπορούν προστεθούν και νέες κατηγορίες χρηστών.

6.2 Μεθοδολογία Ανάπτυξης Ηλεκτρονικού Καταστήματος

Για την ανάπτυξη του Ηλεκτρονικού Καταστήματος ακολουθήσαμε τη μεθοδολογία ανάπτυξης Iconnix. Επιλέχθηκε ως μια ενδιάμεση λύση που βρίσκεται ανάμεσα στην Ενοποιημένη Μεθοδολογία (Unified Process) που είναι πολύ εκτενής και στην προσέγγιση του Ακραίου Προγραμματισμού (Extreme Programming) που δεν περιλαμβάνει καθόλου τη φάση της ανάλυσης και της σχεδίασης. Αποτελείται από δύο μέρη , το Δυναμικό Μοντέλο (Dynamic Model) και το Στατικό Μοντέλο (Static Model).

Το Στατικό Μοντέλο αρχικά χρησιμοποιεί μόνο διαγράμματα κλάσεων για την περιγραφή του πεδίου του προβλήματος (Domain Model) και στη συνέχεια αυτά εξελίσσονται στο Μοντέλο Κλάσεων.

Το Δυναμικό Μοντέλο χρησιμοποιεί αρχικά τα διαγράμματα περιπτώσεων χρήσης σε συνδυασμό με το ενημερωμένο διάγραμμα πεδίου προβλήματος, για να παράγει τη στατική δομή που τα υλοποιεί, δηλαδή, τα διαγράμματα Ευρωστίας (Robustness Diagram). Μετά από το συνδυασμό της αναλυτικής περιγραφής των περιπτώσεων χρήσης, των διαγραμμάτων ευρωστίας και του ενημερωμένου μοντέλου πεδίου προβλήματος, παράγονται τα διαγράμματα ακολουθίας (Sequence Diagram) και το στατικό μοντέλο κλάσεων (Class Model).

Με βάση αυτά, γίνεται η υλοποίηση με την παραγωγή του πηγαίου κώδικα και τέλος οι δοκιμές μονάδων (Unit Testing).

6.3 Αξιολόγηση Ηλεκτρονικού Καταστήματος

Πριν την υλοποίηση της εφαρμογής του Ηλεκτρονικού Καταστήματος είχαν οριστεί από τον πελάτη οι στόχοι που καλείται να εξυπηρετήσει, προκειμένου μετά την ανάπτυξη του να μπορεί να γίνει μια αποτίμηση της χρηστικότητάς του και της μελλοντικής του κερδοφορίας, Γενικά, δεν υπάρχει στη βιβλιογραφία μια συμφωνία του ορισμού της αξιολόγησης λογισμικού, οπότε επιλέξαμε έναν ορισμό που να καλύπτει πλήρως τη φιλοσοφία της αξιολόγησης που θα παρουσιάσουμε. «Ως αξιολόγηση ορίζουμε τη συλλογή, ανάλυση και ερμηνεία πληροφοριών που θα χρησιμοποιηθούν με στόχο τη διαπίστωση της αποδοτικότητας του».

Στην προσπάθειά μας, να δώσουμε στον πελάτη στοιχεία που να δείχνουν εάν κρίνεται σκόπιμη η ύπαρξη ενός Ηλεκτρονικού Καταστήματος, προχωρήσαμε στην ανάρτηση σε σελίδα κοινωνικής δικτύωσης, ενός ερωτηματολογίου για να διαπιστώσουμε κατά πόσο οι χρήστες του διαδικτύου αγοράζουν από Ηλεκτρονικό Κατάστημα. Τα αποτελέσματα είναι τα παρακάτω:

	Hits	Percent	Graph
Συχνά	24	43.6%	
Ποτέ	11	20%	
Σπάνια	10	18.2%	
Πάντα	10	18.2%	

Εικόνα 31. Δειγματοληψία

Από τη δειγματοληψία μας, ο πελάτης μας, έκρινε σκόπιμο να προχωρήσει στην απόκτηση του Ηλεκτρονικού καταστήματος, καθώς βλέπει πως οι χρήστες που ποτέ δε θα αγοράσουν προϊόντα ηλεκτρονικά είναι μόνο 20%. Το πρακτικό αποτέλεσμα του ερωτηματολογίου, είναι ότι ο πελάτης είναι πλέον σίγουρος για την μελλοντική του κερδοφορία, οπότε η επικοινωνία και οι συνεντεύξεις έχουν την απόλυτη συνεργασία του, χωρίς καμία χρονοτριβή.

6.4 Μέθοδοι Συλλογής Απαιτήσεων

Πρώτο βήμα στην ανάλυση των απαιτήσεων είναι η καταγραφή των οντοτήτων του πεδίου που διαπραγματεύεται το Ηλεκτρονικό Κατάστημα που πρόκειται να αναπτυχθεί (πεδίο προβλήματος) και των σχέσεων μεταξύ τους. Οι οντότητες αυτές αποτελούν τη βάση του στατικού αντικειμενοστρεφούς μοντέλου καθώς η λειτουργία του λογισμικού βασίζεται στην αλληλεπίδραση μεταξύ τους.

Με σκοπό να αναλύσουμε εις βάθος τις απαιτήσεις του πελάτη από το σύστημα, προχωράμε σε συνέντευξη του πελάτη και απεικονίζουμε τις ανάγκες υπό μορφή περιπτώσεων χρήσης.

Συνέντευξη: Είναι η πιο διαδεδομένη μέθοδος συλλογής πληροφοριών, αφού οι χρήστες-ενδιαφερόμενοι υποβάλλονται σε ερωτήσεις με σκοπό να αποκτήσουν γνώση του πεδίου εφαρμογής του συστήματος.

Περιγραφή των καθηκόντων των χρηστών: Καταγράφουμε τις δραστηριότητες με τις οποίες ασχολείται ο υπάλληλος, ο οποίος θα έχει τη διαχείριση του Ηλεκτρονικού Καταστήματος και θα είναι υπεύθυνος για τη σωστή λειτουργία του.

Εύρεση καλών λύσεων μέσω brainstorming: Είναι μια διαδεδομένη μέθοδος για την ανάλυση των απαιτήσεων. Σε αυτή τη μέθοδο, συγκεντρώνεται μια ομάδα ειδικών για να δημιουργήσουν πρωτότυπες και έξυπνες ιδέες για την επιτυχία του project. Υπάρχει ελευθερία έκφρασης, αποσκοπώντας στη δημιουργικότητα στο μέγιστο βαθμό.

6.5 Προσδιορισμός Προδιαγραφών συστήματος

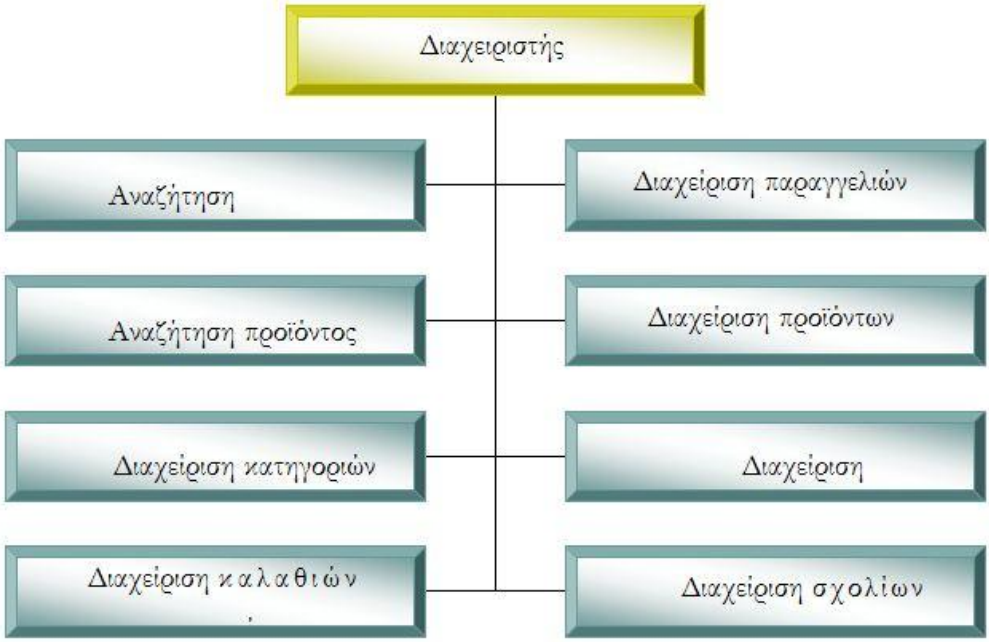
Τελικός Χρήστης-Πελάτης

Οι προδιαγραφές του συστήματος για τον τελικό χρήστη - πελάτη, απεικονίζονται στο διάγραμμα που ακολουθεί.

Εικόνα 32. Προσδιορισμός προδιαγραφών συστήματος για τον Τελικό Χρήστη- Πελάτη

Διαχειριστής Ηλεκτρονικού καταστήματος.

Οι προδιαγραφές του συστήματος για τον διαχειριστή, απεικονίζονται στο διάγραμμα που ακολουθεί.



Εικόνα 33. Προσδιορισμός προδιαγραφών συστήματος για τον Διαχειριστή του ηλεκτρονικού καταστήματος.

6.5.1 Ενδεικτική Ανάλυση Προδιαγραφών

Κάθε ενέργεια που θα εκτελεί ο χρήστης, χρησιμοποιώντας το σχεδιασμένο πληροφοριακό σύστημα, μπορεί να αναλυθεί σε επιμέρους διεργασίες. Ενδεικτικά, θα σχεδιαστεί η ενέργεια *Αναζήτηση Προϊόντος*, η οποία πραγματοποιείται σχεδόν σε όλες τις περιπτώσεις.

Ενέργεια.	**Αναζήτηση Προϊόντος.**
Περίληψη:	Μέσω της ενέργειας αυτής ο χρήστης μπορεί να αναζητήσει το προϊόν που επιθυμεί μέσω της επωνυμίας του, της τιμής του ή των χαρακτηριστικών του.
Χρήστης:	Ο Τελικός Χρήστης- Πελάτης.
Σημείο Έναρξης:	Η στιγμή που θα επιλέξει το κουμπί της ¨Αναζήτησης¨
Σημείο Λήξης:	Μόλις τερματιστεί η αναζήτηση και επιστραφούν τ' αποτελέσματα.
Αποτελέσματα:	Εμφανίζονται τα στοιχεία του προϊόντος που επιθυμεί ο χρήστης.
Ροή Γεγονότων:	Για να εκτελεστεί η λειτουργία, ο χρήστης πρέπει να συμπληρώσει μια φόρμα με συγκεκριμένα κριτήρια και να την υποβάλλει. Μόλις υποβληθεί, εμφανίζονται τα αποτελέσματα.
Εναλλακτική Ροή Γεγονότων:	Αν ο χρήστης συμπληρώσει λανθασμένα κάποιο από τα κριτήρια ή αν δεν υπάρχει το προϊόν που επιθυμεί με τα συγκεκριμένα κριτήρια που έθεσε, θα του εμφανιστεί ανάλογο προειδοποιητικό μήνυμα και θα επιστρέψει πίσω στην αναζήτηση.

Πίνακας 5. Ανάλυση προδιαγραφών Συστήματος για την ενέργεια: «Αναζήτηση Προϊόντος».

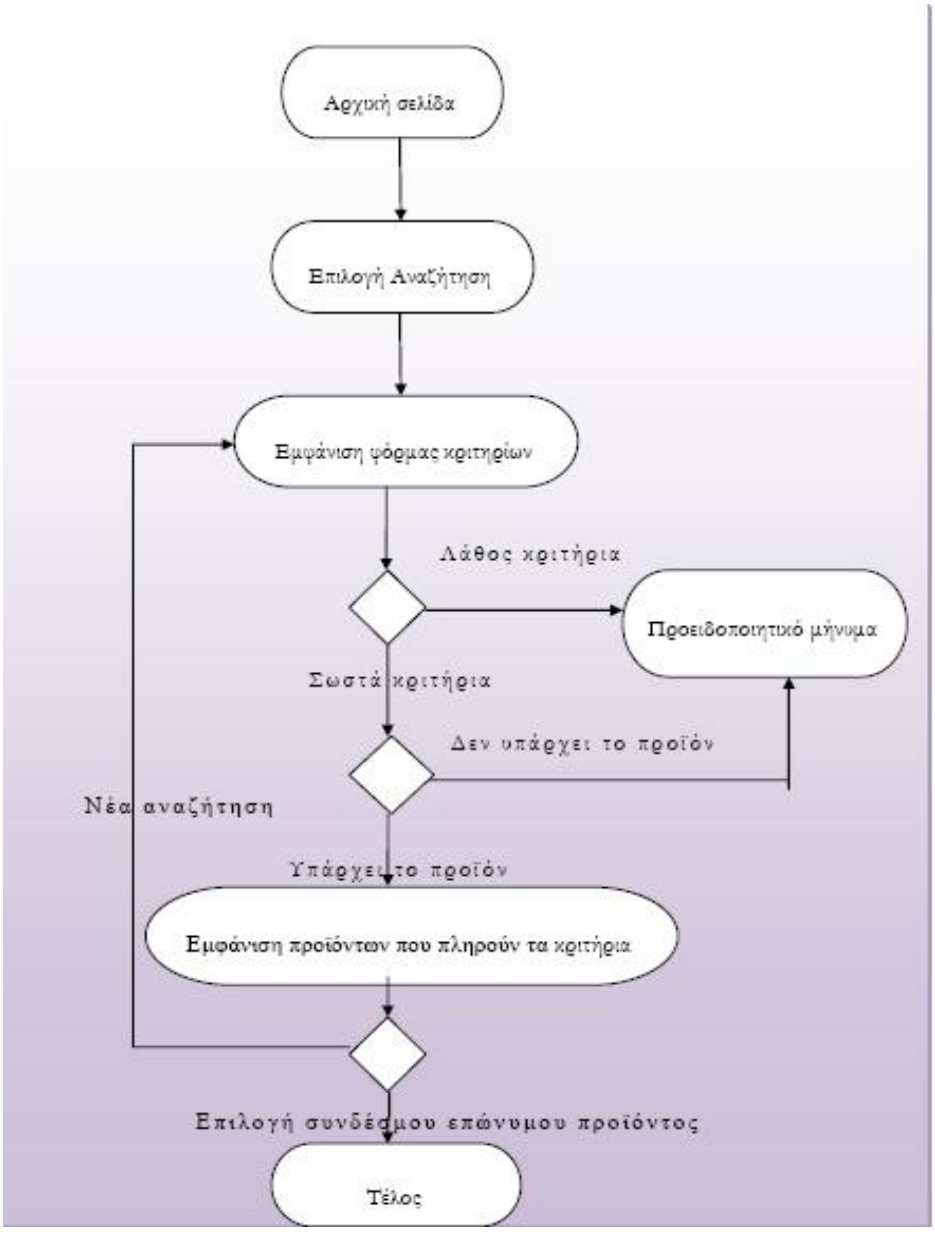
Στον παραπάνω πίνακα, παρουσιάζονται αναλυτικά όλες οι επιμέρους ενέργειες που θα πρέπει να εκτελέσει το πληροφοριακό σύστημα (στην περίπτωσή μας, το ηλεκτρονικό κατάστημα) για την ολοκλήρωση της συγκεκριμένης διαδικασίας (αναζήτηση προϊόντος).

Όπως παρατηρείται στον πίνακα, η κάθε εκτελέσιμη εργασία χαρακτηρίζεται από ένα συγκεκριμένο σημείο έναρξης και σημείο λήξης. Στην περίπτωσή μας «σημείο έναρξης» είναι η στιγμή που ο χρήστης θα «πατήσει» το κουμπί της αναζήτησης και «σημείο λήξης» είναι ο τερματισμός της αναζήτησης και ταυτόχρονα η επιστροφή των αποτελεσμάτων.

Επίσης, κάθε έναρξη μιας εργασίας σηματοδοτεί μια συγκεκριμένη ροή γεγονότων, όπως φαίνεται και στον παραπάνω πίνακα. Η ροή των γεγονότων αυτών μπορεί να αλλάζει ανάλογα με τις καταχωρήσεις του χρήστη, με απόρροια να υπάρχουν εναλλακτικές ροές γεγονότων, που πιθανών να προκύψουν από πιθανά λάθη στις καταχωρήσεις, από διαφορετικές επιλογές ή από έλλειψη αποτελεσμάτων.

Στο παράδειγμά μας, η ροή των γεγονότων πιθανών να επιφέρει εναλλακτική ροή γεγονότων, σε περίπτωση που δεν εμφανισθούν αποτελέσματα διότι το προϊόν που θέλησε ο καταναλωτής, δεν υπάρχει, θα παρουσιαστεί προειδοποιητικό μήνυμα και θα επανέλθει η φόρμα αναζήτησης στην οθόνη του χρήστη.

Τόσο οι κύριες, όσο και οι εναλλακτικές ενέργειες είναι δυνατόν να απεικονισθούν και με τη βοήθεια ενός διαγράμματος ροής, όπως απεικονίζεται παρακάτω:



Εικόνα 34. Διάγραμμα ροής ενεργειών σχετικά με «Αναζήτηση Προϊόντος».

6.6 Διαγράμματα Ευρωστίας

Η ανάλυση ευρωστίας αποτελεί μια τεχνική, ενταγμένη στη φάση της ανάλυσης απαιτήσεων, για τη μετάβαση από τις περιπτώσεις χρήσης σε ένα λεπτομερές σχέδιο. Στόχο έχει να γεφυρώσει το χάσμα μεταξύ της ανάλυσης (που απαντά στο ερώτημα "τι" θα κάνει το σύστημα) και της σχεδίασης (που απαντά στο ερώτημα "πώς" θα ικανοποιηθούν οι απαιτήσεις του πελάτη), η οποία αναφέρεται στη μεθοδολογία ICONIX, ως ο πρωταρχικός στόχος της ανάλυσης ευρωστίας.

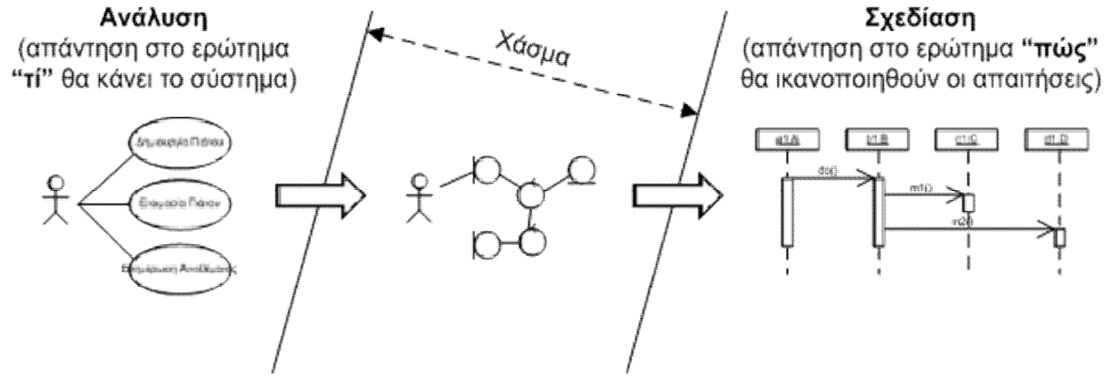
Κατά την ανάλυση ευρωστίας, το κείμενο των περιπτώσεων χρήσης μεταφράζεται σταδιακά σε μια γραφική απεικόνιση κλάσεων και συμπεριφοράς. Η γραφική αυτή απεικόνιση καλείται Διάγραμμα Ευρωστίας.

Η ανάλυση αυτή, δεν έχει κάποιο παραδοτέο που να είναι απαραίτητο ή υποχρεωτικό για τη συνέχεια σε άλλες φάσεις της διαδικασίας ανάπτυξης. Είναι όμως μια εξαιρετική τεχνική για την εκλέπτυνση και αποσαφήνιση του κειμένου των περιπτώσεων χρήσης και τον εντοπισμό ενός αρχικού συνόλου αλληλεπιδρώντων αντικειμένων (κλάσεων) για την ικανοποίηση της ζητούμενης λειτουργικότητας.

Κάθε οντότητα, η οποία σύμφωνα με το μοντέλο πεδίου προβλήματος αποτελεί μια κλάση του συστήματος, απεικονίζεται στο διάγραμμα ευρωστίας, κατηγοριοποιώντας την με βάση ένα από τα ακόλουθα τρία στερεότυπα κλάσεων:

- Συνοριακές κλάσεις, οι οποίες συνδέουν το σύστημα με τους χειριστές,
- Κλάσεις οντοτήτων, οι οποίες απεικονίζουν το μοντέλο πεδίο προβλήματος,
- Κλάσεις ελέγχου, οι οποίες ενώνουν τις συνοριακές με τις κλάσεις οντοτήτων κι απεικονίζουν τη συμπεριφορά του συστήματος.

Στο παρακάτω σχήμα φαίνεται η σημασία της ανάλυσης ευρωστίας στον κύκλο ζωής ανάλυσης λογισμικού.



Εικόνα 35. Σημασία ανάλυσης ευρωστίας στον κύκλο ζωής ανάλυσης λογισμικού.

Επομένως, τα Διαγράμματα Ευρωστίας είναι το κύριο εργαλείο στη φάση ανάλυσης-σχεδίασης. Προσφέρουν δυνατότητα διερεύνησης των ενεργειών που υπαγορεύονται από τις περιπτώσεις χρήσης και βελτίωση του κειμένου των ίδιων των περιπτώσεων χρήσης. Το αποτέλεσμα είναι ένα αναθεωρημένο και εμπλουτισμένο διάγραμμα κλάσεων, με επιπρόσθετες κλάσεις και ιδιότητες κλάσεων.

Στα διαγράμματα ευρωστίας χρησιμοποιούνται:

- Οι Χρήστες (actors)
- Τα συνοριακά αντικείμενα (boundary)
- Οι οντότητες (entities)
- Τα controls
- Τα usecases
- Τα βέλη (connectors), τα οποία ενώνουν τα παραπάνω σχήματα.

Οι κανόνες επικοινωνίας ή αλλιώς σχεδίασης είναι οι εξής:

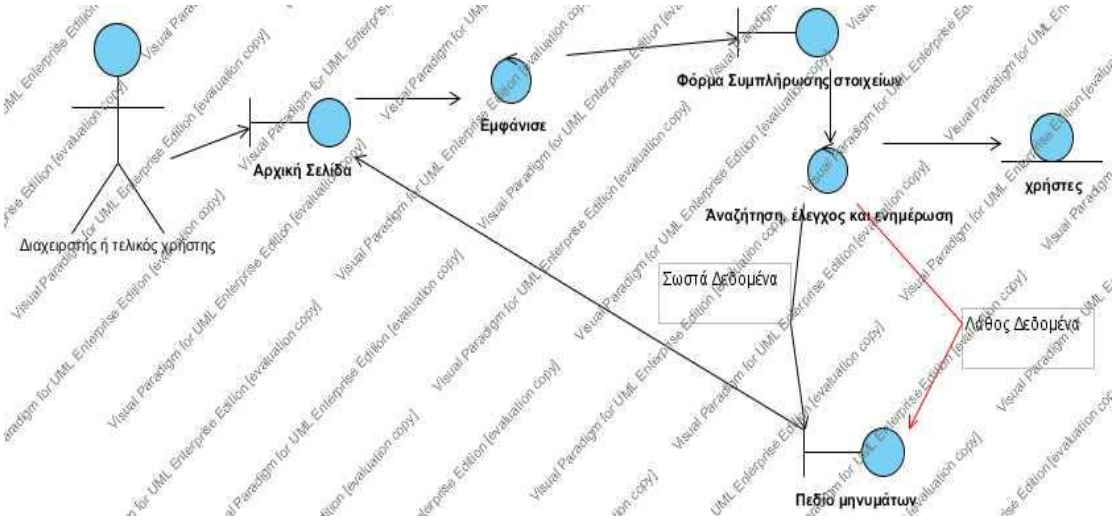
- Οι χειριστές επικοινωνούν μόνο με τα συνοριακά αντικείμενα

- Για να επικοινωνήσουν δύο συνοριακά αντικείμενα απαιτείται ένα control.
- Το control επικοινωνεί με boundary και entity. Δεν επικοινωνεί ποτέ με actor!

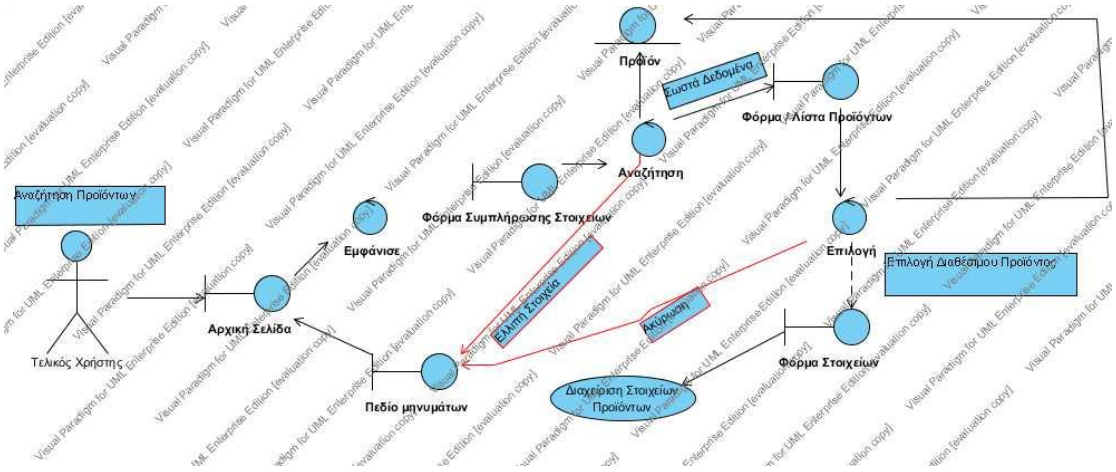
Ουσιαστικά, οι κανόνες επικοινωνίας επιβάλλουν την οργάνωση του κειμένου των περιπτώσεων χρήσης, κατά τέτοιο τρόπο, ώστε να είναι δυνατή στη συνέχεια η παραγωγή λεπτομερούς σχεδίου υπό μορφή διαγραμμάτων ακολουθίας.

Παρακάτω παρουσιάζονται μερικά διαγράμματα ευρωστίας που αφορούν την Είσοδο στο σύστημα από έναν διαχειριστή ή τελικό πελάτη-χρήστη, την Αναζήτηση Προϊόντος και την Καταχώρηση Προϊόντος.

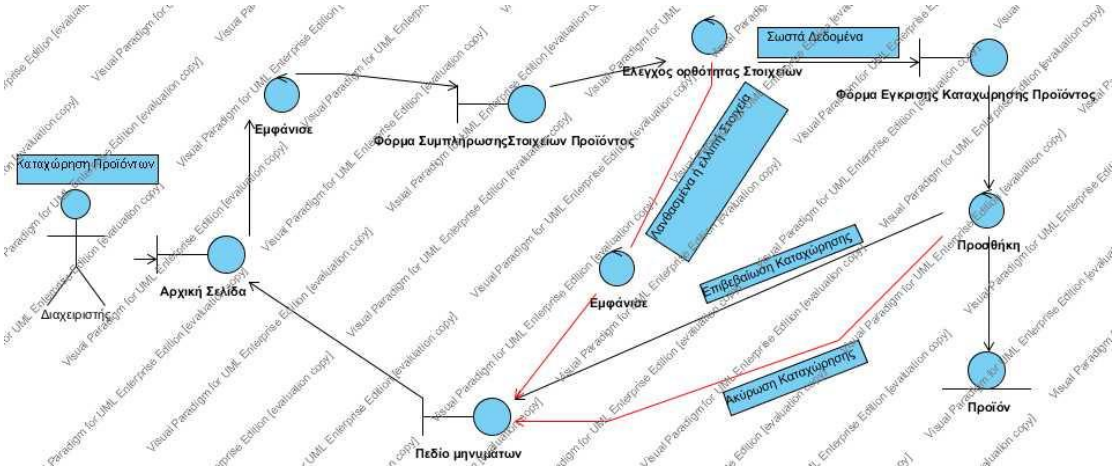
6.6.1 Ενδεικτικά Διαγράμματα ευρωστίας



Εικόνα 36. Διάγραμμα Ευρωστίας Σύνδεσης στο σύστημα.



Εικόνα 37. Διάγραμμα Ευρωστίας: Αναζήτησης Προϊόντος.



Εικόνα 38. Διάγραμμα Ευρωστίας Καταχώρησης προϊόντων.

6.7 Διαγράμματα Ακολουθίας

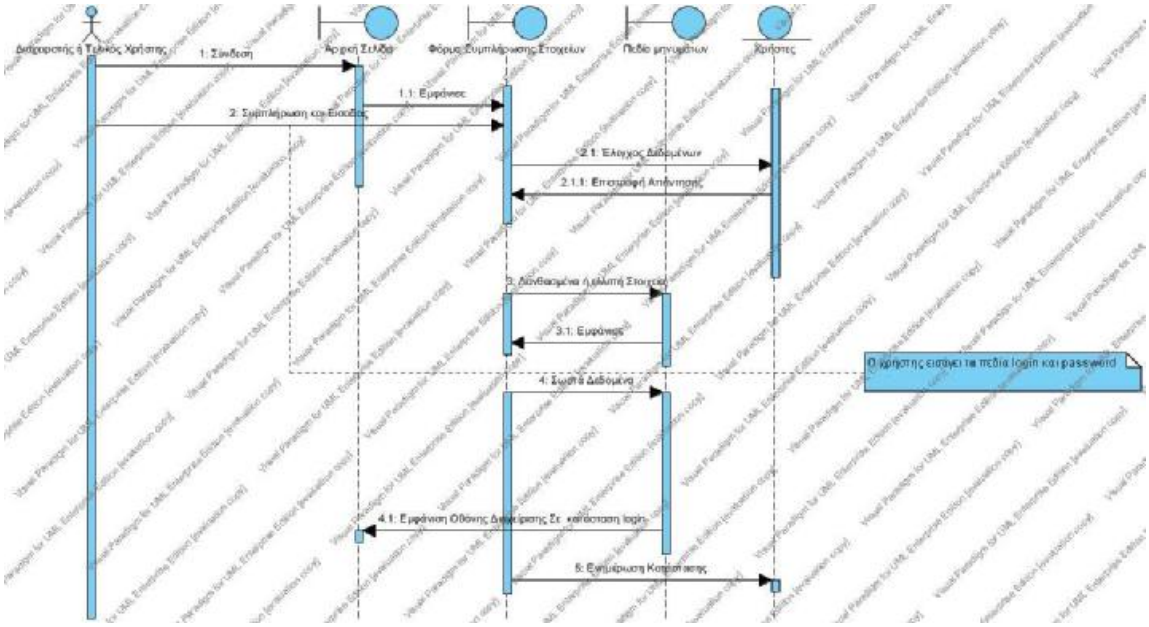
Με την ολοκλήρωση των σταδίων της ανάλυσης, οι προδιαγραφές του συστήματος, μπορούν να θεωρηθούν πλήρεις, ορθές, λεπτομερείς και σαφείς. Επιπλέον, έχουν εντοπιστεί οι περισσότερες κλάσεις, δηλαδή ολοκληρώθηκε το μεγαλύτερο τμήμα της στατικής δομής του λογισμικού.

Παρ’ όλα αυτά, λίγες ενέργειες στη φάση ανάλυσης είχαν ως στόχο την κατανομή της συμπεριφοράς στις κλάσεις του συστήματος. Ο ακριβής καθορισμός του τρόπου με τον οποίο οι κλάσεις αλληλεπιδρούν μεταξύ τους, αποτελούν αντικείμενο της σχεδίασης.

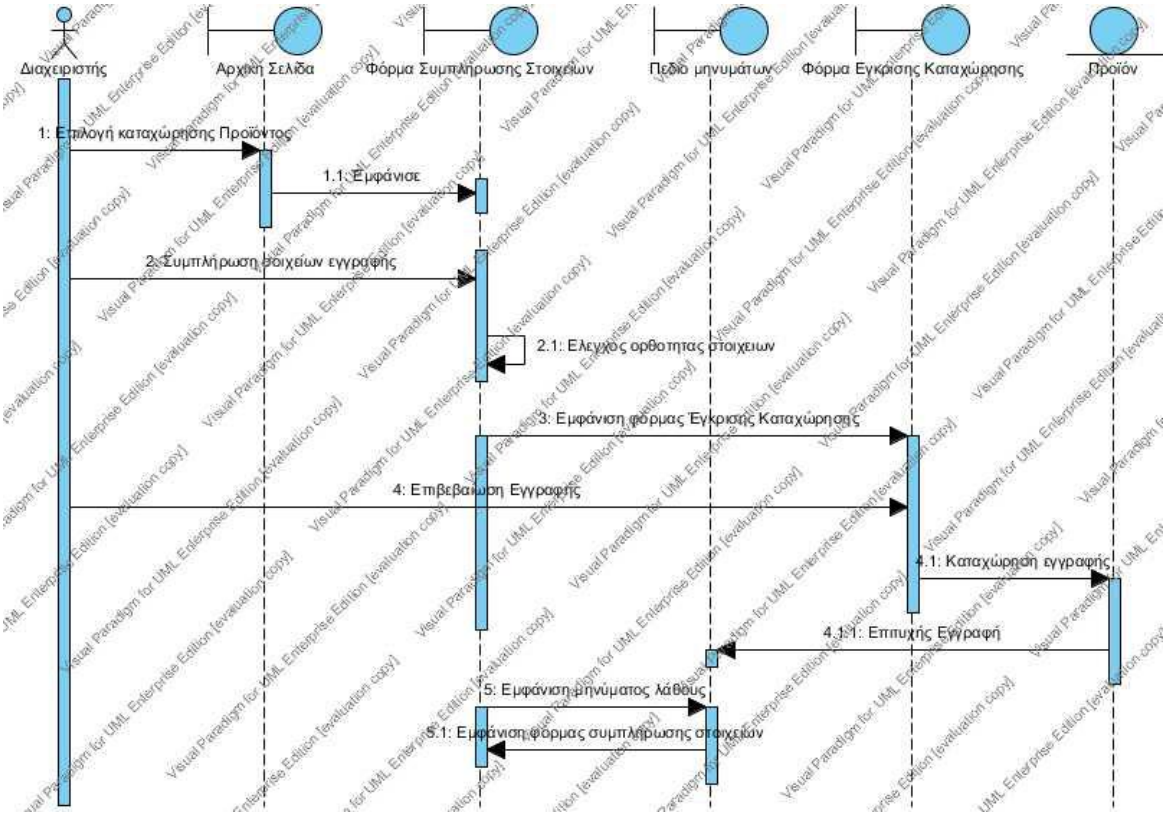
Έτσι, προκύπτει το αναθεωρημένο διάγραμμα κλάσεων, όπου εκτός από τις κλάσεις με τις ιδιότητές τους, απεικονίζονται και οι μέθοδοι με τις τυχόν νέες σχέσεις μεταξύ τους. Το τελικό αυτό διάγραμμα κλάσεων αποτελεί την είσοδο για την έναρξη της κωδικοποίησης του συστήματος (αν και είναι αναμενόμενο το διάγραμμα κλάσεων να τροποποιηθεί, ως ένα βαθμό, κατά την υλοποίηση).

Η κατανομή της λειτουργικότητας στις κλάσεις, επιτυγχάνεται με τα διαγράμματα ακολουθίας, τα οποία απεικονίζουν τα μηνύματα που ανταλλάσουν τα αντικείμενα του συστήματος μεταξύ τους για την ικανοποίηση της λειτουργικότητας ενός σεναρίου χρήσης.

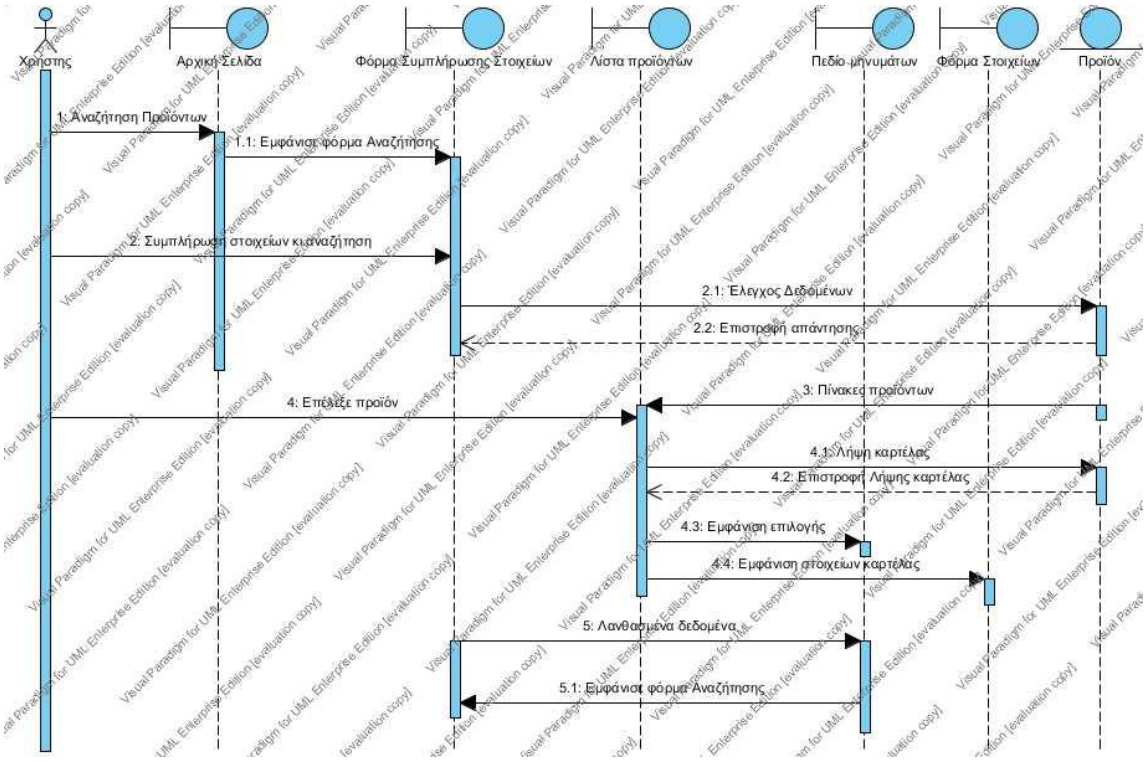
6.7.1 Ενδεικτικά διαγράμματα ακολουθίας



Εικόνα 39. Sequence Diagram -Σύνδεση Στο σύστημα.



Εικόνα 40. Sequence Diagram -Καταχώρηση Προϊόντων



Εικόνα 41. Sequence Diagram -Αναζήτηση Προϊόντος

6.8 Εργαλεία υλοποίησης Διαγραμμάτων UML

6.8.1 Visual Paradigm for UML 7.1

Το Visual Paradigm είναι ένα εργαλείο CASE (Computer-Aided Software Engineering), που παρέχει εκτεταμένη μοντελοποίηση για διαγράμματα UML, διαγράμματα περιπτώσεων χρήσης (use case diagrams) και διαγράμματα οντοτήτων - συσχετίσεων.

Τα στοιχεία του Visual Paradigm που του προσφέρουν ανταγωνιστικό πλεονέκτημα είναι:

Η ευκολία στη χρήση, η αλληλεπίδρασή του με μερικά από τα πιο γνωστά IDE όπως το Visual Studio και το Eclipse και η πληθώρα προηγμένων λειτουργιών που υποστηρίζει μεταξύ των οποίων συμπεριλαμβάνονται οι εξής: Παραγωγή κώδικα από διαγράμματα κλάσεων, Business Process Modeling και Reverse Engineering για κώδικα που έχει γραφτεί σε πάνω από 10 γλώσσες προγραμματισμού.

6.9 Εφαρμογή του Case study

Η εφαρμογή εστιάζει στην ανάλυση, στον σχεδιασμό και στην ανάπτυξη ενός διαδικτυακού καταστήματος που εξειδικεύεται στην πώληση φορητών ηλεκτρονικών υπολογιστών. Το σύστημα θα παρέχει ένα εύχρηστο και φιλικό περιβάλλον μέσα από το οποίο ο χρήστης θα μπορεί, μεταξύ άλλων, να πραγματοποιήσει ηλεκτρονικές αγορές. Ιδιαίτερη σημασία έχει δοθεί στην ανάπτυξη ενός εύχρηστου περιβάλλοντος διαχείρισης, μέσω του οποίου ο διαχειριστής θα μπορεί να επιβλέπει τις παραγγελίες και να διαχειρίζεται τη λειτουργικότητα του ηλεκτρονικού καταστήματος.

6.9.1 Περιγραφή συστήματος

Η ανάπτυξη της εφαρμογής βασίστηκε σε προηγμένα εργαλεία, τα οποία συνθέτουν ένα τεχνολογικό περιβάλλον βασισμένο στον παγκόσμιο ιστό, μέσω του οποίου μπορούν να γίνονται, μεταξύ άλλων, οι ακόλουθες ενέργειες:

- Ηλεκτρονικές αγορές: Ο χρήστης, αφού τοποθετήσει στο καλάθι αγορών τα προϊόντα που επιθυμεί να αγοράσει, κατευθύνεται στο ηλεκτρονικό σύστημα πληρωμών Paypal όπου μπορεί να ολοκληρώσει με ασφάλεια την ηλεκτρονική του αγορά.

- Πλοήγηση στο ηλεκτρονικό κατάστημα: Στον χρήστη παρέχεται λεπτομερή περιγραφή για κάθε προϊόν του ηλεκτρονικού καταστήματος καθώς και η δυνατότητα να αναρτήσει σχόλιο για κάποιο προϊόν και να δει τα είδη υπάρχοντα. Επίσης, μπορεί να πραγματοποιήσει αναζήτηση για κάποιο προϊόν και να επικοινωνήσει με τον διαχειριστή συμπληρώνοντας τη κατάλληλη φόρμα.

- Διαχείριση παραγγελιών: Ο διαχειριστής, έχει τη δυνατότητα να διαχειρίζεται τις παραγγελιές που έχουν πραγματοποιηθεί. Μεταξύ άλλων, μπορεί να αλλάξει το status της παραγγελίας, να προσθέσει κάποιο σχόλιο για την παραγγελία καθώς και να τροποποιήσει τα στοιχεία του κατόχου της παραγγελίας, εάν αυτό ζητηθεί από τον πελάτη. Επίσης μπορεί να διαγράψει μια παραγγελία αν το κρίνει απαραίτητο.

- Διαχείριση λειτουργικότητας ηλεκτρονικού καταστήματος: Ο διαχειριστής, έχει τη δυνατότητα να εμπλουτίσει τον κατάλογο του ηλεκτρονικού καταστήματος προσθέτοντας νέα προϊόντα. Ακόμα μπορεί να προσθέσει μια νέα κατηγορία ή έναν κατασκευαστή προϊόντων, καθώς και να καθορίσει τη διάταξη των αντίστοιχων μενού.

6.9.2 Καταγραφή απαιτήσεων εφαρμογής

Στην ενότητα αυτή παρουσιάζονται οι λειτουργικές απαιτήσεις του συστήματος και οι περιορισμοί κάτω από τους οποίους αυτό θα λειτουργεί.

Κωδικό όνομα απαίτησης
R1
Προσδιορισμός/ Τίτλος Απαίτησης
Διαχείριση κατηγοριών προϊόντων (Προσθήκη, τροποποίηση, διαγραφή)
Περιγραφή αίτησης
Ο Διαχειριστής μπορεί να επιλέξει να προσθέσει μια κατηγορία προϊόντων. Για την προσθήκη μιας κατηγορίας προϊόντων είναι απαραίτητα τα εξής στοιχεία: Όνομα κατηγορίας, περιγραφή κατηγορίας. Ο διαχειριστής μπορεί να επιλέξει να διαγράψει μια κατηγορία προϊόντων. Κάνοντας κλικ στο κουμπί «Διαγραφή», θα εμφανίζεται ένα αναδυόμενο παράθυρο το οποίο θα ρωτάει τον χρήστη να επιβεβαιώσει ή να ακυρώσει την διαγραφή. Διαγράφοντας μια κατηγορία θα διαγράφονται αυτόματα και τα προϊόντα που σχετίζονται με αυτή την κατηγορία.

Πίνακας 6. Διαχείριση κατηγοριών προϊόντος

Κωδικό όνομα απαίτησης
R2
Προσδιορισμός/ Τίτλος Απαίτησης
Διαχείριση κατασκευαστών προϊόντων (Προσθήκη, τροποποίηση, διαγραφή)
Περιγραφή Απαίτησης
Ο διαχειριστής μπορεί να επιλέξει να προσθέσει έναν κατασκευαστή προϊόντων. Ο διαχειριστής μπορεί να επιλέξει να διαγράψει έναν κατασκευαστή προϊόντων. Κάνοντας κλικ στο κουμπί «Διαγραφή» θα εμφανίζεται ένα αναδυόμενο παράθυρο, το οποίο Θα ρωτάει τον χρήστη να επιβεβαιώσει ή να ακυρώσει την διαγραφή. Διαγράφοντας έναν κατασκευαστή θα διαγράφονται αυτόματα και τα προϊόντα που σχετίζονται με αυτόν τον κατασκευαστή.

Πίνακας 7. Διαχείριση κατασκευαστών προϊόντος

Κωδικό όνομα απαίτησης
R3
Προσδιορισμός/ Τίτλος Απαίτησης
Καταμέτρηση και διαγραφή ανενεργών καλαθιών χρήστη
Περιγραφή Απαίτησης
Ο διαχειριστής μπορεί να δει τον αριθμό των καλαθιών του χρήστη του συστήματος που έχουν μείνει ανενεργά για κάποιο αριθμό ημερών που ορίζει ο ίδιος. Ο διαχειριστής μπορεί να διαγράψει τα καλάθια χρήστη που έχουν μείνει ανενεργά για κάποιο αριθμό ημερών που ορίζει ο ίδιος, ώστε να μη δεσμεύεται άσκοπος χώρος στη βάση δεδομένων. «Ανενεργό καλάθι χρήστη» ορίζεται το καλάθι χρήστη, του οποίου η πιο πρόσφατη τροποποίηση ή προσθήκη προϊόντος έγινε πριν από έναν συγκεκριμένο αριθμό ημερών, που ορίζεται από τον διαχειριστή

Πίνακας 8. Καταμέτρηση και διαγραφή καλαθιών χρήστη.

Κωδικό όνομα απαίτησης
R4
Προσδιορισμός/ Τίτλος Απαίτησης
Διαχείριση προϊόντων (προσθήκη, τροποποίηση, διαγραφή)
Περιγραφή Απαίτησης
Ο διαχειριστής μπορεί να επιλέξει την προσθήκη νέου προϊόντος. Τα απαραίτητα στοιχεία για την εισαγωγή προϊόντος είναι τα εξής: Τεχνικά χαρακτηριστικά προϊόντος (σκληρός δίσκος, οθόνη, κλπ.), όνομα, περιγραφή, τιμή, κατηγορία, κατασκευαστής, εγγύηση, επιλογή εμφάνισης στην αρχική σελίδα της εφαρμογής. Ο διαχειριστής μπορεί να επιλέξει μέχρι δύο φωτογραφίες για ένα προϊόν. Ο διαχειριστής μπορεί να τροποποιήσει τα στοιχεία ενός προϊόντος. Ο διαχειριστής μπορεί να επιλέξει τη διαγραφή ενός προϊόντος. Κάνοντας κλικ στο κουμπί «Delete» θα εμφανίζεται ένα αναδυόμενο παράθυρο, το οποίο θα ρωτάει τον χρήστη για την επιβεβαίωση ή ακύρωση της διαγραφής. Διαγράφοντας ένα προϊόν θα διαγράφεται αυτόματα και από τα καλάθια χρηστών, στα οποία έχει προστεθεί.

Πίνακας 9. Διαχείριση προϊόντων.

Κωδικό όνομα απαίτησης
R5
Προσδιορισμός/ Τίτλος Απαίτησης
Προβολή και διαχείριση σχολίων χρήστη (διαγραφή, τροποποίηση)
Περιγραφή Απαίτησης
Ο διαχειριστής μπορεί να βλέπει τα σχόλια που έχουν γίνει για κάποιο προϊόν. Τα στοιχεία ενός σχολίου που είναι διαθέσιμα προς τον διαχειριστή είναι τα εξής: Κωδικός σχολίου, όνομα χρήστη, ημερομηνία υποβολής, περιεχόμενο σχολίου, e-mail χρήστη, IP χρήστη. Ο διαχειριστής μπορεί εάν το κρίνει απαραίτητο να τροποποιήσει το περιεχόμενο κάποιου σχολίου.

Πίνακας 10. Προβολή και διαχείριση σχολίων χρήστη.

Κωδικό όνομα απαίτησης
R6
Προσδιορισμός/ Τίτλος Απαίτησης
Καταχώρηση παραγγελίας πελάτη
Αρχικά ζητείται από τον πελάτη να συμπληρώσει τα στοιχεία του. Τα απαραίτητα προς συμπλήρωση στοιχεία είναι τα εξής: ονοματεπώνυμο, διεύθυνση αποστολής, πόλη, Τ.Κ., χώρα, τηλέφωνο, ηλεκτρονική διεύθυνση. Τα στοιχεία θα ελέγχονται για την εγκυρότητά τους και θα εμφανίζεται κατάλληλο μήνυμα σε περίπτωση σφάλματος. Θα αποθηκεύνται τα στοιχεία του πελάτη και τα στοιχεία της παραγγελίας που θα πραγματοποιήσει. Τα στοιχεία της παραγγελίας που θα αποθηκεύονται σε αυτό το στάδιο είναι τα εξής: Κωδικός παραγγελίας, ημερομηνία δημιουργίας, τα προϊόντα που περιλαμβάνει κάθε παραγγελία, η ποσότητα κάθε προϊόντος, το κόστος μονάδας, το συνολικό κόστος κάθε προϊόντος και το συνολικό κόστος της παραγγελίας. Στη συνέχεια ο χρήστης θα κατευθύνεται στο ηλεκτρονικό σύστημα πληρωμών PayPal, όπου θα μπορεί να ολοκληρώσει την παραγγελία του. Η εφαρμογή υποστηρίζει αυτοματοποιημένη διαδικασία αποστολής μηνύματος στον διαχειριστή από το PayPal για κάθε παραγγελία που εξοφλείται. Η εφαρμογή υποστηρίζει αυτοματοποιημένη διαδικασία αποστολής ηλεκτρονικού μηνύματος στον πελάτη μετά την εξόφληση της παραγγελίας του. Το ηλεκτρονικό μήνυμα που αποστέλλεται στον πελάτη περιλαμβάνει τα εξής στοιχεία: Τον κωδικό της παραγγελίας, τα προϊόντα που περιλαμβάνει η παραγγελία, τα στοιχεία του χρήστη, τον εκτιμώμενο χρόνο παράδοσης.

Πίνακας 11. Καταχώρηση παραγγελίας πελάτη.

Κωδικό όνομα απαίτησης
R7
Προσδιορισμός/ Τίτλος Απαίτησης
Αναζήτηση και διαχείριση παραγγελιών πελάτη (επεξεργασία, διαγραφή)
Περιγραφή Απαίτησης
Τα κριτήρια αναζήτησης παραγγελιών που είναι διαθέσιμα στον διαχειριστή είναι τα εξής: Εμφάνιση των Ν τελευταίων παραγγελιών (όπου Ν είναι ακέραιος αριθμός που δίνεται ως είσοδος από τον χρήστη), εμφάνιση των παραγγελιών που δημιουργήθηκαν μεταξύ δύο ημερομηνιών που δίνονται από τον χρήστη, εμφάνιση των παραγγελιών που δεν έχουν

Κωδικό όνομα απαίτησης
εξοφληθεί, εμφάνιση των παραγγελιών που έχουν ακυρωθεί. Για κάθε αναζήτηση που περιλαμβάνει είσοδο στοιχείων από τον χρήστη, θα γίνεται έλεγχος της εγκυρότητας των στοιχείων αυτών. Κάθε παραγγελία παρέχει συνοπτικές πληροφορίες σχετικά με τον κωδικό της παραγγελίας, την ημερομηνία δημιουργίας, το ονοματεπώνυμο του πελάτη, καθώς και το στάδιο εξέλιξης στο οποίο βρίσκεται. Επιλέγοντας μια παραγγελία, παρέχονται αναλυτικότερες πληροφορίες, όπως είναι η διεύθυνση αποστολής της παραγγελίας, το συνολικό ποσόν πληρωμής, τα πλήρη στοιχεία του πελάτη, καθώς και τα πλήρη στοιχεία των προϊόντων που περιλαμβάνει η παραγγελία. Ο διαχειριστής μπορεί να επεξεργαστεί τα στοιχεία μιας παραγγελίας, όπως την προσθήκη κάποιου σχολίου που αφορά την παραγγελία ή να τροποποιήσει τα στοιχεία του πελάτη αν το κρίνει απαραίτητο. Ο διαχειριστής μπορεί να αλλάζει το status μιας παραγγελίας. Μια παραγγελία μπορεί να οριστεί ως: α) εξοφλημένη, β) ολοκληρωμένη, γ) ακυρωθείσα. Μια παραγγελία που καταχωρείται ως ολοκληρωμένη ενεργοποιεί την αυτόματη διαδικασία αποστολής ηλεκτρονικού μηνύματος στον κάτοχο της παραγγελίας (πελάτη), τον οποίο ενημερώνει ότι η διαδικασία αποστολής των προϊόντων που έχει αγοράσει έχει ξεκινήσει. Το ηλεκτρονικό μήνυμα που αποστέλλεται στον πελάτη περιλαμβάνει τα εξής στοιχεία: Ενημέρωση ότι η διαδικασία αποστολής των προϊόντων που έχει αγοράσει έχει ξεκινήσει, τον κωδικό της παραγγελίας, τα προϊόντα που περιλαμβάνει η παραγγελία, το συνολικό κόστος, τα στοιχεία του πελάτη, τον εκτιμώμενο χρόνο παράδοσης. Ο διαχειριστής μπορεί να διαγράψει μια παραγγελία.

Πίνακας 12. Αναζήτηση και διαχείριση προϊόντων πελάτη.

Κωδικό όνομα απαίτησης
R8
Προσδιορισμός/ Τίτλος Απαίτησης
Οικονομική εκκάθαρση (πληρωμή προϊόντων)
Περιγραφή απαίτησης
Ο χρήστης αφού τοποθετήσει στο καλάθι αγορών τα προϊόντα που σκοπεύει να αγοράσει, συμπληρώνει τη φόρμα υποβολής παραγγελίας.

Κωδικό όνομα απαίτησης
Τα πεδία της φόρμας υποβολής παραγγελίας είναι τα εξής: Όνομα, επίθετο, διεύθυνση αποστολής, πόλη, Τ.Κ., χώρα, τηλέφωνο, ηλεκτρονική διεύθυνση. Τα στοιχεία θα ελέγχονται για την εγκυρότητά τους και θα εμφανίζεται κατάλληλο μήνυμα σε περίπτωση σφάλματος. Ο χρήστης θα έχει την επιλογή ακύρωσης της παραγγελίας και σε αυτή την περίπτωση θα κατευθύνεται στο ηλεκτρονικό σύστημα πληρωμών PayPal, όπου θα μπορεί να ολοκληρώσει την παραγγελία του. Τα πεδία της φόρμας θα ελέγχονται ως προς την εγκυρότητά τους και θα εμφανίζεται κατάλληλο μήνυμα σε περίπτωση λάθους. Στο PayPal, ο χρήστης θα μπορεί να δημιουργήσει έναν λογαριασμό ή να συνδεθεί στον ήδη υπάρχοντα, ώστε να εξοφλήσει την παραγγελία.

Πίνακας 13. Οικονομική εκκάθαρση.

Κωδικό όνομα απαίτησης
R9
Προσδιορισμός/Τίτλος απαίτησης
Αποστολή ηλεκτρονικού μηνύματος στον ιδιοκτήτη/ διαχειριστή της εφαρμογής
Περιγραφή Απαίτησης
Ο χρήστης μπορεί να υποβάλλει ένα ερώτημα ή ένα σχόλιο στον ιδιοκτήτη/ διαχειριστή της εφαρμογής συμπληρώνοντας τη φόρμα επικοινωνίας. Τα πεδία της φόρμας επικοινωνίας είναι τα εξής: Ονοματεπώνυμο, ηλεκτρονική διεύθυνση, τηλέφωνο, κείμενο μηνύματος. Πριν την αποστολή του μηνύματος θα ελέγχονται τα πεδία της φόρμας ως προς την εγκυρότητά τους και θα εμφανίζεται κατάλληλο μήνυμα σε περίπτωση λάθους.

Πίνακας 14. Αποστολή e-mail στον διαχειριστή.

Κωδικό όνομα απαίτησης
R10
Προσδιορισμός/ Τίτλος απαίτησης
Υποβολή σχολίου
Περιγραφή Απαίτησης

Κωδικό όνομα απαίτησης
Ο χρήστης μπορεί να υποβάλλει το σχόλιό του για κάποιο προϊόν συμπληρώνοντας τη φόρμα υποβολής σχολίου. Για την υποβολή σχολίου είναι απαραίτητα τα εξής στοιχεία: Όνομα χρήστη, διεύθυνση ηλεκτρονικού ταχυδρομείου, κείμενο σχολίου. Πριν την ανάρτηση του σχολίου είναι απαραίτητα τα εξής στοιχεία: Όνομα χρήστη, διεύθυνση ηλεκτρονικού ταχυδρομείου, κείμενο σχολίου. Πριν την ανάρτηση του σχολίου θα ελέγχονται τα πεδία της φόρμας ως προς την εγκυρότητά τους και θα εμφανίζεται κατάλληλο μήνυμα σε περίπτωση λάθους.

Πίνακας 15. Υποβολή σχολίου.

Κωδικό όνομα απαίτησης
R11
Προσδιορισμός/ Τίτλος Απαίτησης
Διαχείριση καλαθιού αγορών (προσθήκη προϊόντος, διαγραφή προϊόντος, ανανέωση ποσότητας προϊόντος)
Περιγραφή Απαίτησης
Ο χρήστης μπορεί κάθε στιγμή, να δει τα προϊόντα που έχει τοποθετήσει στο καλάθι αγορών, καθώς και το συνολικό ποσό χρέωσης των προϊόντων αυτών. Ο χρήστης μπορεί να προσθέσει ένα προϊόν στο καλάθι αγορών. Ο χρήστης μπορεί να αφαιρέσει ένα προϊόν από το καλάθι αγορών. Ο χρήστης μπορεί να ανανεώσει την ποσότητα κάποιου προϊόντος που υπάρχει ήδη στο καλάθι αγορών. Αν η ποσότητα κάποιου προϊόντος οριστεί ως μηδέν, τότε το προϊόν αυτό θα αφαιρείται από το καλάθι αγορών. Τα περιεχόμενα του καλαθιού αγορών κάθε χρήστη δεν θα χάνονται μετά την έξοδό του από την εφαρμογή, ώστε την επόμενη φορά που θα εισέλθει να είναι διαθέσιμα σε αυτόν. Το χρονικό διάστημα που θα παραμείνει ενεργό το καλάθι αγορών κάποιου χρήστη ορίζεται από τον διαχειριστή εφαρμογής.

Πίνακας 16. Διαχείρηση Καλαθιού αγορών.

Κωδικό όνομα απαίτησης

R12
Προσδιορισμός/ Τίτλος Απαίτησης
Αναζήτηση προϊόντος
Περιγραφή Απαίτησης
Ο χρήστης μπορεί να πραγματοποιήσει αναζήτηση προϊόντος χρησιμοποιώντας το search texbox. Σε περίπτωση που τα αποτελέσματα της αναζήτησης ξεπερνούν τα έξι, τότε θα χωρίζονται σε σελίδες και ο χρήστης θα μπορεί να πλοηγείται μεταξύ των σελίδων, με τη βοήθεια συνδέσμων (links). Ο διαχειριστής μπορεί να καθορίσει τον αριθμό των αποτελεσμάτων που θα εμφανίζονται ανά σελίδα.

Πίνακας 17. Αναζήτηση προϊόντος

.**Κωδικό όνομα απαίτησης**
R13
Προσδιορισμός/ Τίτλος Απαίτησης
Προβολή λεπτομερειών προϊόντος
Περιγραφή Απαίτησης
Ο χρήστης επιλέγοντας κάποιο προϊόν από τον κατάλογο του ηλεκτρονικού καταστήματος, θα μπορεί να βλέπει όλα τα χαρακτηριστικά του προϊόντος (όνομα, τιμή, περιγραφή, κατασκευαστής, εγγύηση, τεχνικά χαρακτηριστικά), καθώς και μια φωτογραφία μεγαλύτερων διαστάσεων.

Πίνακας 18. Προβολή λεπτομερειών προϊόντος.

6.9.3 Περιπτώσεις χρήσης και σενάρια

Σε αυτή την ενότητα θα αναλυθούν όλες οι περιπτώσεις χρήσης ώστε να προσδιοριστούν τα σενάρια χρήσης της εφαρμογής και να καταγραφούν όλες οι περιπτώσεις αλληλεπίδρασης των χρηστών με το σύστημα.

Για την περίπτωση χρήσης "Διαχείριση προϊόντων" υφίστανται τα εξής σενάρια:

- Προσθήκη προϊόντος.
- Επεξεργασία προϊόντος.
- Διαγραφή προϊόντος.
- Προσθήκη κατηγορίας ή κατασκευαστή.
- Επεξεργασία κατηγορίας ή κατασκευαστή.
- Διαγραφή κατηγορίας ή κατασκευαστή.
- Τροποποίηση ή διαγραφή σχολίου χρήστη.
- Αναζήτηση προϊόντος.
- Προβολή λεπτομερειών προϊόντος.
- Υποβολή σχολίου για κάποιο προϊόν.

Για την περίπτωση χρήσης "Διαχείριση παραγγελιών" υφίστανται τα εξής σενάρια:

- Επεξεργασία στοιχείων παραγγελίας.
- Διαγραφή παραγγελίας.
- Αναζήτηση παραγγελιών βάση κριτηρίων.

- Διαχείριση καλαθιού αγορών (προσθήκη προϊόντος, διαγραφή προϊόντος, ανανέωση ποσότητας προϊόντος).

- Οικονομική εκκαθάριση (πληρωμή προϊόντων).

- Καταμέτρηση και διαγραφή ανενεργών καλαθιών αγορών.

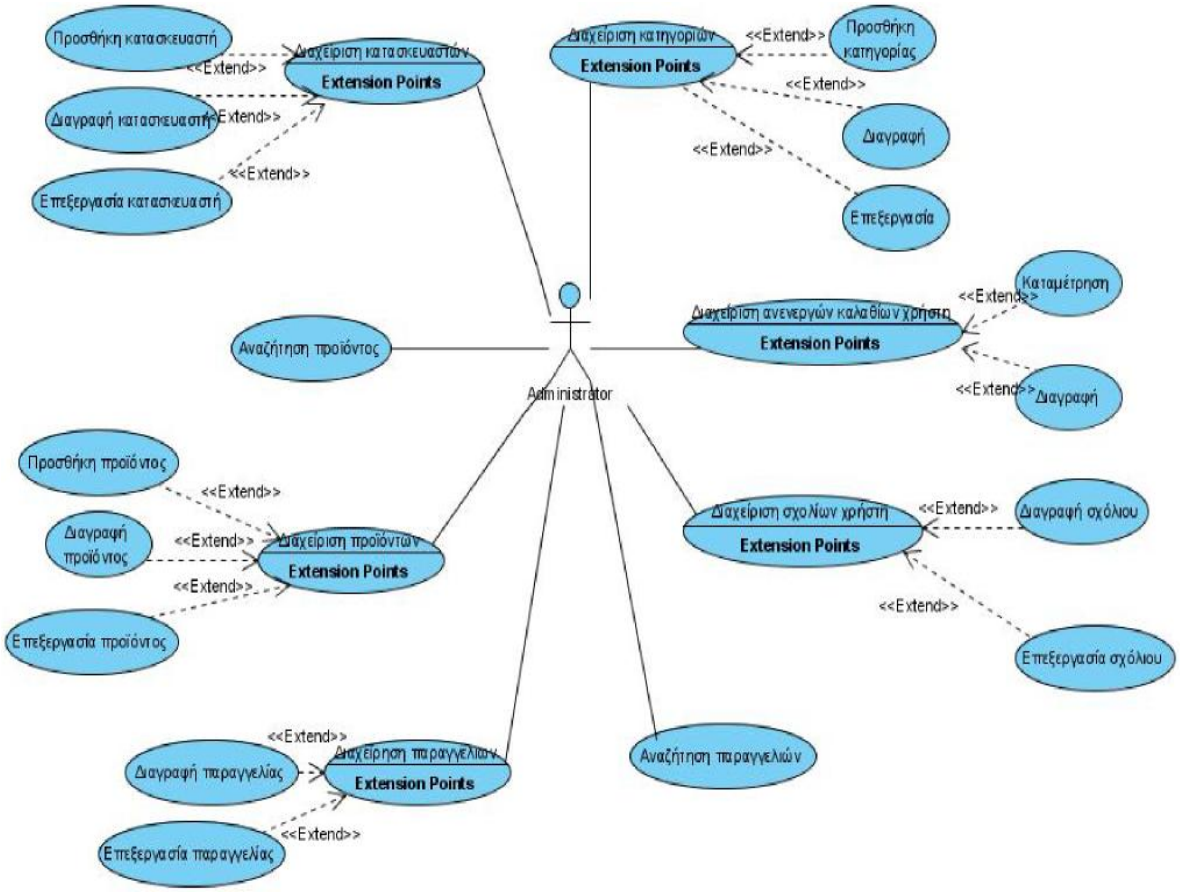
- Επικοινωνία με τον διαχειριστή του ηλεκτρονικού καταστήματος.

Πίνακας σεναρίων ανα περίπτωση χρήσης

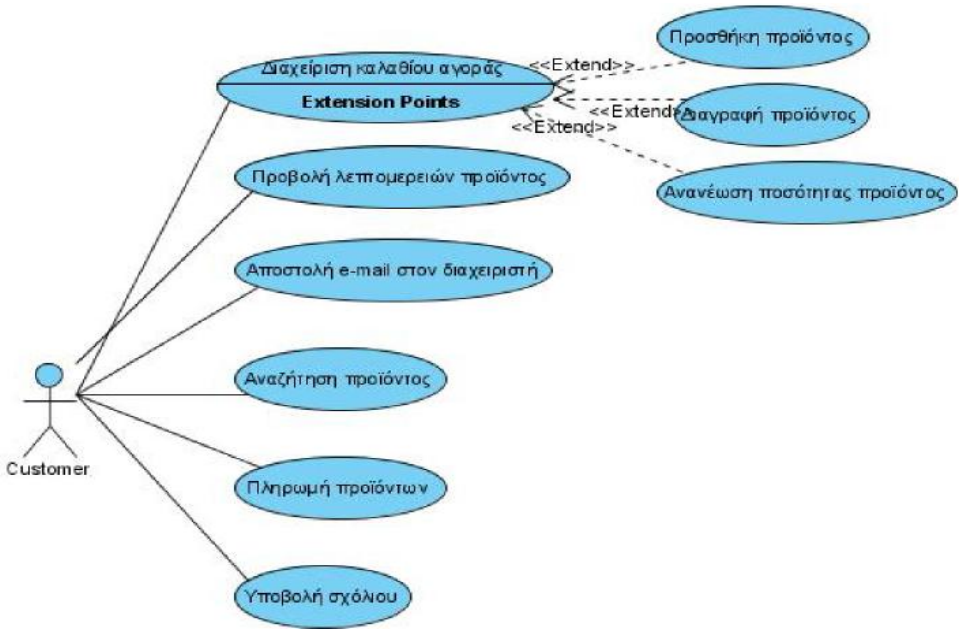
ΠΕΡΙΠΤΩΣΕΙΣ ΧΡΗΣΗΣ	ΣΕΝΑΡΙΑ
Διαχείριση προϊόντων	Προσθήκη προϊόντος Διαγραφή προϊόντος Επεξεργασία προϊόντος Προσθήκη κατηγορίας ή κατασκευαστή Διαγραφή κατηγορίας ή κατασκευαστή Επεξεργασία κατηγορίας ή κατασκευαστή Τροποποίηση ή διαγραφή σχόλιου χρήστη. Αναζήτηση προϊόντος. Προβολή λεπτομερειών προϊόντος Υποβολή σχολίου για κάποιο προϊόν.
Διαχείριση παραγγελιών	Επεξεργασία στοιχείων παραγγελίας Διαγραφή παραγγελίας Αναζήτηση παραγγελιών βάση κριτηρίων. Διαχείριση καλαθιού αγοράς (προσθήκη προϊόντος, διαγραφή προϊόντος, ανανέωση ποσότητας προϊόντος). Οικονομική εκκαθάριση (πληρωμή προϊόντων) Καταμέτρηση και διαγραφή ανενεργών καλαθιών αγοράς. Επικοινωνία με τον διαχειριστή του ηλεκτρονικού καταστήματος.

Πίνακας 19. Πίνακας σεναρίων ανά περίπτωση χρήσης.

Στη συνέχεια, παρουσιάζονται τα Use Case Diagrams που περιγράφουν τις δράσεις των δυο κατηγοριών χρηστών του συστήματος, δηλαδή του διαχειριστή και του πελάτη.



Εικόνα 42. Use Case Diagram – Διαχειριστής



Εικόνα 43. Use Case Diagram – Πελάτης

6.9.3.1 Περίπτωση Χρήσης και Σενάρια: Διαχείριση προϊόντων

Στη συνέχεια, όπως φαίνεται και στον παρακάτω πίνακα, εντοπίζονται οι δράσεις για κάθε σενάριο της περίπτωσης χρήσης «Διαχείριση προϊόντων».

Δράσεις σεναρίων περίπτωσης χρήσης «Διαχείριση προϊόντων»

Σύστημα: E-laptops	
User: Διαχειριστής, τελικός - χρήστης.	
Περίπτωση χρήσης: Διαχείριση προϊόντων	
Σενάρια	**Δράσεις**
Προσθήκη προϊόντος.	Επιλογή φωτογραφιών προϊόντος. Καταχώρηση ονόματος και τεχνικών χαρακτηριστικών προϊόντος. Επιλογή κατηγορίας και κατασκευαστή προϊόντος. Υποβολή προσθήκης.
Επεξεργασία προϊόντος.	Αλλαγή φωτογραφιών προϊόντος. Τροποποίηση τεχνικών χαρακτηριστικών προϊόντος Αλλαγή κατηγορίας ή/και κατασκευαστή προϊόντος. Υποβολή αλλαγών.
Διαγραφή προϊόντος.	Επιλογή προϊόντος. Διαγραφή προϊόντος. Επιβεβαίωση διαγραφής.
Προσθήκη κατηγορία ή κατασκευαστή προϊόντων.	Καταχώρηση ονομασίας κατηγορίας / κατασκευαστή. Καταχώρηση περιγραφής κατηγορίας/κατασκευαστή. Υποβολή προσθήκης.
Επεξεργασία κατηγορίας ή κατασκευαστή προϊόντων.	Αλλαγή ονομασίας κατηγορίας / κατασκευαστή. Αλλαγή περιγραφής κατηγορίας / κατασκευαστή. Υποβολή αλλαγών.

Σύστημα: E-laptops	
Διαγραφή κατηγορίας ή κατασκευαστή προϊόντων.	Επιλογή κατηγορίας/κατασκευαστή. Υποβολή διαγραφής κατηγορίας / κατασκευαστή. Επιβεβαίωση διαγραφής.
Τροποποίηση ή διαγραφή σχολίου χρήστη.	Επιλογή προϊόντος. Προβολή σχολίων που έχουν αναρτηθεί για το προϊόν . Επιλογή σχολίου. Διαγραφή ή τροποποίηση σχολίου.
Αναζήτηση προϊόντος.	Εισαγωγή λέξης στο searchtextbox της εφαρμογής. Προβολή αποτελεσμάτων αναζήτησης.
Προβολή λεπτομερειών προϊόντος.	Επιλογή προϊόντος.
Υποβολή σχολίου για κάποιο προϊόν.	Εισαγωγή στοιχείων χρήστη. Εισαγωγή σχολίου. Υποβολή σχολίου.

Πίνακας 20. Δράσεις σεναρίων περίπτωσης χρήσης «Διαχείριση προϊόντων»

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.1 Σενάριο: Προσθήκη κατηγορίας / κατασκευαστή προϊόντων

Η εφαρμογή παρέχει τη δυνατότητα προσθήκης μιας νέας κατηγορίας ή κατασκευαστή προϊόντων. Η καινούργια κατηγορία/κατασκευαστής θα αποθηκεύεται στη βάση δεδομένων και θα εμφανίζεται άμεσα στο αντίστοιχο μενού του ηλεκτρονικού καταστήματος. Για την προσθήκη κατηγορίας/κατασκευαστή πρέπει να εκτελεστούν τα παρακάτω βήματα.

Βήματα σεναρίου:

1. Καταχώρηση ονομασίας κατηγορίας / κατασκευαστή.
2. Καταχώρηση περιγραφής κατηγορίας / κατασκευαστή.
3. Υποβολή προσθήκης.

Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Καταχώρηση ονομασίας κατηγορίας / κατασκευαστή.

Ο χρήστης καταχωρεί μια σύντομη ονομασία για την κατηγορία.

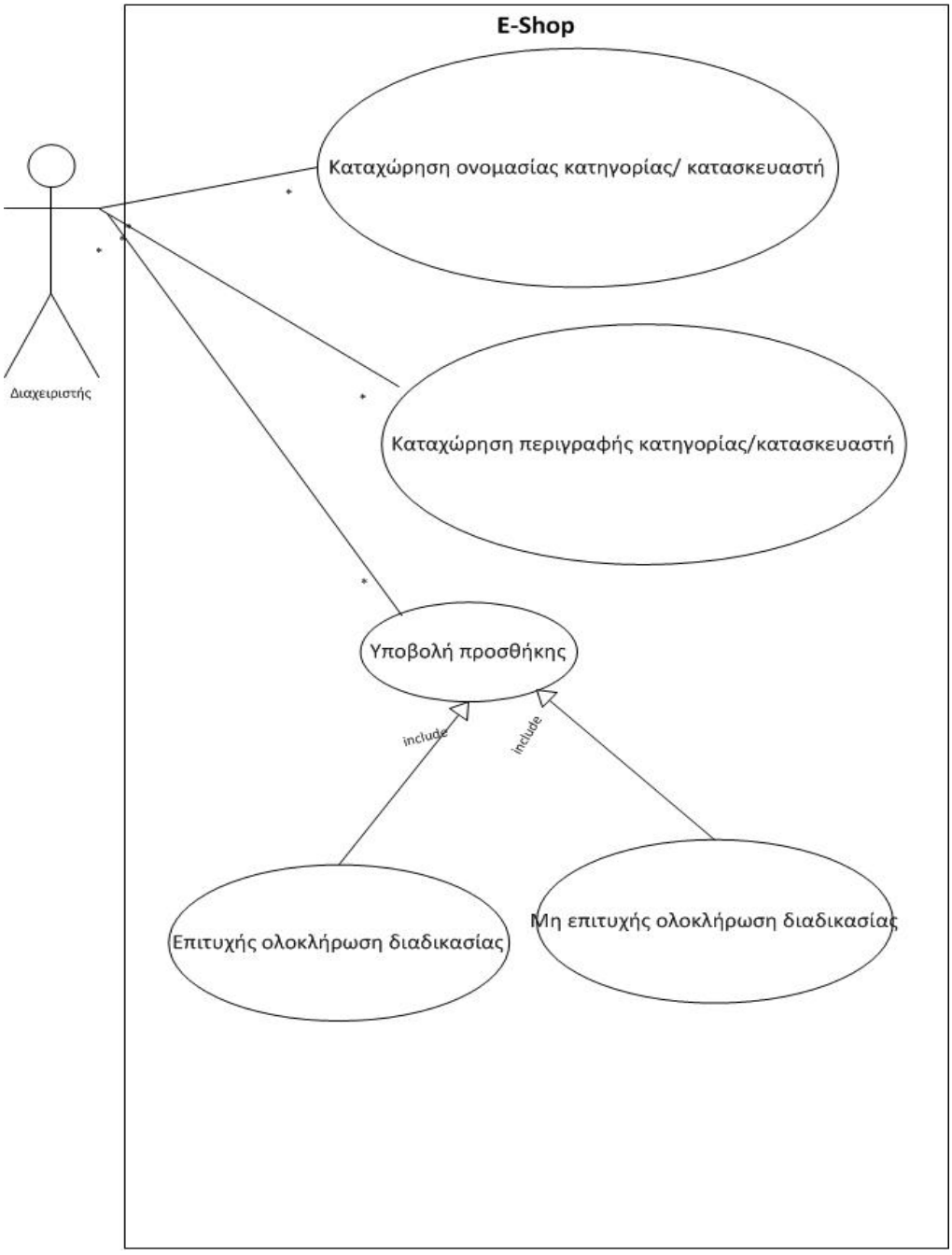
Βήμα 2: Καταχώρηση περιγραφής κατηγορίας / κατασκευαστή.

Ο χρήστης καταχωρεί μια σύντομη περιγραφή για την κατηγορία.

Βήμα 3: Υποβολή προσθήκης.

Πατώντας το κουμπί “Create category”, το σύστημα ελέγχει τα στοιχεία που εισήγαγε ο χρήστης και σε περίπτωση που είναι έγκυρα καταχωρεί την κατηγορία/κατασκευαστή στην εφαρμογή και ενημερώνει τον χρήστη για την επιτυχή ολοκλήρωση της διαδικασίας. Σε αντίθετη περίπτωση, εμφανίζει κατάλληλο μήνυμα λάθους.

Εικόνα 44. Σχηματική απεικόνιση σεναρίου



Εικόνα 45. Διάγραμμα περίπτωσης χρήσης προσθήκης κατηγορίας/κατασκευαστή προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.2 Σενάριο: Επεξεργασία κατηγορίας / κατασκευαστή προϊόντων

Η εφαρμογή παρέχει την δυνατότητα επεξεργασίας μιας υπάρχουσας κατηγορίας ή κατασκευαστή προϊόντων. Η τροποποιημένη κατηγορία/κατασκευαστής θα αποθηκεύεται στη βάση δεδομένων και θα εμφανίζεται άμεσα στο αντίστοιχο μενού του ηλεκτρονικού καταστήματος Για την επεξεργασία κατηγορίας/κατασκευαστή πρέπει να εκτελεστούν τα παρακάτω βήματα.

Βήματα σεναρίου:

1. Αλλαγή ονομασίας κατηγορίας / κατασκευαστή.

2. Αλλαγή περιγραφής κατηγορίας / κατασκευαστή.

3. Υποβολή αλλαγών.

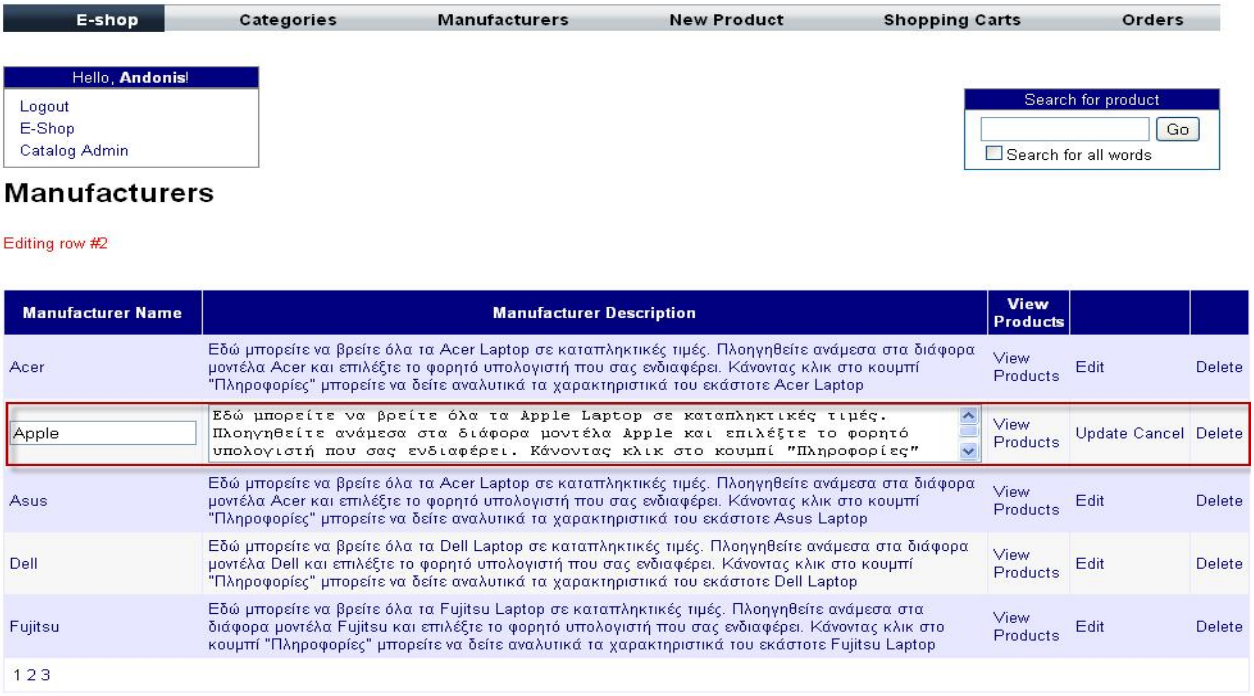
Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1-2: Αλλαγή ονομασίας και περιγραφής κατηγορίας/κατασκευαστή.

Ο χρήστης επιλέγει την κατηγορία και στη συνέχεια πατώντας το κουμπί “Edit” μπορεί να τροποποιήσει το όνομα και την περιγραφή της.

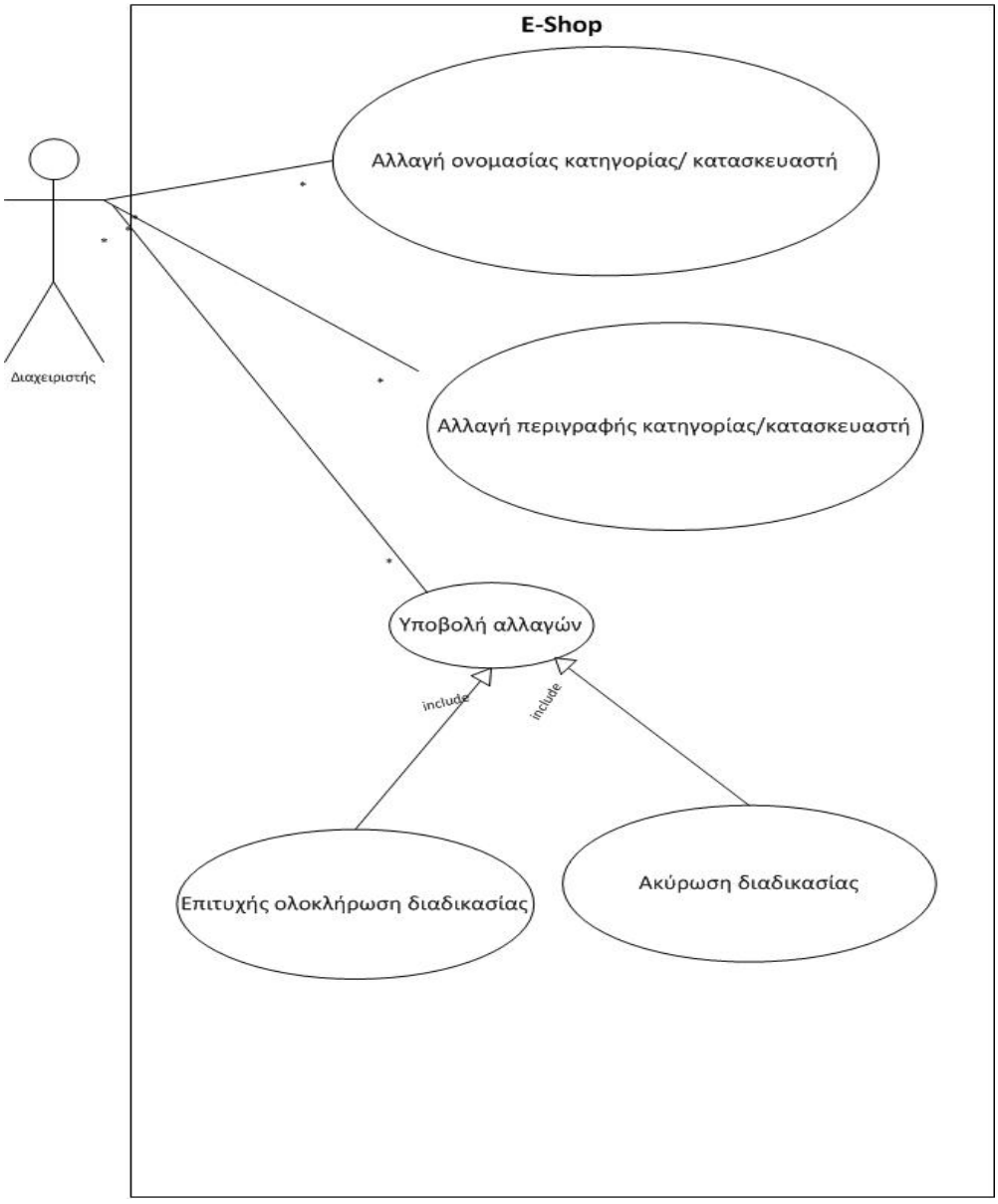
Βήμα 3: Υποβολή αλλαγών.

Πατώντας το κουμπί “Update” το σύστημα ελέγχει τα στοιχεία που εισήγαγε ο χρήστης και σε περίπτωση που είναι έγκυρα καταχωρεί τις αλλαγές στην εφαρμογή και ενημερώνει τον χρήστη για την επιτυχή ολοκλήρωση της διαδικασίας. Ο χρήστης μπορεί να ακυρώσει τη διαδικασία επεξεργασίας πατώντας το κουμπί “Cancel”.



Βήματα 1-3:

Εικόνα 46. Σχηματική απεικόνιση σεναρίου



Εικόνα 47. Διάγραμμα περίπτωσης χρήσης επεξεργασίας κατηγορίας/κατασκευαστή προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.3 Σενάριο: Διαγραφή κατηγορίας / κατασκευαστή προϊόντων

Η εφαρμογή παρέχει την δυνατότητα διαγραφής μίας κατηγορίας ή κατασκευαστή προϊόντων. Μετά τη διαγραφή θα αφαιρείται από τη βάση δεδομένων και από το αντίστοιχο μενού του ηλεκτρονικού καταστήματος. Για την διαγραφή μίας κατηγορίας/κατασκευαστή πρέπει να εκτελεστούν τα παρακάτω βήματα:

Βήματα σεναρίου:

1. Επιλογή κατηγορίας / κατασκευαστή.

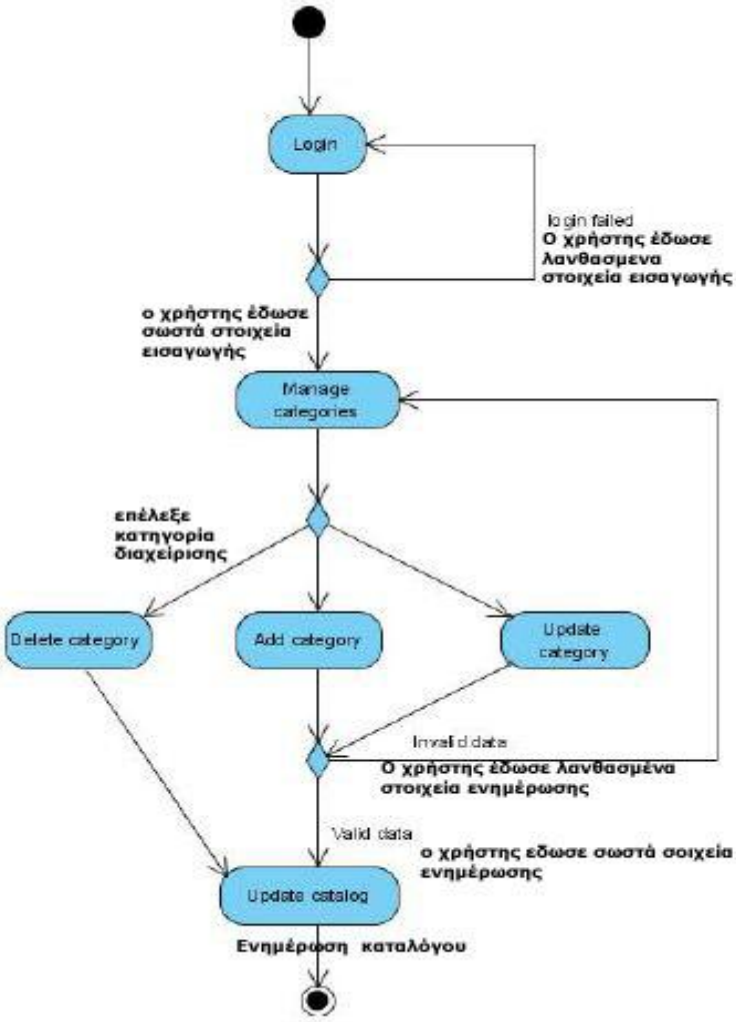
2. Διαγραφή κατηγορίας / κατασκευαστή.

3. Επιβεβαίωση διαγραφής.

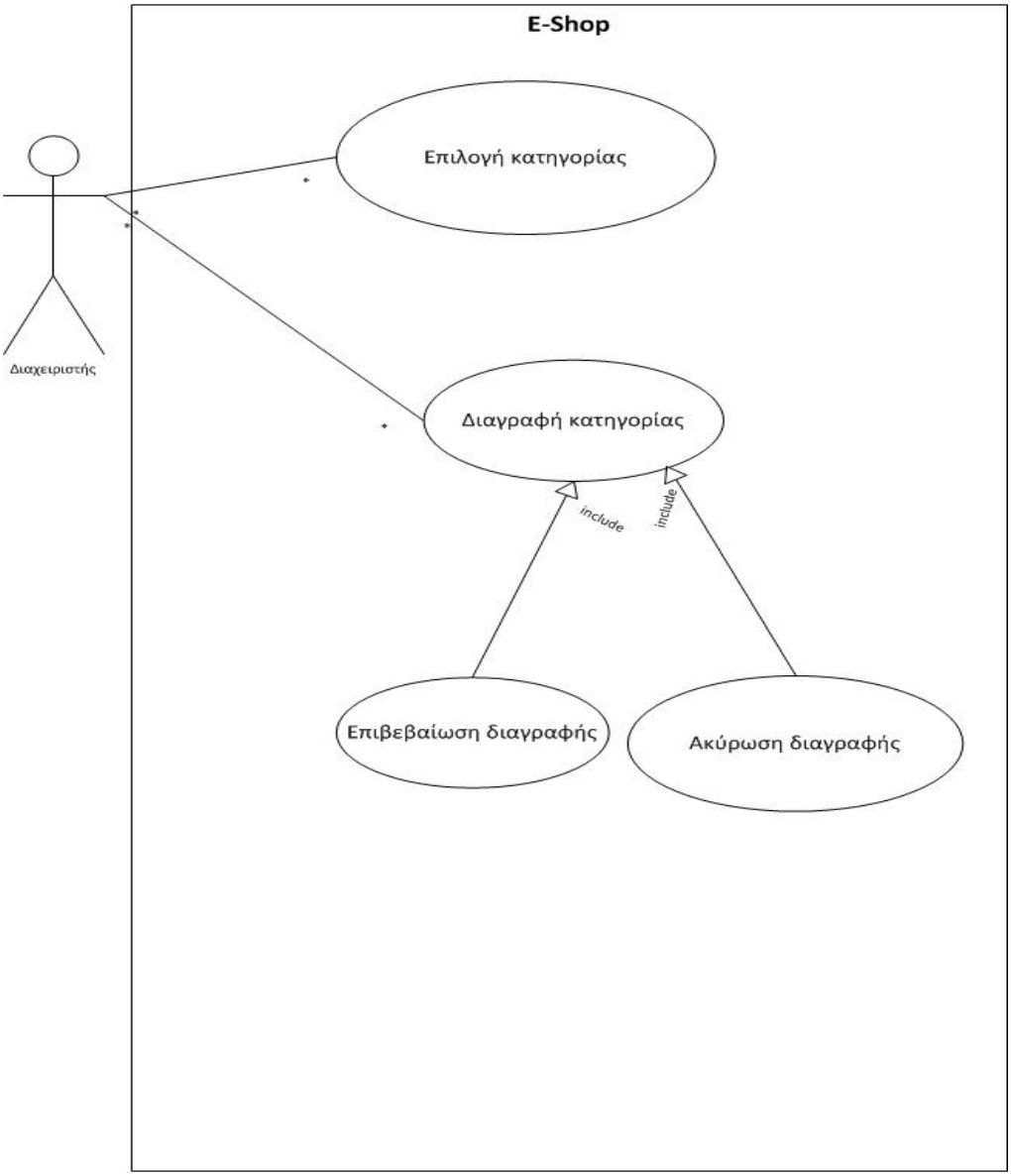
Αναλυτική περιγραφή βημάτων σεναρίου:

Βήματα 1-3: Επιλογή και διαγραφή κατηγορίας/κατασκευαστή.

Ο χρήστης επιλέγει την κατηγορία/κατασκευαστή που επιθυμεί να διαγράψει και πατάει το κουμπί “Delete”. Τότε, το σύστημα τον ενημερώνει ότι μαζί με την κατηγορία θα διαγραφτούν και όλα τα προϊόντα που σχετίζονται με αυτή. Ο χρήστης μπορεί είτε να επιβεβαιώσει την διαγραφή είτε να την ακυρώσει.



Εικόνα 48. Activity Diagram – Διαχείριση κατηγοριών



Εικόνα 49. Διάγραμμα περίπτωσης χρήσης διαγραφή κατηγορίας προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.4 Σενάριο: Προσθήκη προϊόντος

Η εφαρμογή παρέχει τη δυνατότητα προσθήκης νέου προϊόντος. Το νέο προϊόν θα αποθηκεύεται στη βάση δεδομένων και θα εμφανίζεται άμεσα στον κατάλογο προϊόντων του ηλεκτρονικού καταστήματος. Για την προσθήκη προϊόντος πρέπει να εκτελεστούν τα παρακάτω βήματα.

Βήματα σεναρίου:

1.Επιλογή φωτογραφιών προϊόντος.

2.Καταχώρηση ονόματος και τεχνικών χαρακτηριστικών προϊόντος.

3.Επιλογή κατηγορίας και κατασκευαστή προϊόντος.

4.Υποβολή προσθήκης.

Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Επιλογή φωτογραφιών προϊόντος.

Ο διαχειριστής μπορεί να κάνει upload μέχρι δυο φωτογραφίες που θα αντιπροσωπεύουν το προϊόν. Η πρώτη φωτογραφία θα εμφανίζεται στον κατάλογο προϊόντων ενώ η δεύτερη, που θα είναι μεγαλύτερων διαστάσεων, θα εμφανίζεται στη σελίδα “Product Details”.

Βήμα 2: Καταχώρηση ονόματος και τεχνικών χαρακτηριστικών προϊόντος.

Στο βήμα αυτό ο διαχειριστής επιλέγει μια ονομασία για το προϊόν και καταχωρεί τα τεχνικά χαρακτηριστικά του (σκληρός δίσκος, οθόνη κλπ). Επίσης, μπορεί να επιλέξει εάν το προϊόν θα εμφανίζεται στην αρχική σελίδα του ηλεκτρονικού καταστήματος.

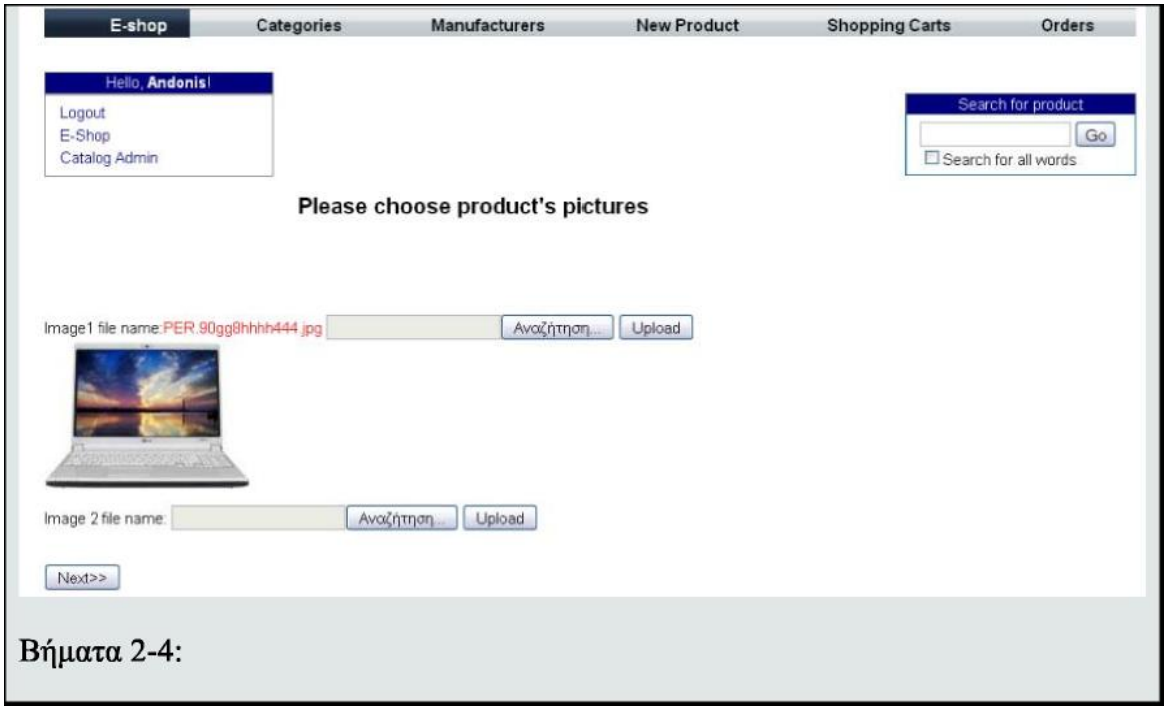
Βήμα 3: Επιλογή κατηγορίας και κατασκευαστή προϊόντος

Επιλέγεται η κατηγορία και ο κατασκευαστής στον οποίον θα ανήκει το προϊόν. Η επιλογή γίνεται από το αντίστοιχο drop-downlist.

Βήμα 4: Υποβολή προσθήκης

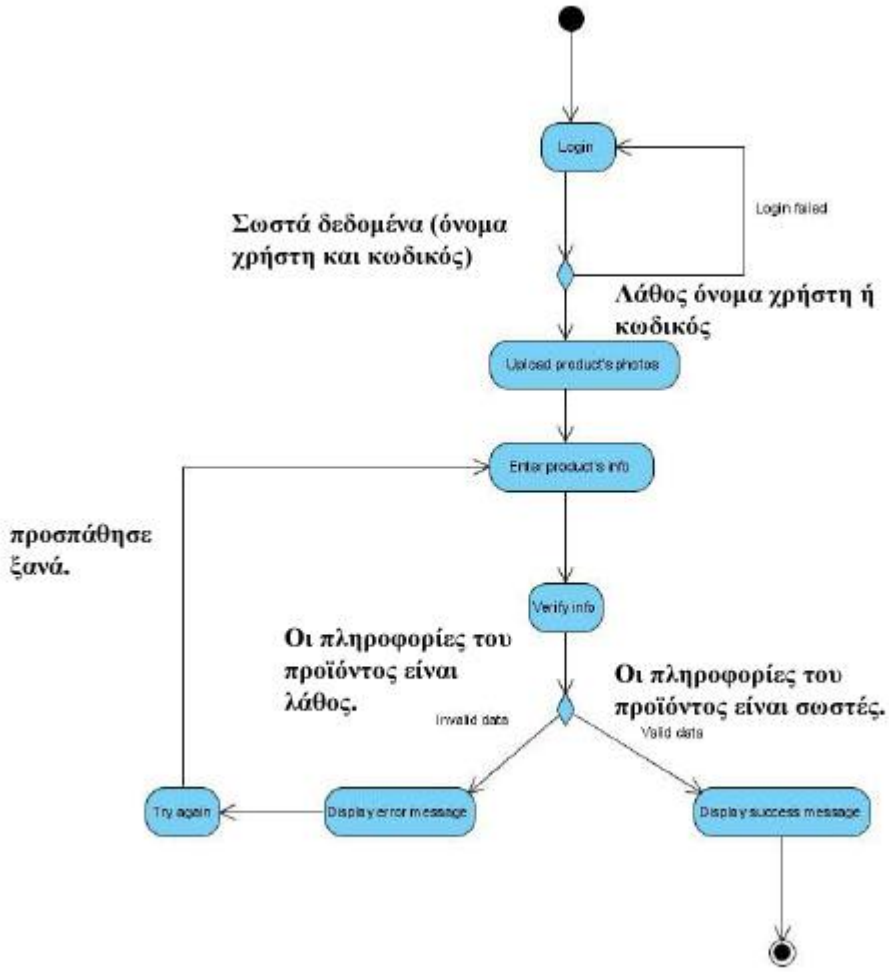
Στο βήμα αυτό τα δεδομένα που εισήχθησαν ελέγχονται από το σύστημα. Στην περίπτωση που είναι έγκυρα, το νέο προϊόν καταχωρείται στη βάση δεδομένων και ο διαχειριστής ενημερώνεται για την επιτυχή ολοκλήρωση της διαδικασίας. Στην αντίθετη περίπτωση, εμφανίζεται επεξηγηματικό μήνυμα που δείχνει στον διαχειριστή ότι πρέπει να διορθώσει τα μη έγκυρα δεδομένα.

Βήμα1:

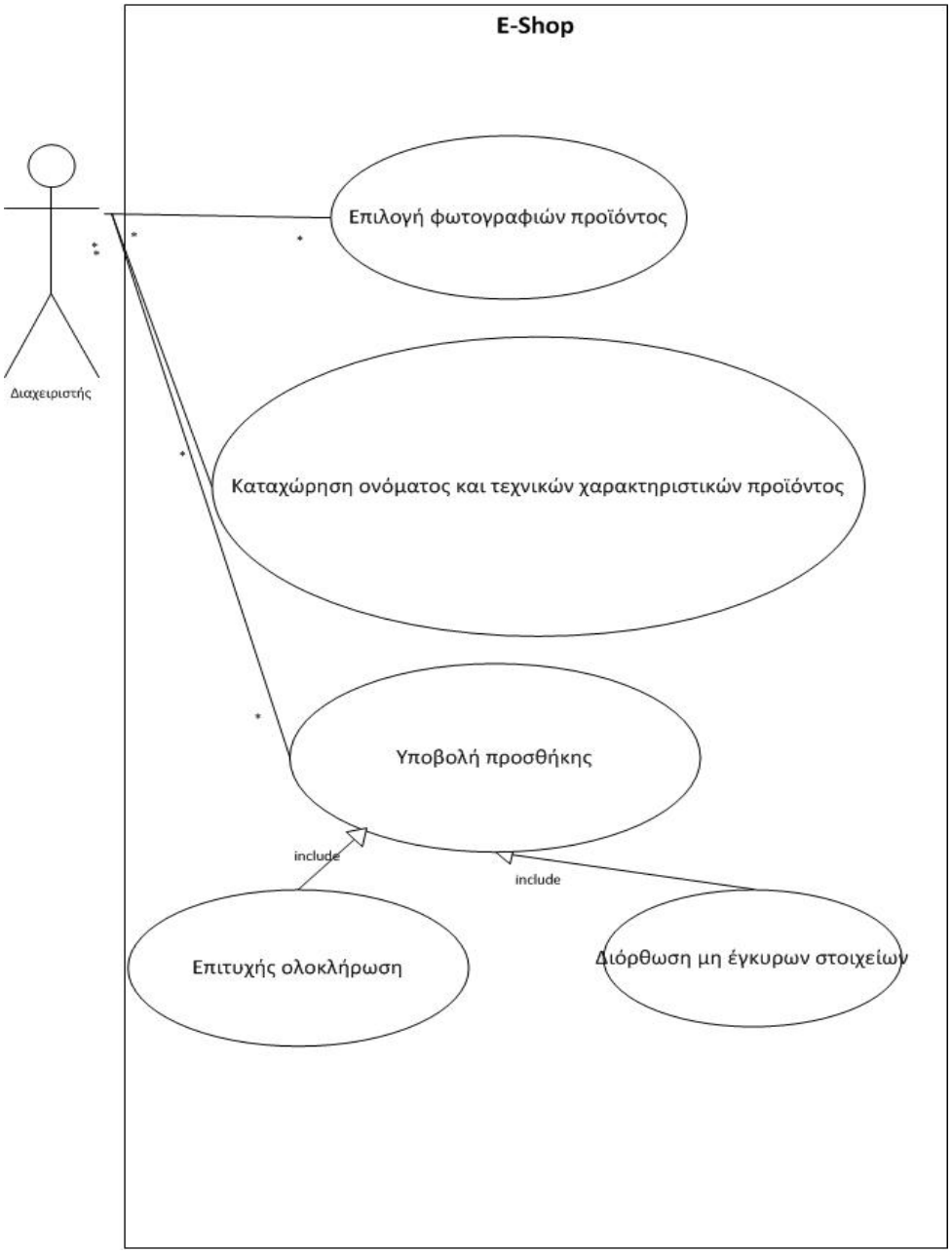


Εικόνα 50. Σχηματική απεικόνιση σεναρίου

Εικόνα 51. Activity Diagram – Προσθήκη προϊόντος



Εικόνα 52. Activity Diagram – Προσθήκη προϊόντος



Εικόνα 53. Διάγραμμα περίπτωσης χρήσης προσθήκης προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.5 Σενάριο: Επεξεργασία προϊόντος

Η εφαρμογή παρέχει τη δυνατότητα τροποποίησης ενός προϊόντος. Ο χρήστης μπορεί να αλλάξει τις φωτογραφίες που θα αντιπροσωπεύουν το προϊόν, τα τεχνικά χαρακτηριστικά του, καθώς και τον κατασκευαστή ή την κατηγορία στην οποία ανήκει. Για τη επεξεργασία ενός προϊόντος πρέπει να εκτελεστούν τα παρακάτω βήματα.

Βήματα σεναρίου:

- Αλλαγή φωτογραφιών προϊόντος.
- Τροποποίηση τεχνικών χαρακτηριστικών προϊόντος.
- Αλλαγή κατηγορίας ή/και κατασκευαστή προϊόντος.
- Υποβολή αλλαγών.

Αναλυτική περιγραφή βημάτων σεναρίου:

Βήμα 1: Αλλαγή φωτογραφιών προϊόντος.

Ο διαχειριστής πατώντας το κουμπί “Browse Images” μπορεί να κάνει upload διαφορετικές φωτογραφίες που θα αντιπροσωπεύουν το προϊόν.

Βήμα 2: Τροποποίηση τεχνικών χαρακτηριστικών προϊόντος.

Ο διαχειριστής μπορεί να αλλάξει οποιοδήποτε από τα τεχνικά χαρακτηριστικά του επιλεγμένου προϊόντος (σκληρός δίσκος, οθόνη κλπ).

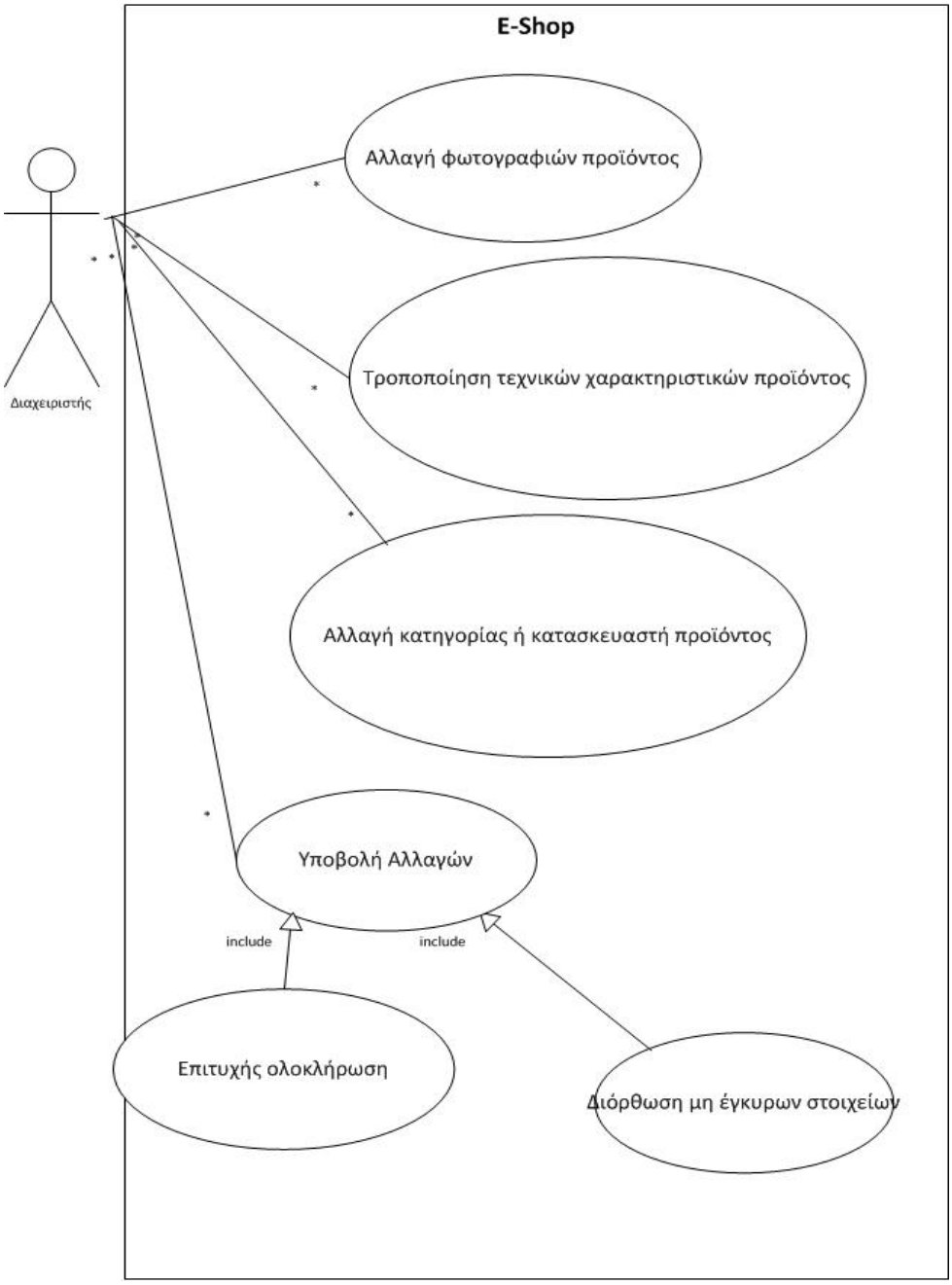
Βήμα 3: Αλλαγή κατηγορίας ή/και κατασκευαστή προϊόντος.

Ο διαχειριστής μπορεί να αλλάξει την κατηγορία ή τον κατασκευαστή στον οποίο θα ανήκει το προϊόν. Η αλλαγή γίνεται από το αντίστοιχο drop-downlist.

Βήμα 4: Υποβολή αλλαγών

Στο βήμα αυτό τα δεδομένα που εισήχθησαν ελέγχονται από το σύστημα.

Σε περίπτωση που είναι έγκυρα, οι αλλαγές που έγιναν καταχωρούνται στη βάση δεδομένων και ο διαχειριστής ενημερώνεται για την επιτυχή ολοκλήρωση της διαδικασίας. Σε αντίθετη περίπτωση εμφανίζεται επεξηγηματικό μήνυμα που δείχνει στον διαχειριστή ότι πρέπει να διορθώσει τα μη έγκυρα δεδομένα.



Εικόνα 54. Διάγραμμα περίπτωσης χρήσης επεξεργασίας προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.6 Σενάριο: Διαγραφή Προϊόντος

Η εφαρμογή παρέχει τη δυνατότητα διαγραφής ενός προϊόντος. Το προϊόν θα αφαιρείται άμεσα από τη βάση δεδομένων και από τον κατάλογο του ηλεκτρονικού καταστήματος.

Βήματα σεναρίου:

1.Επιλογή προϊόντος.

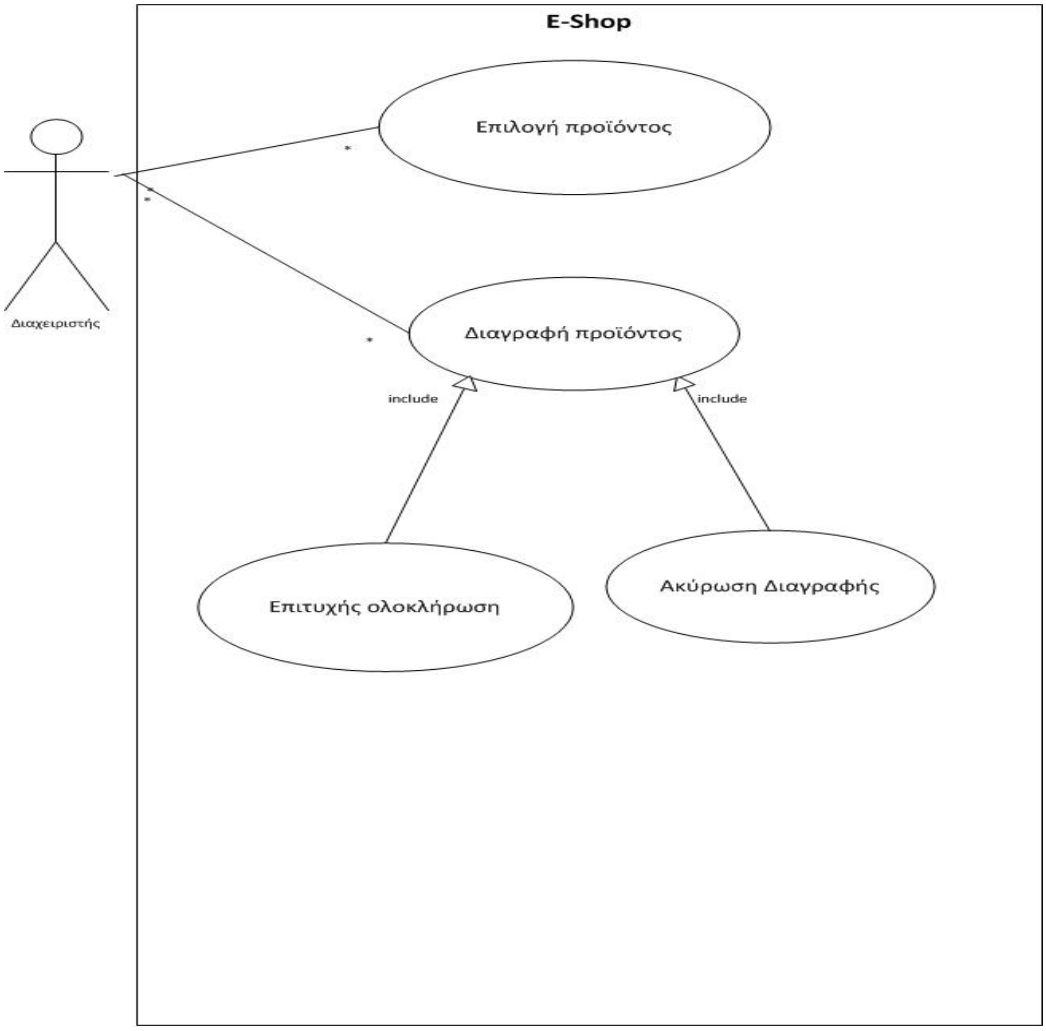
2.Διαγραφή προϊόντος.

3.Επιβεβαίωση διαγραφής.

Αναλυτική περιγραφή βημάτων σεναρίου:

Βήμα 1-3: Επιλογή και διαγραφή προϊόντος

Ο διαχειριστής, επιλέγει το προϊόν που επιθυμεί να διαγράψει και πατάει το κουμπί“Delete”. Το σύστημα ζητάει την επιβεβαίωση της διαγραφής. Ο χρήστης μπορεί είτε να επιβεβαιώσει τη διαγραφή είτε να την ακυρώσει.



Εικόνα 55. Διάγραμμα περίπτωσης χρήσης διαγραφής προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.7 Σενάριο: Τροποποίηση ή διαγραφή σχόλιου χρήστη

Ο διαχειριστής μπορεί να διαγράψει ένα σχόλιο η να τροποποιήσει το περιεχόμενο του εάν το κρίνει απαραίτητο. Για τις παραπάνω ενέργειες πρέπει να εκτελεστούν τα εξής βήματα.

Βήματα σεναρίου:

1.Επιλογή προϊόντος.

2.Προβολή σχολίων που έχουν αναρτηθεί για το προϊόν.

3.Επιλογή σχολίου.

4.Διαγραφή ή τροποποίηση σχολίου.

Αναλυτική περιγραφή βημάτων σεναρίου:

Βήμα 1-2: Επιλογή προϊόντος και προβολή των σχολίων που έχουν γίνει για αυτό.

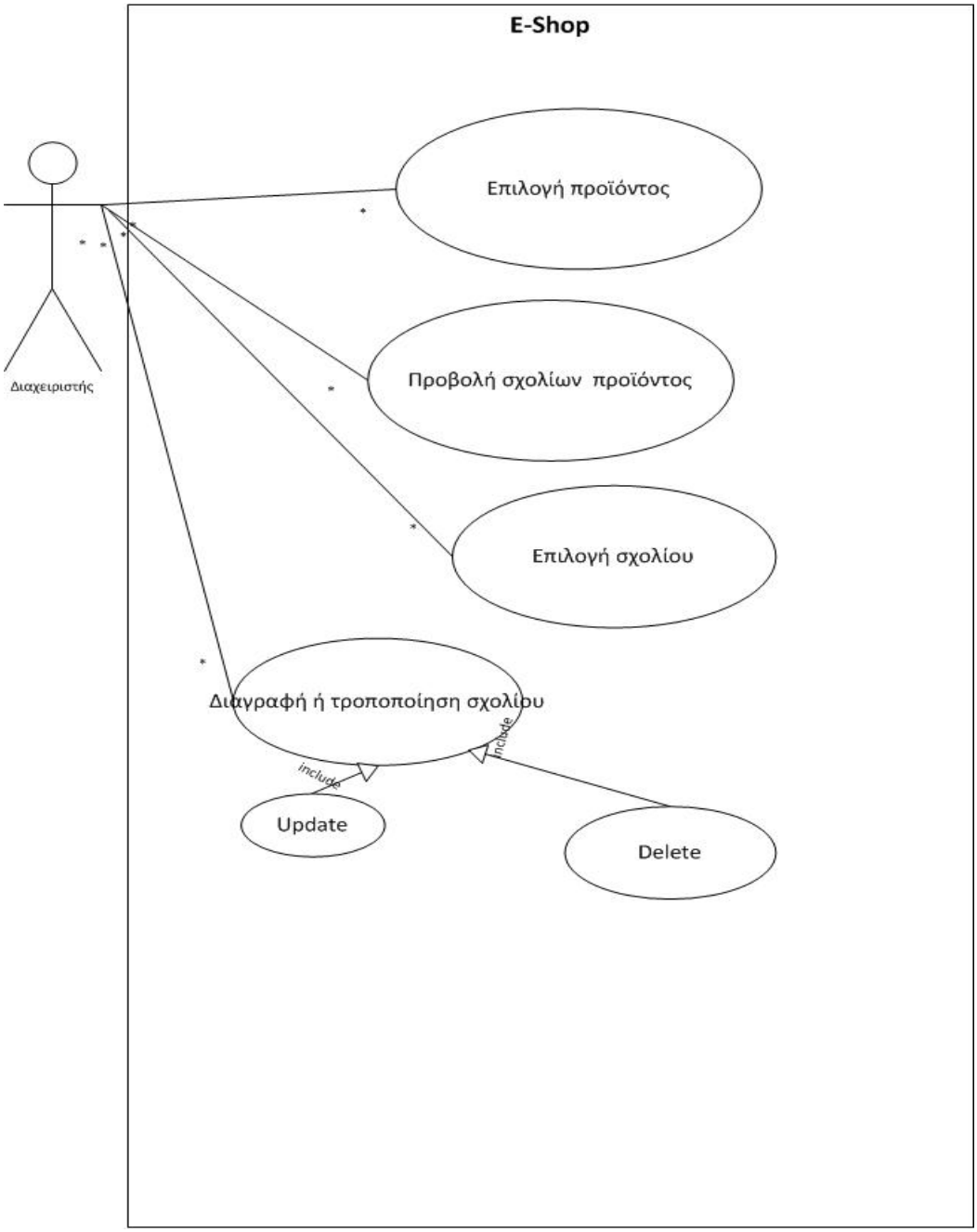
Ο διαχειριστής επιλέγει κάποιο προϊόν και κάνει κλικ στο πεδίοComments ώστε να δει τα σχόλια που έχουν αναρτηθεί για αυτό.

Βήμα 3: Επιλογή σχολίου.

Στη συνέχεια, ο διαχειριστής επιλέγει κάποιο σχόλιο πατώντας το κουμπί“Select”, οπότε ένα νέο interfaceεμφανίζεται με τα στοιχεία του σχολίου.

Βήμα 4: Διαγραφή ή τροποποίηση σχολίου.

Ο διαχειριστής μπορεί να τροποποιήσει το περιεχόμενο του σχολίου πατώντας το κουμπί “Update” ή να το διαγράψει πατώντας το κουμπί “Delete”.



Εικόνα 56. Διάγραμμα περίπτωσης χρήσης τροποποίησης ή διαγραφής σχολίου

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.8 Σενάριο: Προβολή λεπτομερειών προϊόντος

Ο χρήστης, από τον κατάλογο του ηλεκτρονικού καταστήματος, κάνοντας κλικ πάνω σε ένα προϊόν κατευθύνεται στη σελίδα “Product Details”, όπου του παρέχεται αναλυτική περιγραφή των χαρακτηριστικών του επιλεγμένου προϊόντος και μια φωτογραφία μεγαλύτερων διαστάσεων.

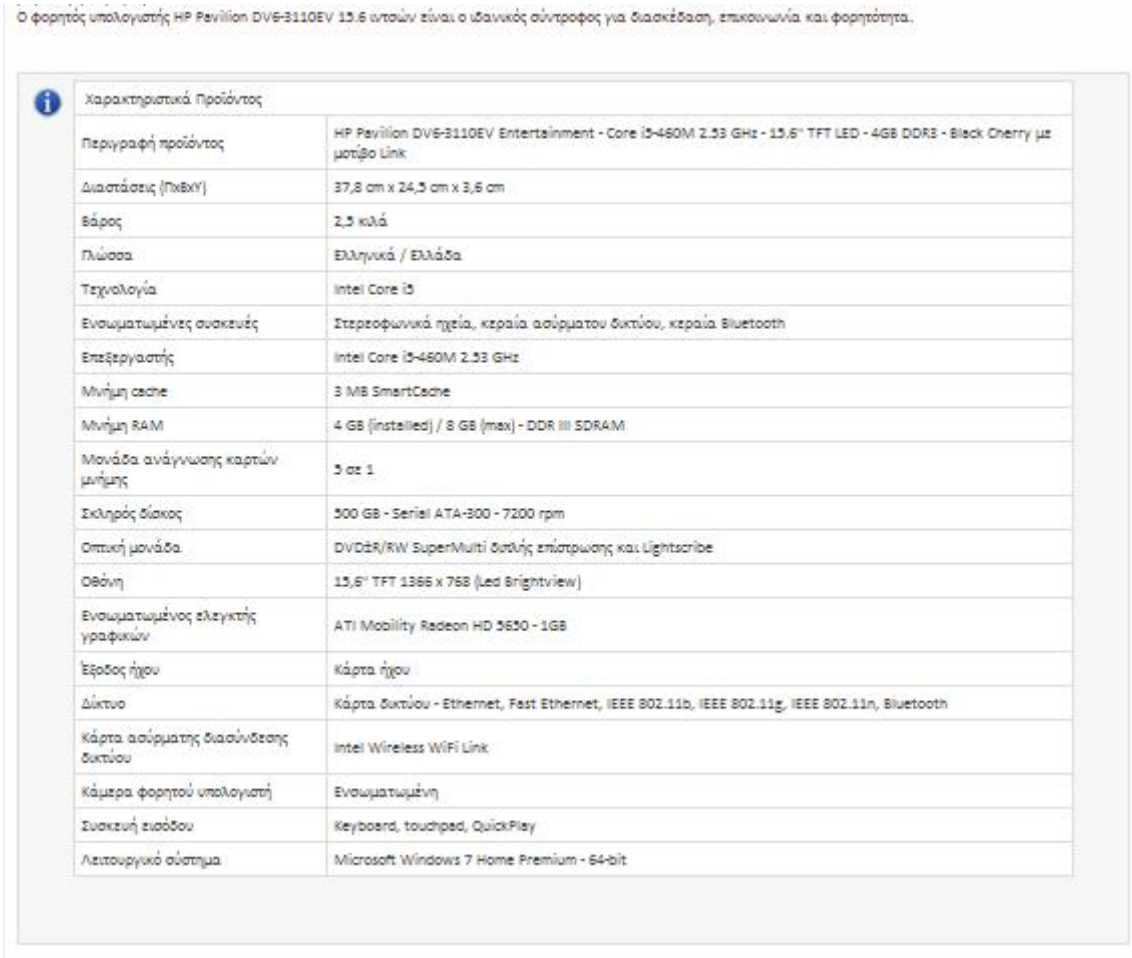
Βήματα σεναρίου:

Βήμα 1: Επιλογή προϊόντος

Αναλυτική περιγραφή βημάτων σεναρίου

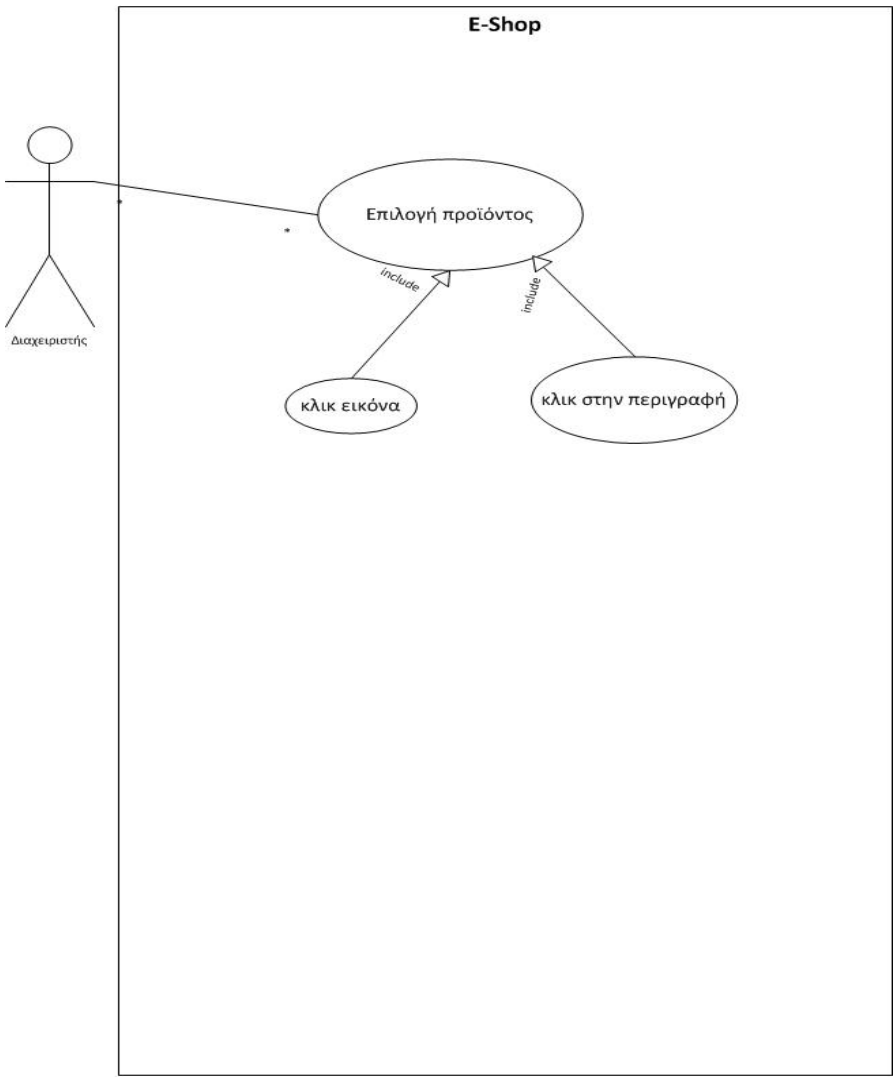
Βήμα 1: Επιλογή προϊόντος

Ο χρήστης επιλέγει κάποιο προϊόν κάνοντας κλικ είτε στην εικόνα του προϊόντος, είτε στο όνομα του είτε στη περιγραφή του και κατευθύνεται στη σελίδα “Product Details”.


Ο φορητός υπολογιστής HP Pavilion DV6-3110EV 15.6 ιντσών είναι ο ιδανικός σύντροφος για διασκέδαση, επικοινωνία και φορητότητα.

Χαρακτηριστικά Προϊόντος	
Περιγραφή προϊόντος	HP Pavilion DV6-3110EV Entertainment - Core i5-460M 2.53 GHz - 15.6" TFT LED - 4GB DDR3 - Black Cherry με μοτίβο Link
Διαστάσεις (ΠxΒxΥ)	37,8 cm x 24,5 cm x 3,6 cm
Βάρος	2,5 κιλά
Πώσσα	Ελληνικά / Ελλάδα
Τεχνολογία	Intel Core i5
Ενσωματωμένες συσκευές	Στερεοφωνικά ηχεία, κεραία ασύρματου δικτύου, κεραία Bluetooth
Επεξεργαστής	Intel Core i5-460M 2.53 GHz
Μνήμη cache	3 MB SmartCache
Μνήμη RAM	4 GB (installed) / 8 GB (max) - DDR III SDRAM
Μονάδα ανάγνωσης καρτών μνήμης	5 σε 1
Σκληρός δίσκος	500 GB - Serial ATA-300 - 7200 rpm
Οπτική μονάδα	DVD±R/RW SuperMulti διπλής επίστρωσης και Lightscribe
Οθόνη	15,6" TFT 1366 x 768 (Led Brightview)
Ενσωματωμένος ελεγκτής γραφικών	ATI Mobility Radeon HD 5650 - 1GB
Έξοδος ήχου	Κάρτα ήχου
Δίκτυο	Κάρτα δικτύου - Ethernet, Fast Ethernet, IEEE 802.11b, IEEE 802.11g, IEEE 802.11n, Bluetooth
Κάρτα ασύρματης διασύνδεσης δικτύου	Intel Wireless WiFi Link
Κάμερα φορητού υπολογιστή	Ενσωματωμένη
Συσκευή εισόδου	Keyboard, touchpad, QuickPlay
Λειτουργικό σύστημα	Microsoft Windows 7 Home Premium - 64-bit

Εικόνα 57. Σχηματική απεικόνιση σεναρίου



Εικόνα 58. Διάγραμμα περίπτωσης χρήσης προβολής λεπτομερειών προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.9 Σενάριο: Αναζήτηση προϊόντος

Ο χρήστης μπορεί να πραγματοποιήσει αναζήτηση για κάποιο προϊόν εκτελώντας τα παρακάτω βήματα:

Βήματα σεναρίου:

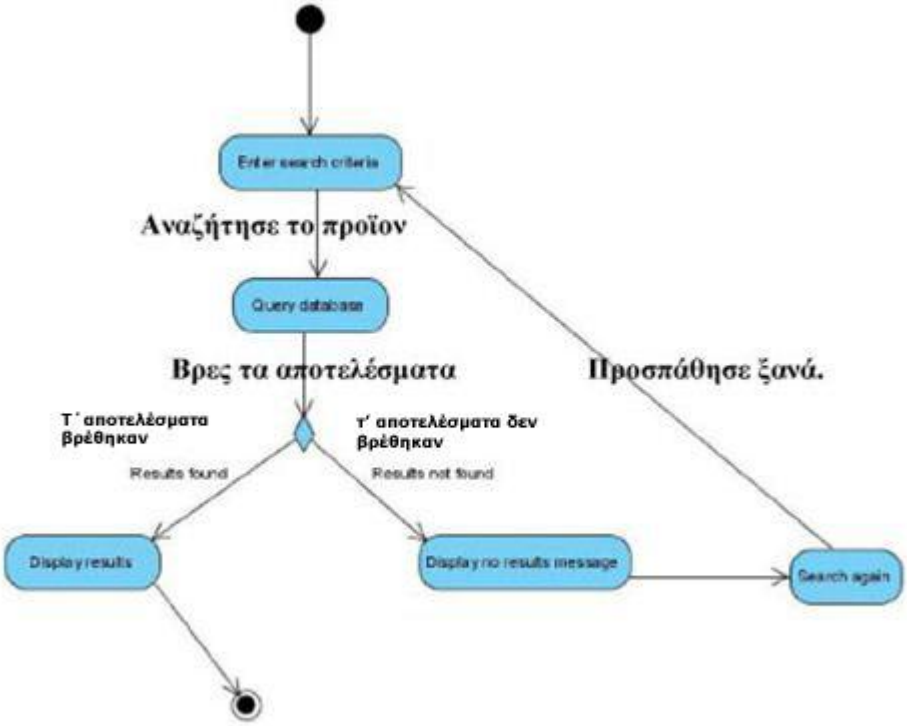
1.Εισαγωγή λέξης στο searchtextbox της εφαρμογής.

2.Προβολή αποτελεσμάτων αναζήτησης.

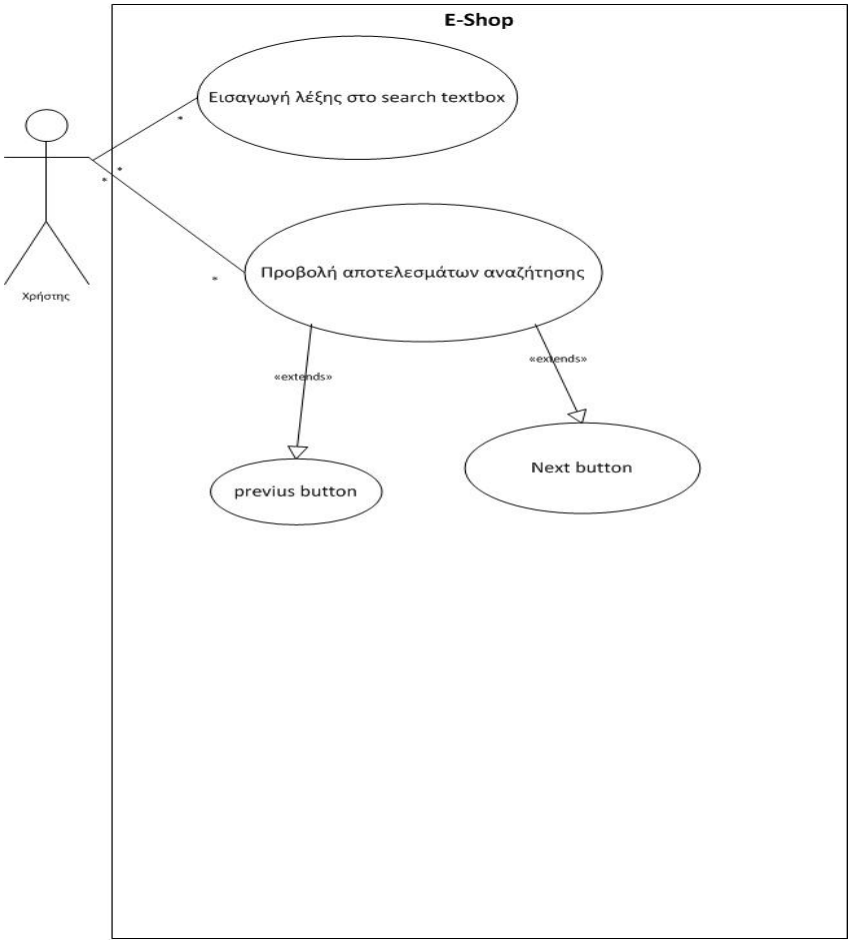
Αναλυτική περιγραφή βημάτων σεναρίου

Βήματα 1+2: Εισαγωγή λέξης στο search textbox της εφαρμογής και προβολή αποτελεσμάτων αναζήτησης.

Ο χρήστης εισάγει τη λέξη στο search textbox και πατώντας το κουμπί αναζήτησης κατευθύνεται στη σελίδα αποτελεσμάτων όπου μπορεί να δει τα προϊόντα που βρέθηκαν. Αν τα προϊόντα είναι περισσότερα από έξι τότε ο χρήστης μπορεί να πλοηγηθεί στις σελίδες αποτελεσμάτων χρησιμοποιώντας τα κουμπιά previous και next.



Εικόνα 59. Activity Diagram - Αναζήτηση προϊόντος



Εικόνα 60. Διάγραμμα περίπτωσης χρήσης αναζήτησης προϊόντος

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.10 Σενάριο: Υποβολή σχολίου για κάποιο προϊόν

Η εφαρμογή παρέχει τη δυνατότητα υποβολής σχολίου για τα προϊόντα του ηλεκτρονικού καταστήματος. Τα σχόλια που θα αναρτώνται θα είναι διαθέσιμα για όλους τους χρήστες και θα τους βοηθούν στην αξιολόγηση του εκάστοτε προϊόντος. Για την υποβολή σχολίου πρέπει να εκτελεστούν τα παρακάτω βήματα.

Βήματα σεναρίου:

1.Εισαγωγή στοιχείων χρήστη.

2.Εισαγωγή σχολίου.

3.Υποβολή σχολίου.

Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Εισαγωγή στοιχείων χρήστη.

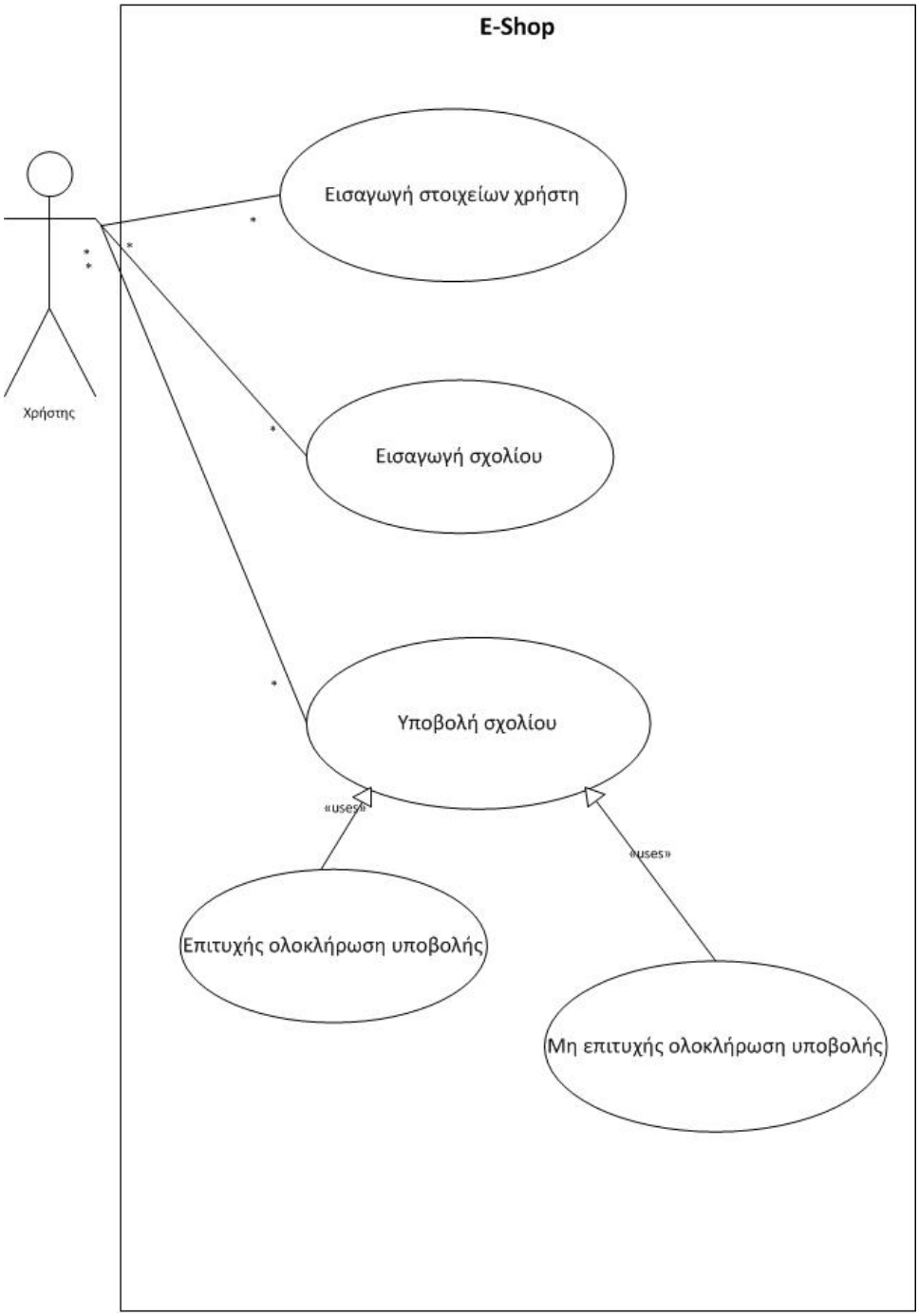
Ο χρήστης καλείται να συμπληρώσει το όνομα και το e-mail του τα οποία είναι υποχρεωτικά στοιχεία. Το e-mail του χρήστη θα είναι διαθέσιμο μόνο στον διαχειριστή.

Βήμα 2: Εισαγωγή σχολίου

Ο χρήστης εισάγει το σχόλιό του στο κατάλληλο textbox.

Βήμα 3: Υποβολή σχολίου.

Πατώντας το κουμπί “Submit”, τα στοιχεία ελέγχονται από το σύστημα. Στην περίπτωση που είναι έγκυρα, το σχόλιο αναρτάται και ο χρήστης ενημερώνεται για την επιτυχή ολοκλήρωση της διαδικασίας. Στην αντίθετη περίπτωση, εμφανίζεται επεξηγηματικό μήνυμα που δείχνει στον πελάτη ότι πρέπει να διορθώσει τα μη έγκυρα στοιχεία.



Εικόνα 61. Διάγραμμα περίπτωσης χρήσης υποβολής σχολίου για κάποιο προϊόν

Περίπτωση χρήσης: Διαχείριση προϊόντων

6.9.3.1.11 Περίπτωση Χρήσης και Σενάρια: Διαχείριση παραγγελιών

Στη συνέχεια, όπως φαίνεται και στον παρακάτω πίνακα, εντοπίζονται οι ενέργειες για κάθε σενάριο της περίπτωσης χρήσης «Διαχείριση παραγγελιών».

Δράσεις σεναρίων περίπτωσης χρήσης «Διαχείριση παραγγελιών»

Σύστημα: E-laptops	
User: Διαχειριστής, τελικός χρήστης	
Περίπτωση χρήσης: Διαχείριση παραγγελιών	
Σενάρια	**Ενέργειες**
Επεξεργασία στοιχείων παραγγελίας	Ορισμός του status της παραγγελίας Προσθήκη σχολίου που αφορά την παραγγελία Αλλαγή στοιχείων πελάτη Υποβολή αλλαγών
Αναζήτηση παραγγελιών βάση	Επιλογή τρόπου αναζήτησης Εμφάνιση αποτελεσμάτων
Διαγραφή παραγγελίας	Επιλογή παραγγελίας Διαγραφή παραγγελίας Επιβεβαίωση παραγγελίας
Διαχείριση καλαθιού αγορών	Προσθήκη προϊόντος στο καλάθι Αφαίρεση προϊόντος από το καλάθι Ανανέωση ποσότητας προϊόντος Έναρξη οικονομικής εκκάθαρσης (διαδικασία πληρωμής)

Σύστημα: E-laptops	
Οικονομική εκκάθαρση (πληρωμή)	Συμπλήρωση φόρμας παραγγελίας Υποβολή φόρμας Ανακατεύθυνση στο ηλεκτρονικό σύστημα Εξόφληση παραγγελίας
Καταμέτρηση ή διαγραφή καλαθιών αγορών	Επιλογή χρονικής περιόδου Καταμέτρηση ή διαγραφή καλαθιών αγοράς
Επικοινωνία με τον διαχειριστή	Συμπλήρωση φόρμας επικοινωνίας Αποστολή μηνύματος

Πίνακας 21. Δράσεις σεναρίων περίπτωσης χρήσης «Διαχείρισης παραγγελιών»

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.12 Σενάριο: Επεξεργασία στοιχείων παραγγελίας

Ο διαχειριστής μπορεί να τροποποιήσει τα στοιχεία μιας παραγγελίας όπως το status της παραγγελίας, την προσθήκη σχολίου και την αλλαγή των στοιχείων του πελάτη. Για την επεξεργασία των στοιχείων μίας παραγγελίας πρέπει να εκτελεστούν τα εξής βήματα:

Βήματα σεναρίου:

1.Ορισμός του status της παραγγελίας.

2.Προσθήκη σχολίου που αφορά την παραγγελία.

3.Αλλαγή στοιχείων πελάτη.

4.Υποβολή αλλαγών.

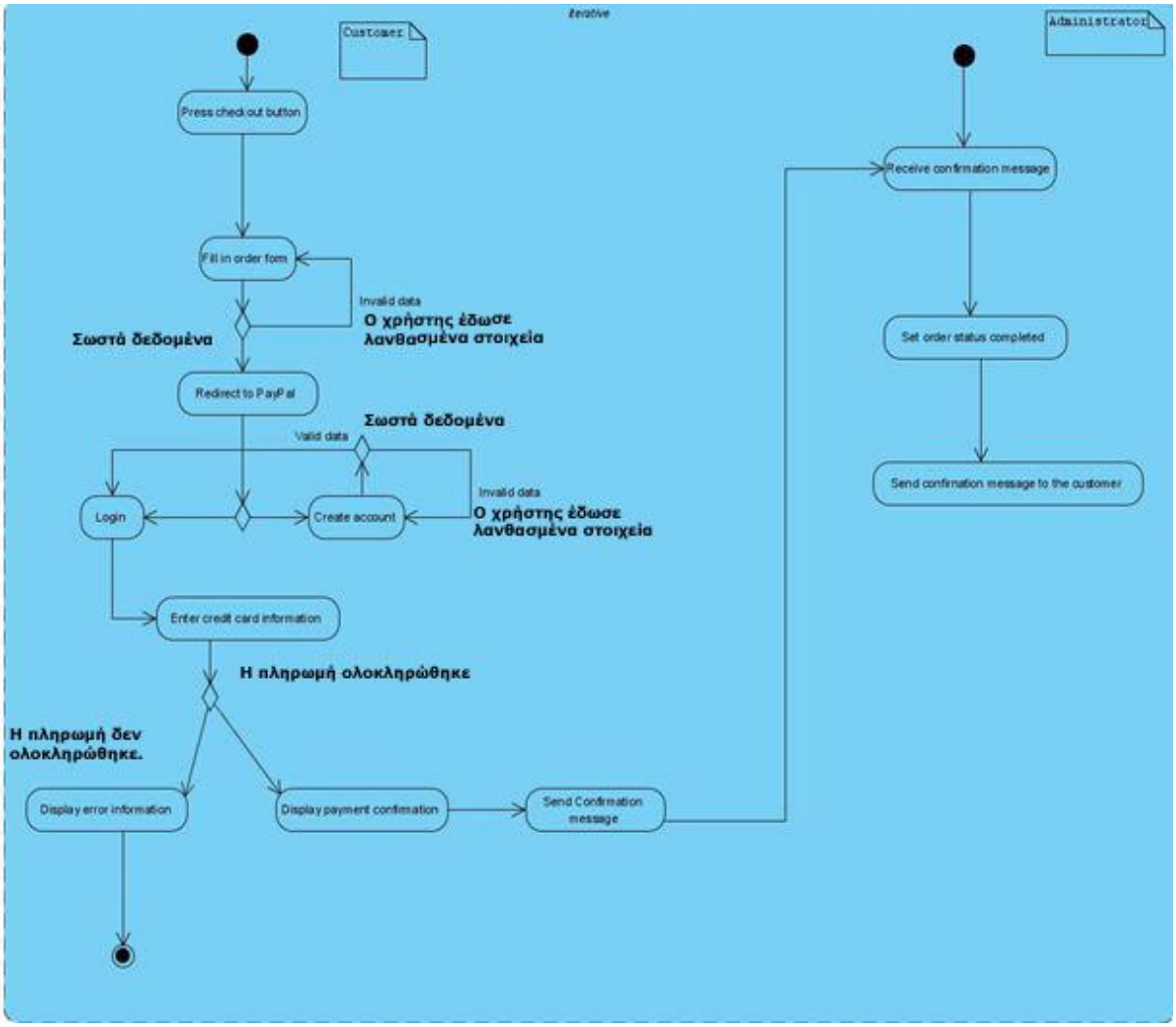
Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Ορισμός του status της παραγγελίας.

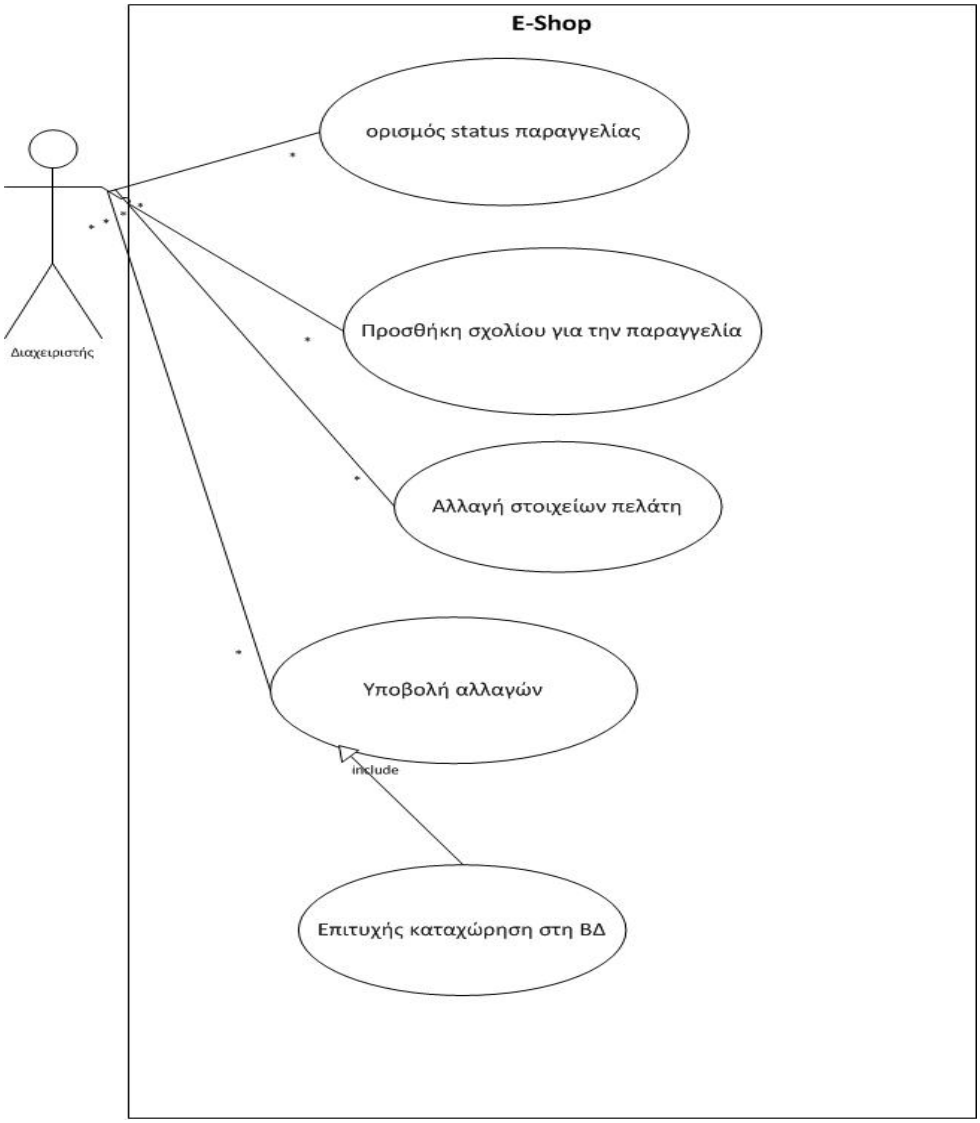
Ο διαχειριστής μπορεί να ορίσει tostatusτης παραγγελίας είτε πατώντας το κατάλληλο κουμπί, είτε κάνοντας κλικ στο κατάλληλο checkbox. Αν το status οριστεί ως“Completed”, τότε θα εμφανίζεται μήνυμα που θα τον ενημερώνει ότι θα αποσταλεί e-mail στον πελάτη το όποιο θα τον ειδοποιεί ότι η διαδικασία αποστολής των προϊόντων που έχει αγοράσει έχει ξεκινήσει. Ο διαχειριστής μπορεί να επιβεβαιώσει tostatusτης παραγγελίας ή να το ακυρώσει.

Βήματα 2-4: Προσθήκη σχολίου, αλλαγή στοιχείων πελάτη και υποβολή αλλαγών.

Ο διαχειριστής μπορεί να εισάγει κάποιο σχόλιο για την παραγγελία, καθώς και να αλλάξει τα στοιχεία του πελάτη αν έχει λάβει σχετική ενημέρωση απ’ αυτόν. Στη συνέχεια, πατώντας το κουμπί “Update” οι αλλαγές θα καταχωρούνται στη βάση δεδομένων.



Εικόνα 62. Activity Diagram – Διαδικασία αποστολής confirmation message



Εικόνα 63. Διάγραμμα περίπτωσης χρήσης επεξεργασίας στοιχείων παραγγελίας

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.15 Σενάριο: Αναζήτηση παραγγελιών βάσει κριτηρίων

Ο διαχειριστής μπορεί να πραγματοποιήσει αναζήτηση παραγγελιών επιλέγοντας έναν από τους τέσσερις διαθέσιμους τρόπους.

Βήματα σεναρίου:

1. Επιλογή τρόπου αναζήτησης.

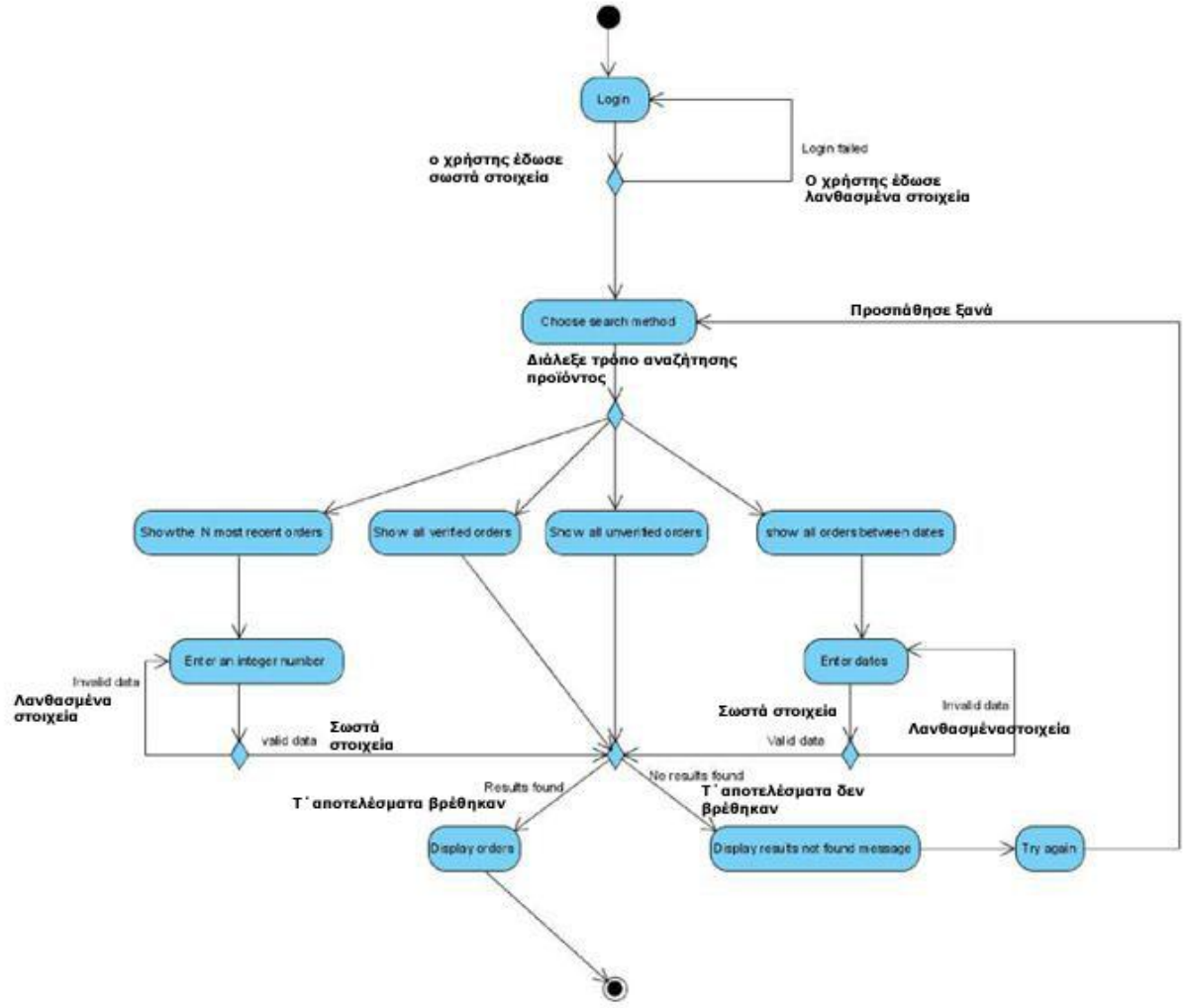
2. Εμφάνιση αποτελεσμάτων.

Αναλυτική περιγραφή βημάτων σεναρίου

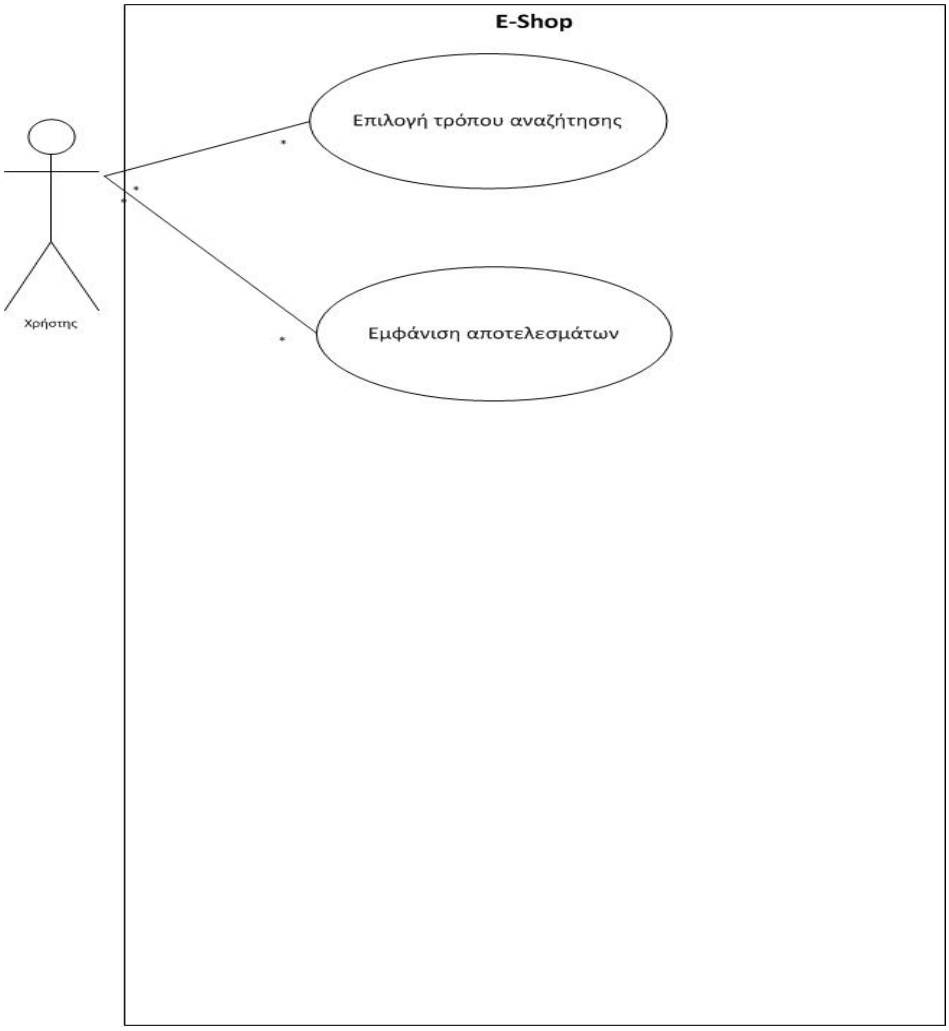
Βήμα 1: Επιλογή τρόπου αναζήτησης.

Ο διαχειριστής επιλέγει έναν από τους 4 διαθέσιμους τρόπους αναζήτησης παραγγελίας.

Βήμα 2: Αν βρεθούν παραγγελίες που να ανταποκρίνονται στα κριτήρια αναζήτησης, θα εμφανίζονται σε πίνακα που θα περιέχει συνοπτικές πληροφορίες για κάθε παραγγελία.



Εικόνα 64. Activity Diagram - Αναζήτηση παραγγελιών



Εικόνα 65. Αναζήτηση παραγγελιών βάσει κριτηρίων

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.16 Σενάριο: Διαγραφή παραγγελίας

Ο διαχειριστής μπορεί να διαγράψει μια παραγγελία αν το κρίνει απαραίτητο. Για τη διαγραφή μιας παραγγελίας πρέπει να εκτελεστούν τα παρακάτω βήματα. Βήματα σεναρίου:

1. Διαγραφή παραγγελίας.

2. Επιβεβαίωση διαγραφής.

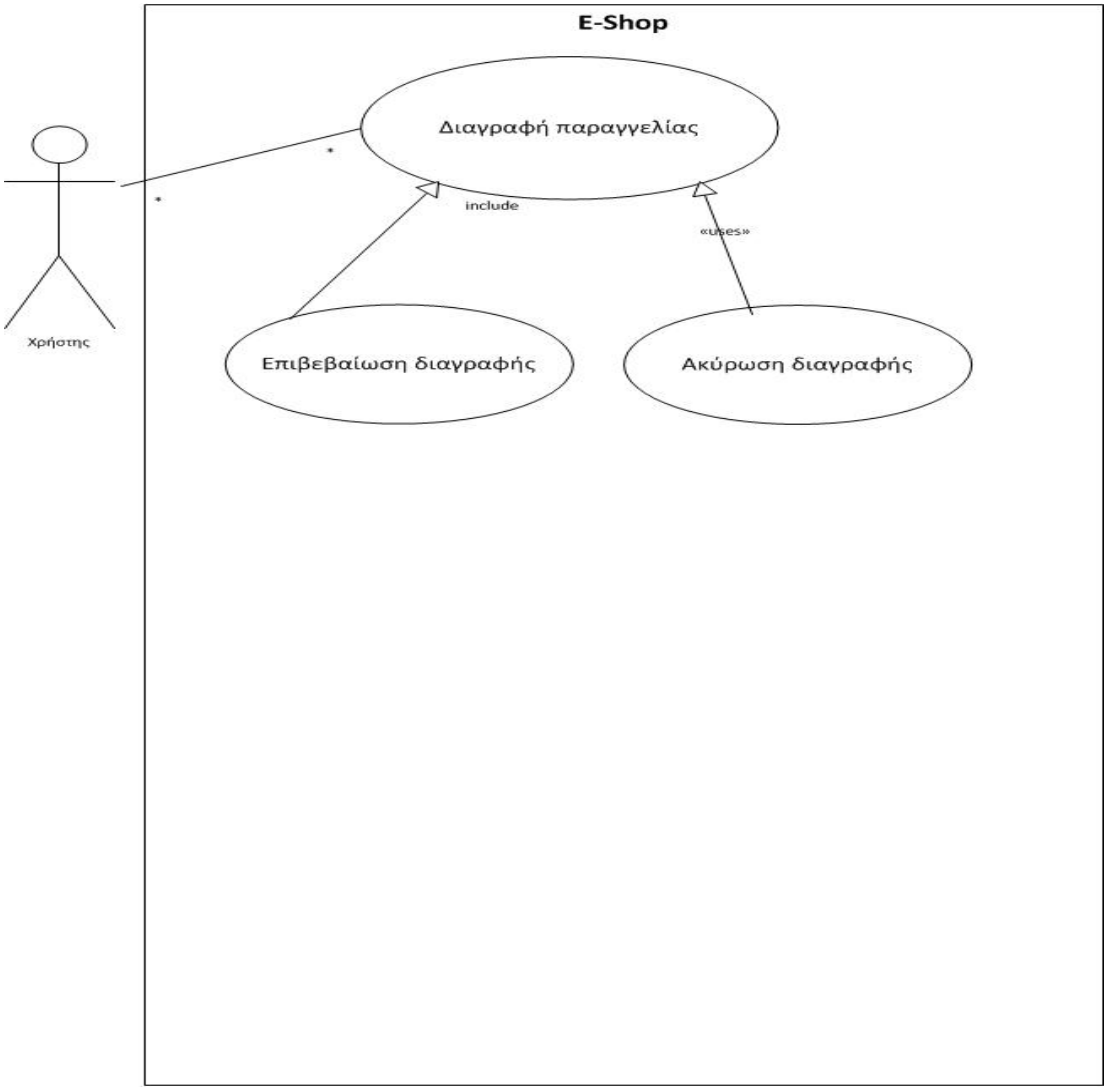
Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Διαγραφή παραγγελίας

Ο διαχειριστής, αφού επιλέξει τη παραγγελία που επιθυμεί να διαγράψει πατάει το κουμπί “Delete”.

Βήμα 2: Επιβεβαίωση διαγραφής

Το σύστημα ζητάει την επιβεβαίωση, της διαγραφής, της παραγγελίας. Ο διαχειριστής μπορεί είτε να επιβεβαιώσει τη διαγραφή είτε να την ακυρώσει.



Εικόνα 66. Διάγραμμα περίπτωσης χρήσης διαγραφής παραγγελίας

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.17 Σενάριο: Διαχείριση καλαθιού αγορών

Ο πελάτης όποτε το θελήσει μπορεί να προσθέσει και να αφαιρέσει προϊόντα από το καλάθι αγορών του καθώς και να ανανεώσει την ποσότητα κάποιου προϊόντος που βρίσκεται ήδη στο καλάθι.

Βήματα σεναρίου:

1. Προσθήκη προϊόντος στο καλάθι αγορών.

2. Αφαίρεση προϊόντος από το καλάθι αγορών.

3. Ανανέωση ποσότητας προϊόντος αγορών.

4. Έναρξη οικονομικής εκκαθάρισης (διαδικασίας πληρωμής).

Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Προσθήκη προϊόντος στο καλάθι αγορών.

Ο πελάτης για να προσθέσει ένα προϊόν στο καλάθι, το επιλέγει και στη συνέχεια κάνει κλικ στο κατάλληλο εικονίδιο. Το προϊόν και το συνολικό κόστος θα εμφανίζονται άμεσα στο καλάθι χρήστη.

Βήμα 2: Αφαίρεση προϊόντος από το καλάθι αγορών.

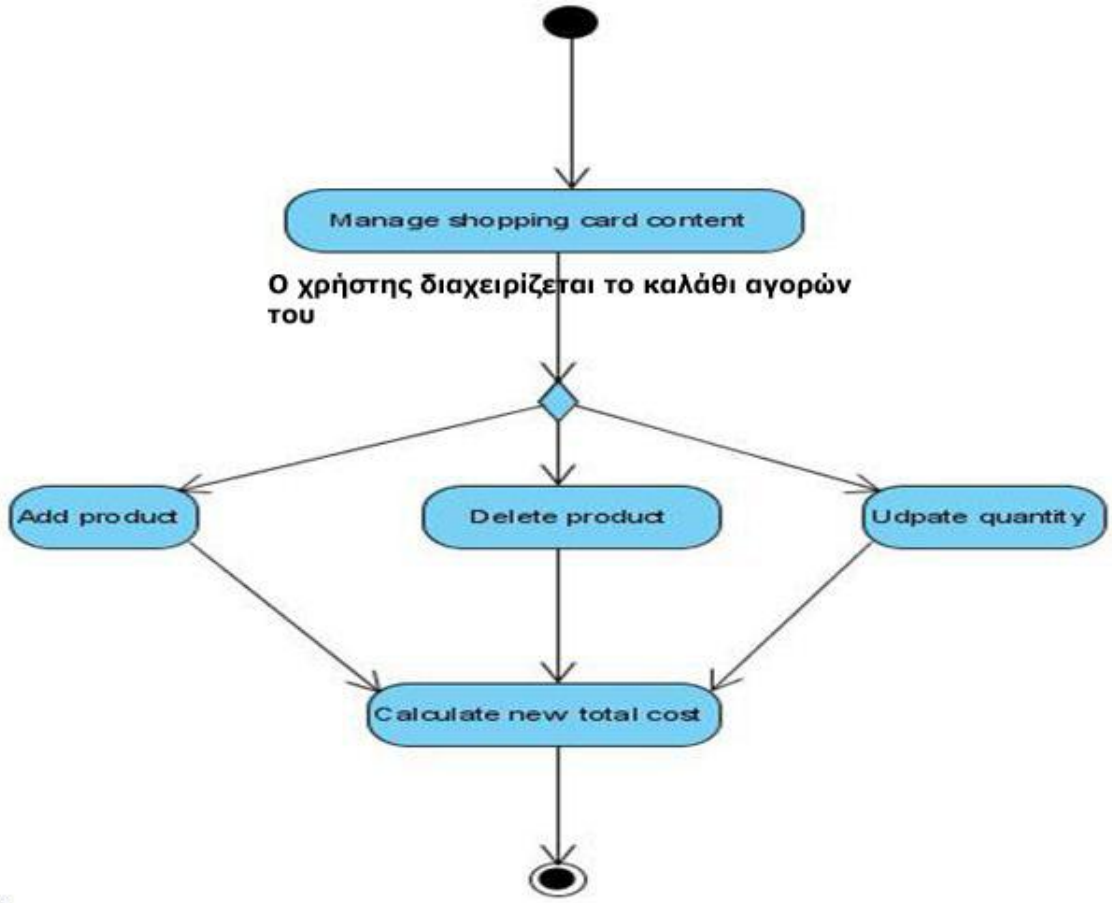
Ο πελάτης μπορεί να αφαιρέσει ένα προϊόν από το καλάθι κάνοντας κλικ στο κουμπί Delete. Το συνολικό κόστος της παραγγελίας θα υπολογίζεται και θα εμφανίζεται εκ νέου. Σε περίπτωση που όλα τα προϊόντα αφαιρεθούν από το καλάθι τα κουμπιά “Proceed to Checkout” και “Update Quantities” θα απενεργοποιούνται.

Βήμα 3: Ανανέωση ποσότητας προϊόντος αγορών.

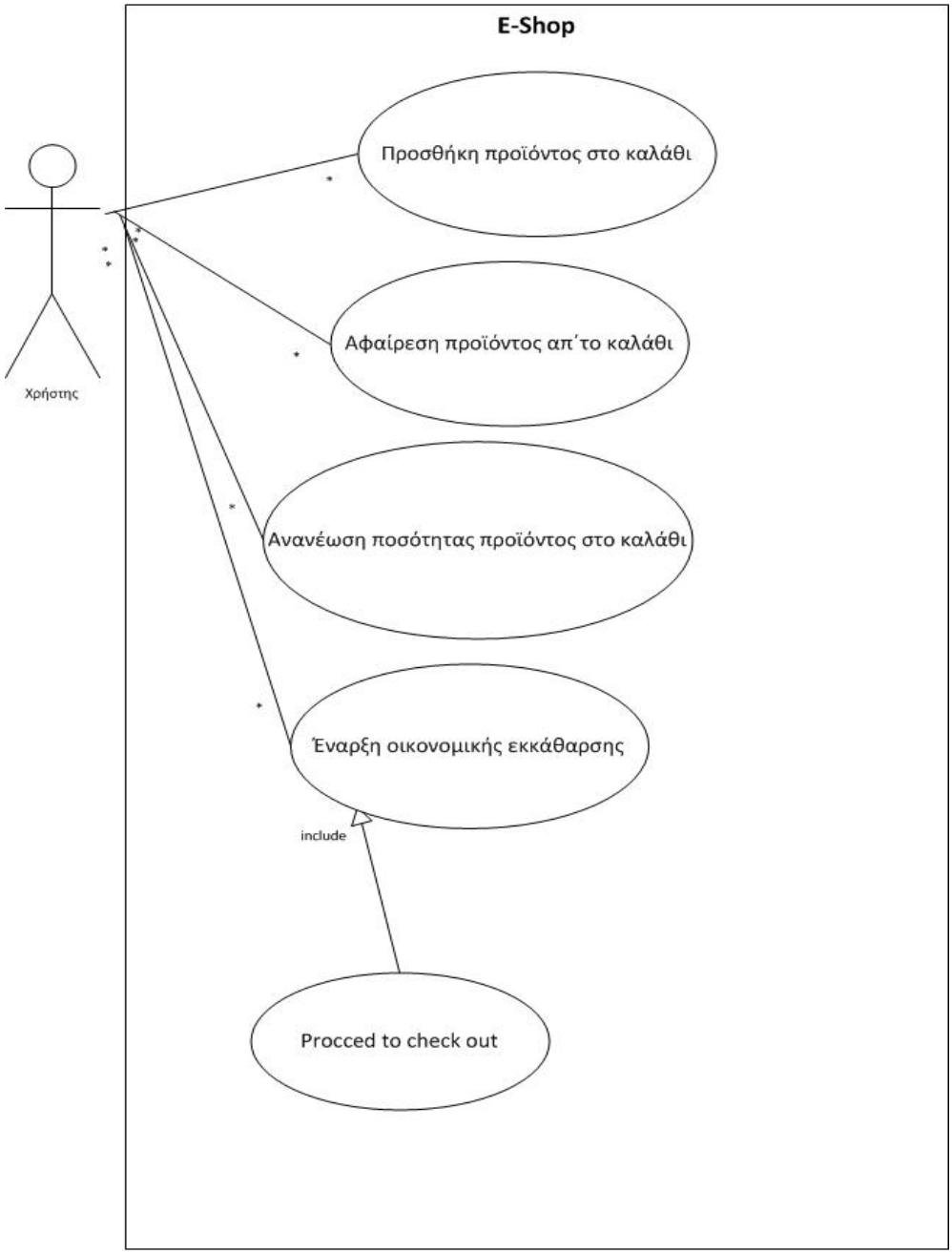
Ο πελάτης για να αλλάξει την ποσότητα κάποιου προϊόντος, εισάγει τη νέα ποσότητα στο κατάλληλο textbox και πατάει το κουμπί “Update Quantities”. Το συνολικό κόστος της παραγγελίας θα υπολογίζεται και θα εμφανίζεται εκ νέου. Σε περίπτωση που ο χρήστης εισάγει μηδενική ποσότητα σε όλα τα προϊόντα, τότε αυτά θα αφαιρούνται από το καλάθι και τα κουμπιά “Proceed to Checkout” και “Update Quantities “θα απενεργοποιούνται“

Βήμα 4: Έναρξη διαδικασίας πληρωμής.

Ο πελάτης μπορεί να ξεκινήσει την οικονομική εκκαθάριση (πληρωμή προϊόντων) πατώντας το κουμπί “Proceed to Checkout”.



Εικόνα 67. Activity Diagram - Διαχείριση καλαθιού αγοράς



Εικόνα 68. Διάγραμμα περίπτωσης χρήσης διαχείρισης καλαθιού αγοράς

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.18 Σενάριο: Οικονομική εκκαθάριση (πληρωμή προϊόντων)

Ο πελάτης, αφού καταλήξει στα προϊόντα που επιθυμεί να αγοράσει, συμπληρώνει τη φόρμα υποβολής παραγγελίας και κατευθύνεται στο ηλεκτρονικό σύστημα πληρωμών Paypal όπου εξοφλεί τη παραγγελία του.

Βήματα σεναρίου:

1. Συμπλήρωση φόρμας υποβολής παραγγελίας.
2. Υποβολή φόρμας.
3. Ανακατεύθυνση στο ηλεκτρονικό σύστημα πληρωμών Paypal.
4. Εξόφληση παραγγελίας.

Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Συμπλήρωση φόρμας υποβολής παραγγελίας

Ο πελάτης καλείται να εισάγει τα στοιχεία του (ονοματεπώνυμο, διεύθυνση αποστολής, e-mailκλπ.) .

Βήμα 2: Υποβολή φόρμας.

Πατώντας το κουμπί Submit, τα στοιχεία ελέγχονται από το σύστημα. Στην περίπτωση που είναι έγκυρα, ο χρήστης κατευθύνεται στοPaypal. Στην αντίθετη περίπτωση, εμφανίζεται επεξηγηματικό μήνυμα που προτρέπει τον πελάτη να διορθώσει τα μη έγκυρα στοιχεία.

Βήμα 3 : Ανακατεύθυνση στο ηλεκτρονικό σύστημα πληρωμών PayPal.

Ο χρήστης ευρισκόμενος πλέον στο Paypal, καλείται να επιλέξει τον τρόπο εξόφλησης της παραγγελίας του.

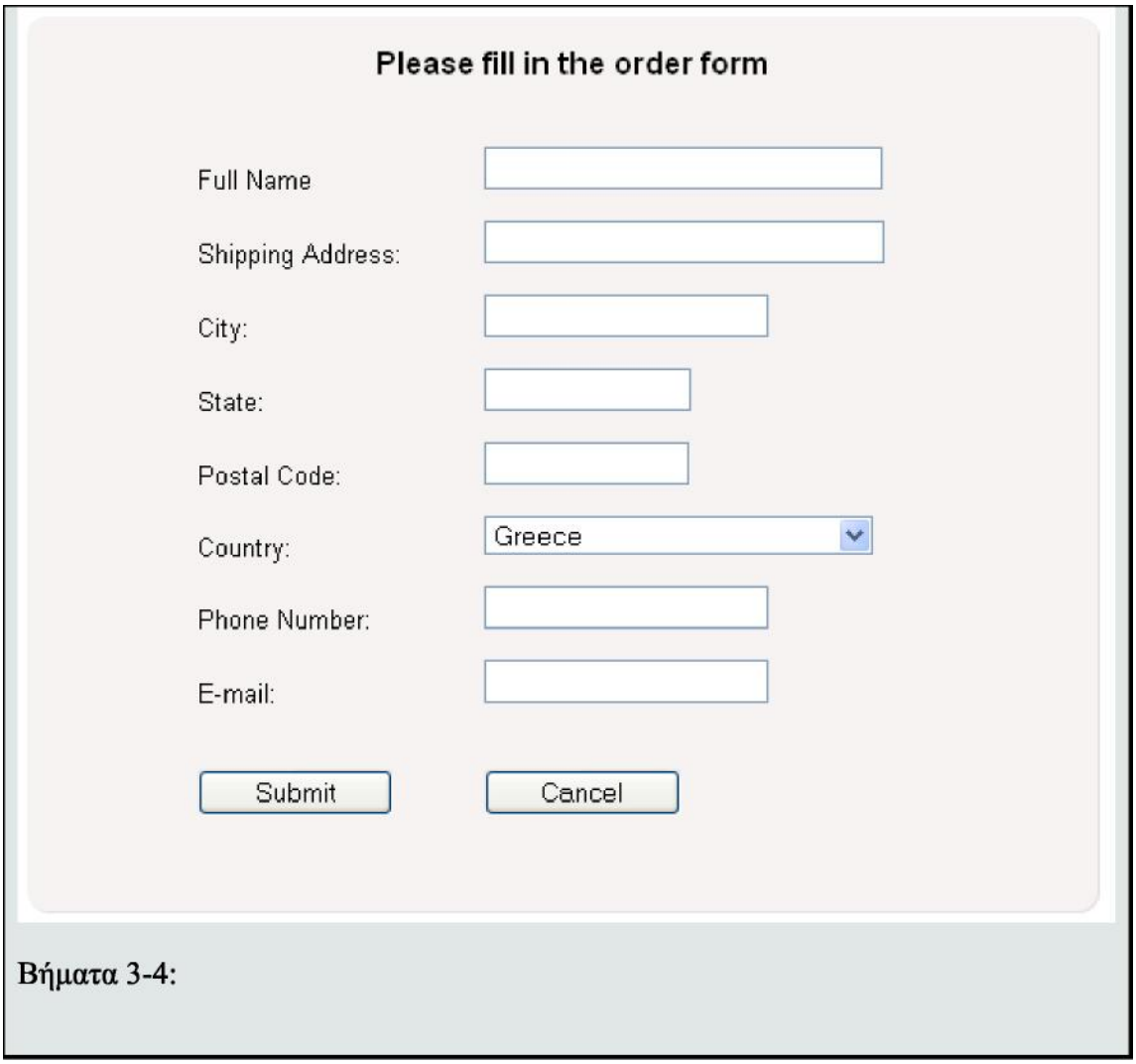
Οι διαθέσιμες επιλογές είναι:

1) Να δημιουργήσει έναν λογαριασμό ή να συνδεθεί στον ήδη υπάρχοντα.

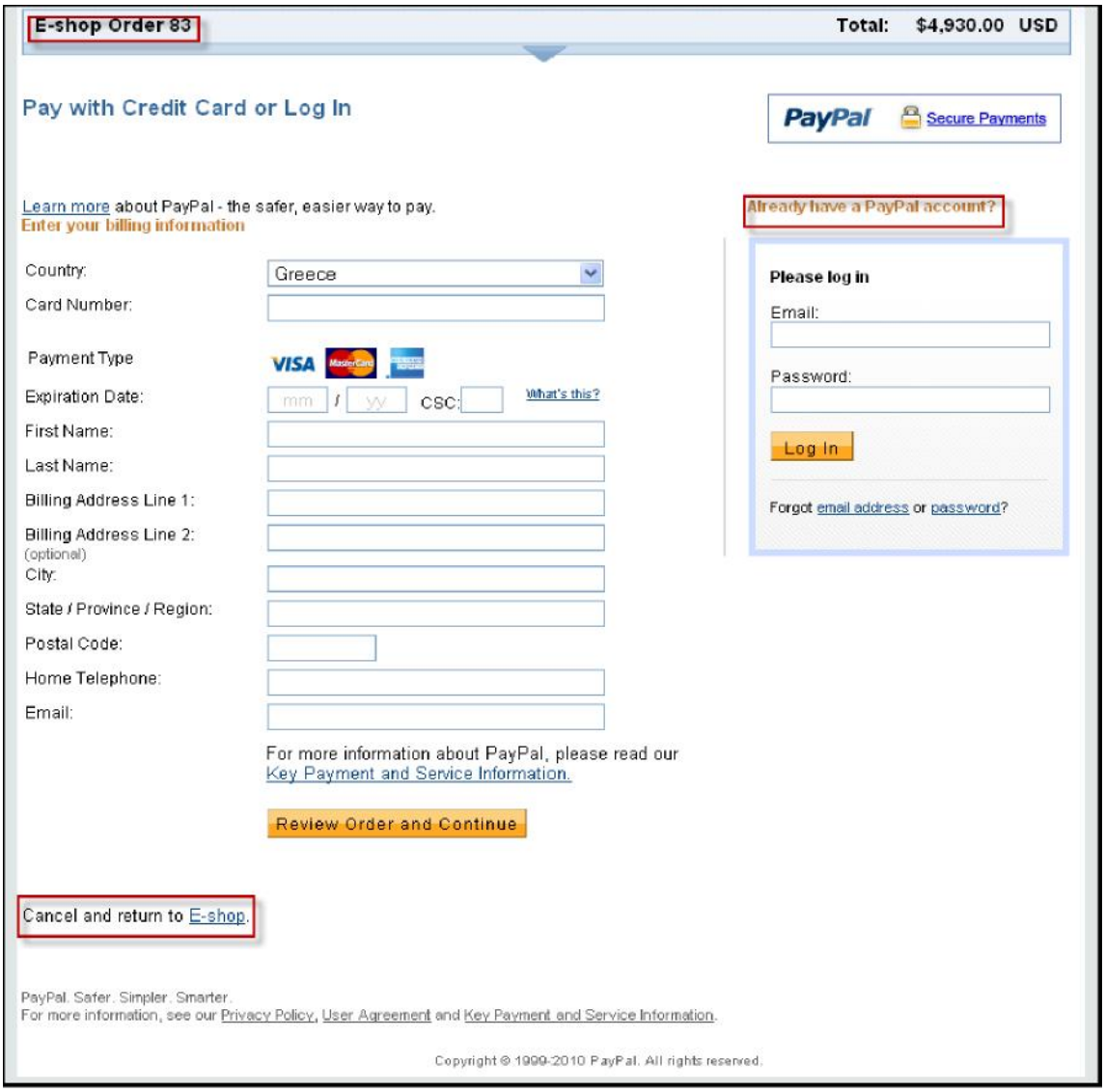
2) Να ολοκληρώσει τη διαδικασία ως επισκέπτη (guest).

Βήμα 4: Εξόφληση παραγγελίας.

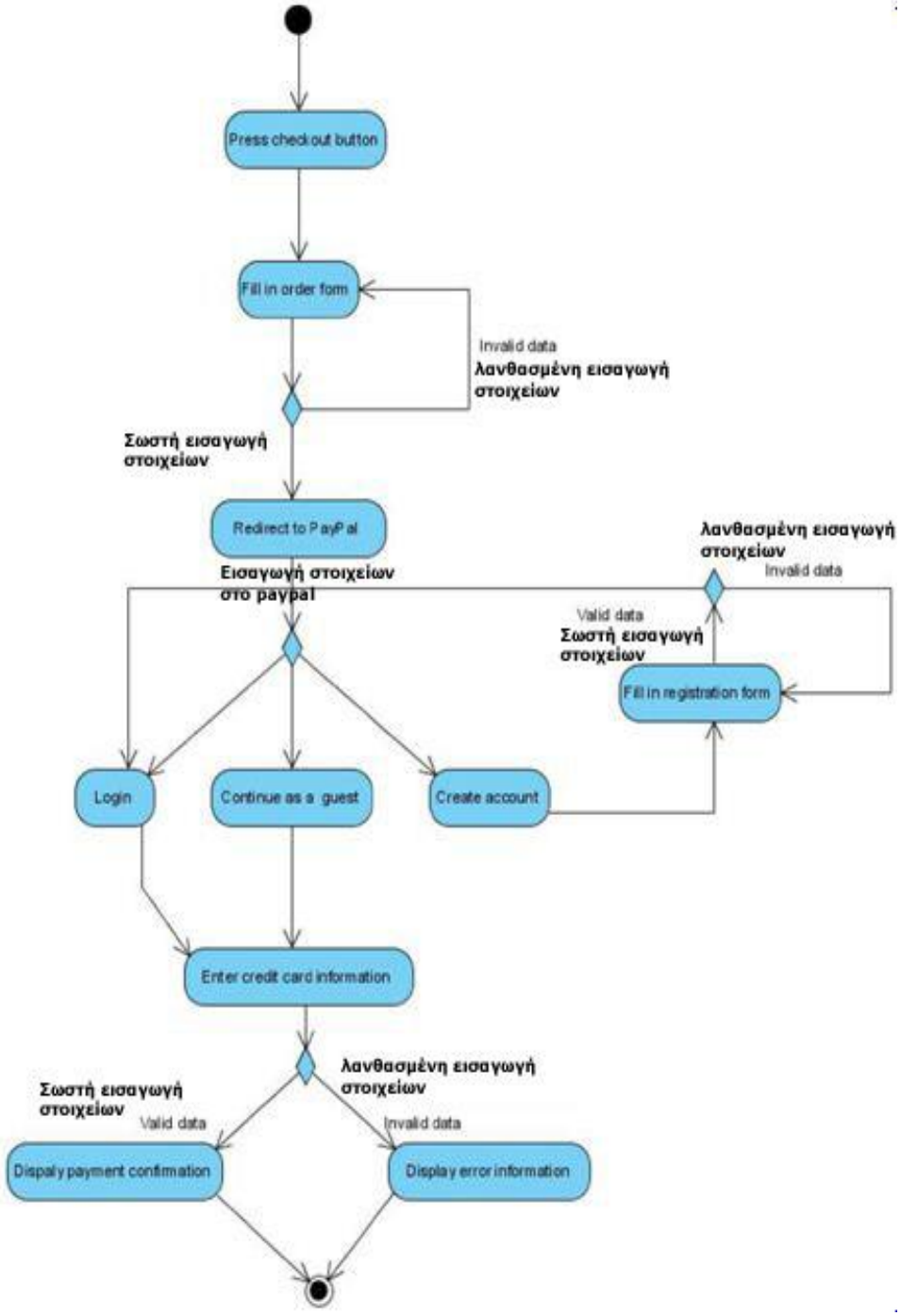
Ο χρήστης καλείται να εισάγει ξανά τα στοιχεία του καθώς και τον αριθμό της πιστωτικής του κάρτας. Τα στοιχεία ελέγχονται από τοPayPal. Στην περίπτωση που είναι έγκυρα, ο πελάτης ενημερώνεται για την επιτυχή εξόφληση της παραγγελίας. Στην αντίθετη περίπτωση, εμφανίζεται επεξηγηματικό μήνυμα που προτρέπει τον πελάτη να διορθώσει τα μη έγκυρα στοιχεία.



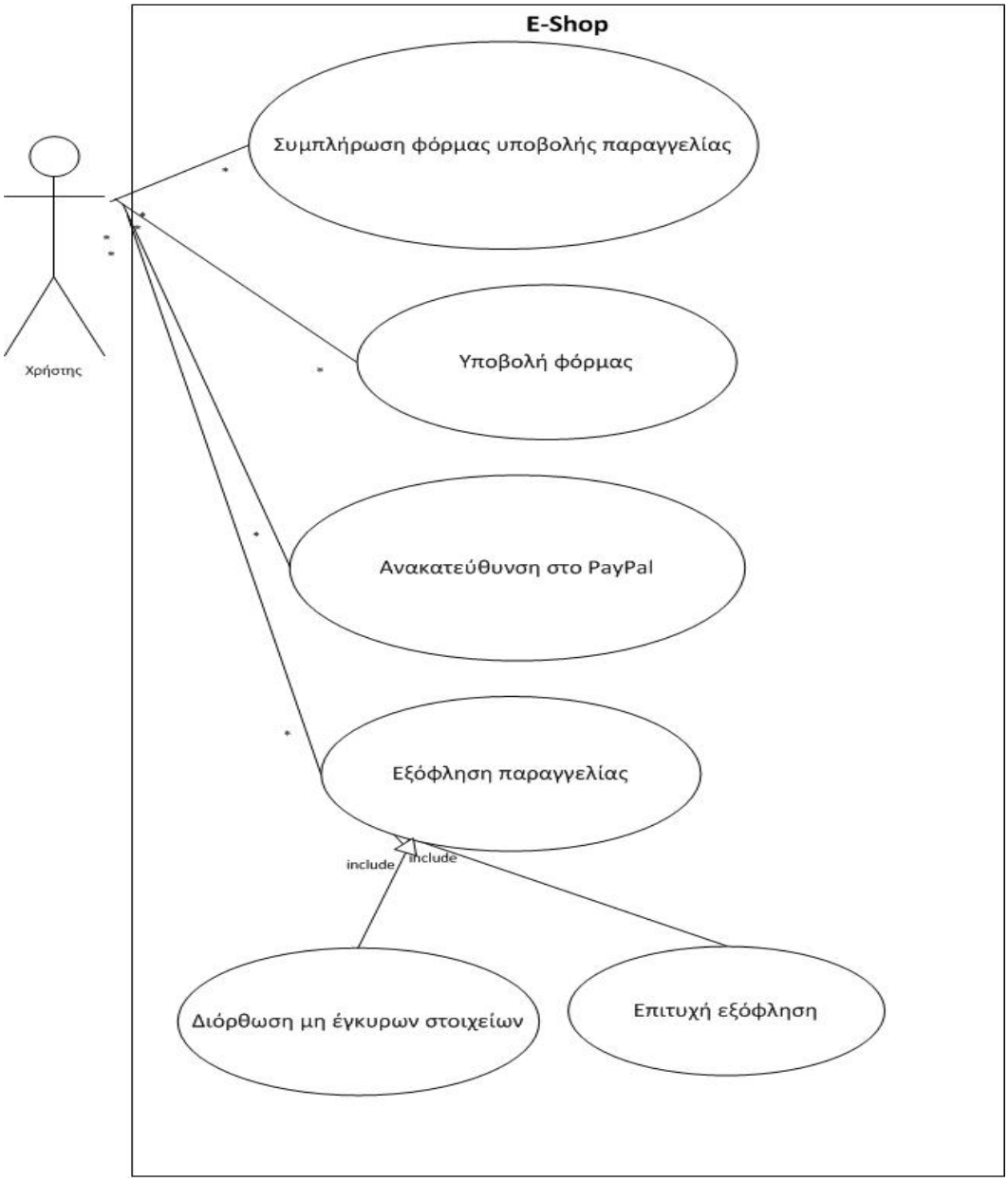
Εικόνα 69. Σχηματική απεικόνιση σεναρίου



Εικόνα 70. Σχηματική απεικόνιση σεναρίου



Εικόνα 71. Activity Diagram - Οικονομική εκκαθάριση



Εικόνα 72. Διάγραμμα περίπτωσης χρήσης οικονομικής εκκάθαρσης

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.19 Σενάριο: Καταμέτρηση ή διαγραφή καλαθιών αγορών

Ο διαχειριστής μπορεί να δει τον αριθμό των καλαθιών αγοράς που έχουν δημιουργηθεί σε μία συγκεκριμένη χρονική περίοδο που επιλέγει ο ίδιος και αν το κρίνει απαραίτητο να τα διαγράψει, ώστε να μη δεσμεύεται άσκοπος χώρος στη βάση δεδομένων.

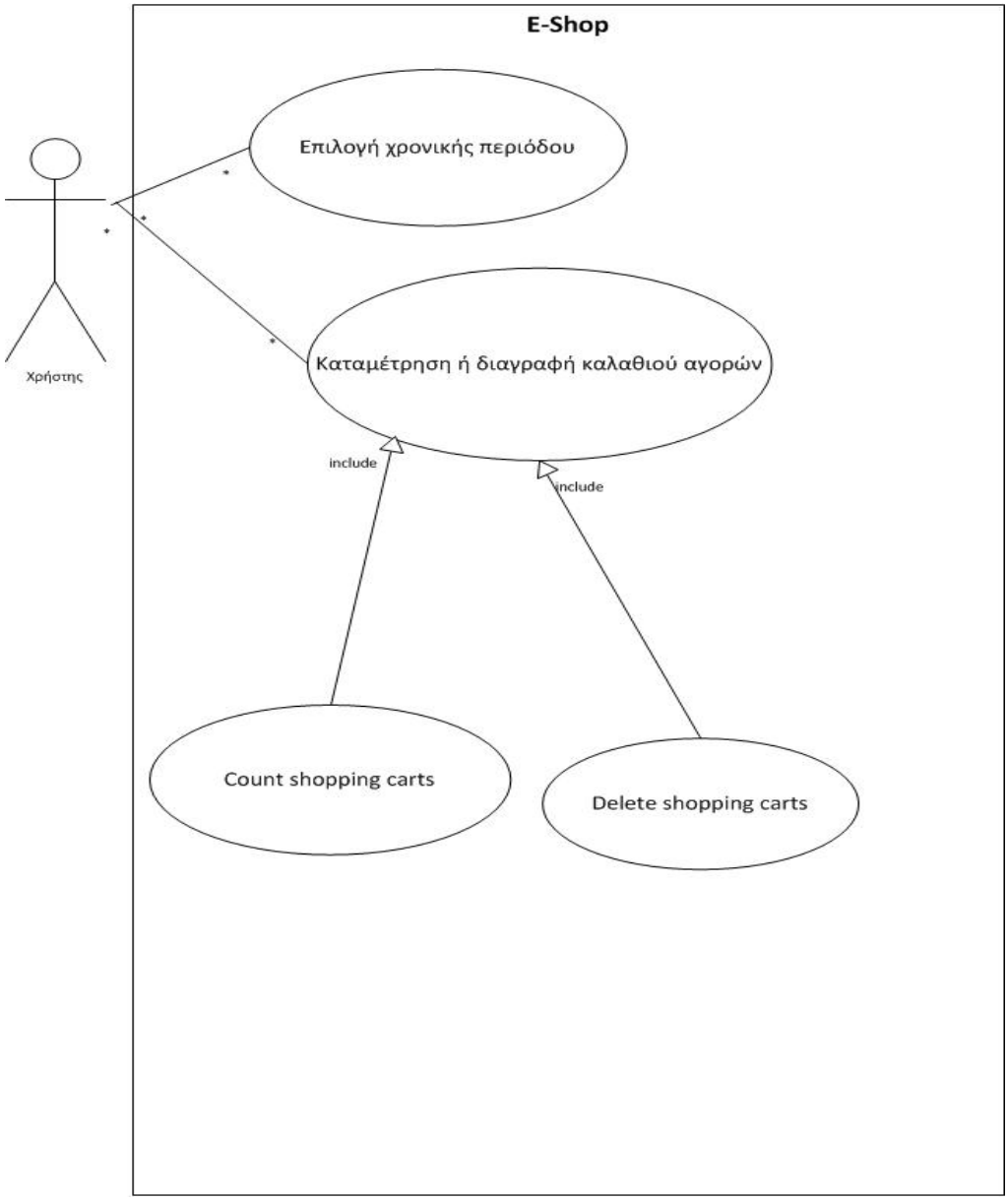
Βήματα σεναρίου:

- Επιλογή χρονικής περιόδου.
- Καταμέτρηση ή διαγραφή ανενεργών καλαθιών αγορών.

Αναλυτική περιγραφή βημάτων σεναρίου

Βήματα 1-2: Καταμέτρηση ή διαγραφή ανενεργών καλαθιών αγοράς ανά επιλεγμένη χρονική περίοδο.

Ο διαχειριστής επιλέγει μέσω του drop-down list έναν αριθμό ημερών και στη συνέχεια είτε μετράει τα καλάθια αγορών πατώντας το κουμπί “Count shopping carts”, είτε τα διαγράφει πατώντας το κουμπί “Delete shopping carts”.



Εικόνα 73. Διάγραμμα περίπτωσης χρήσης καταμέτρησης ή διαγραφής καλαθιών αγοράς

Περίπτωση χρήσης: Διαχείριση παραγγελιών

6.9.3.1.20 Σενάριο: Επικοινωνία με τον διαχειριστή του ηλεκτρονικού καταστήματος.

Ο χρήστης μπορεί να υποβάλει ένα ερώτημα ή ένα σχόλιο στον διαχειριστή του ηλεκτρονικού καταστήματος συμπληρώνοντας τη κατάλληλη φόρμα.

Βήματα σεναρίου:

1.Συμπλήρωση φόρμας επικοινωνίας

2.Αποστολή μηνύματος.

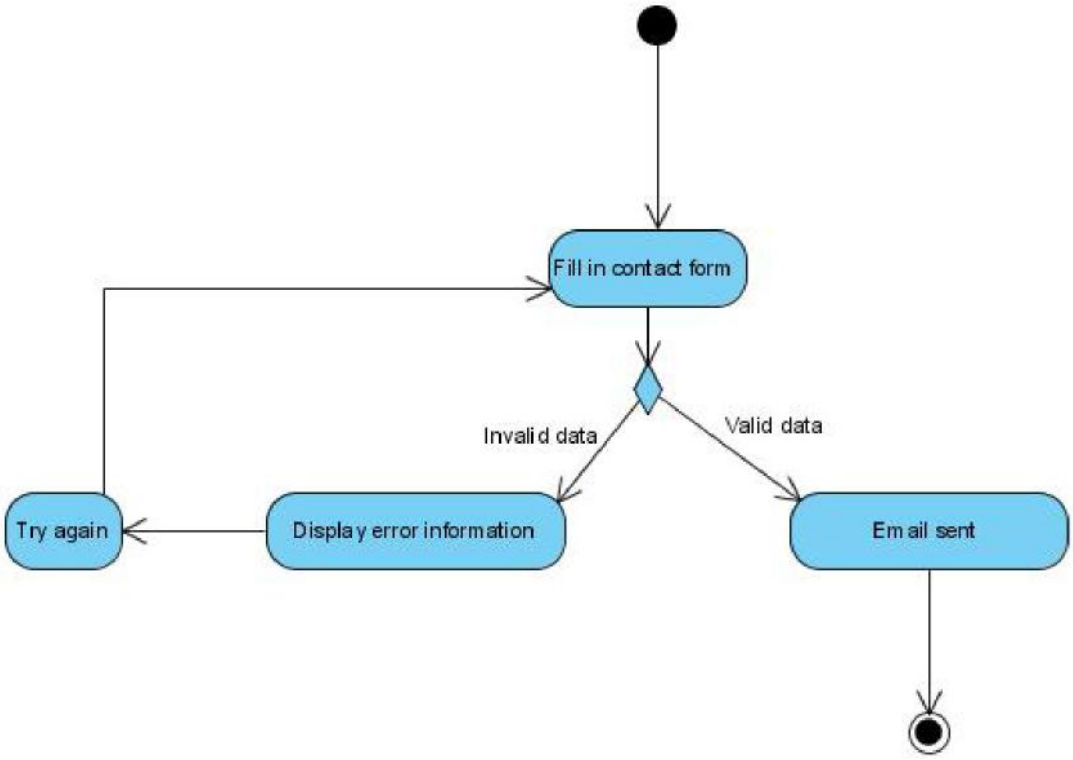
Αναλυτική περιγραφή βημάτων σεναρίου

Βήμα 1: Συμπλήρωση φόρμας επικοινωνίας

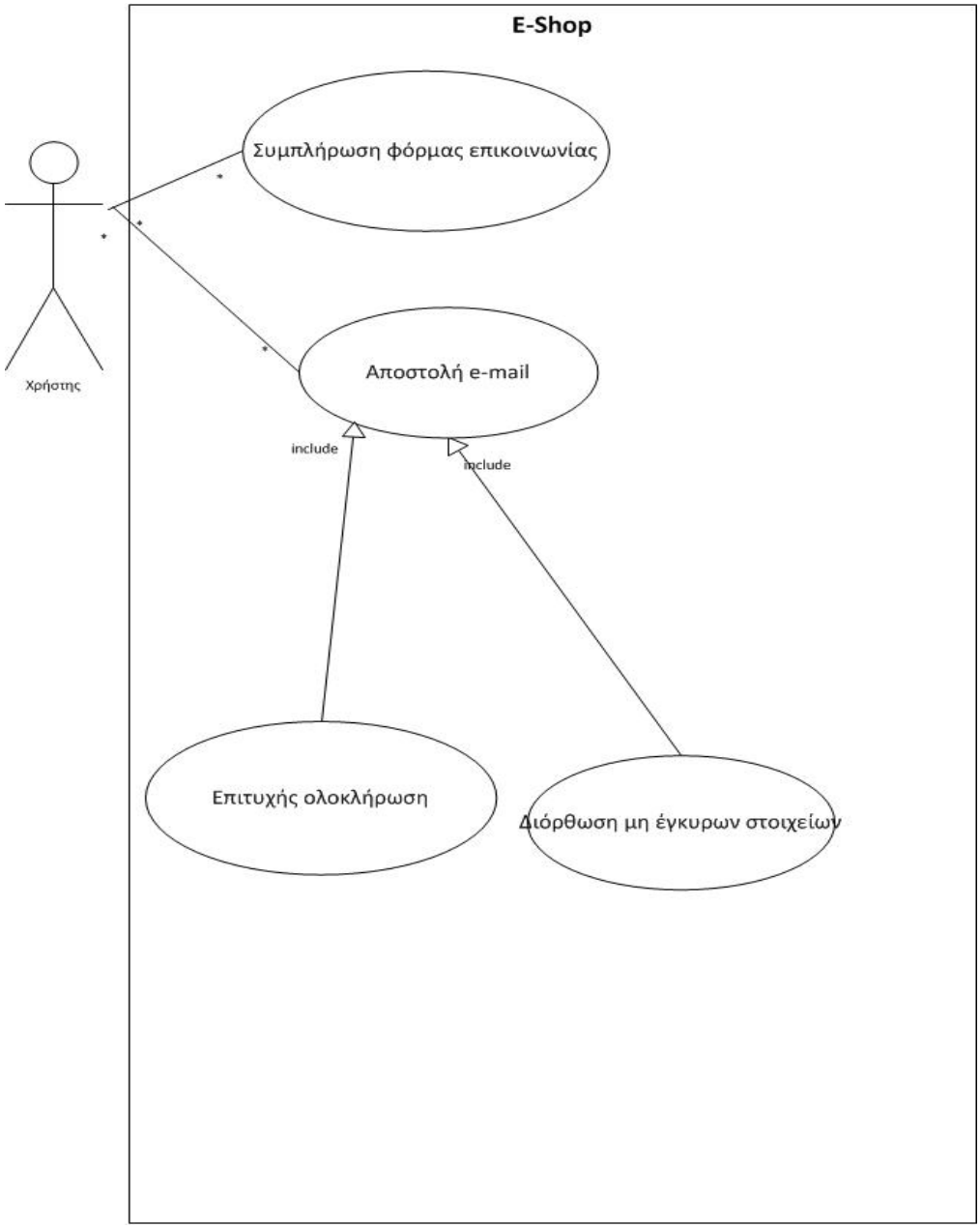
Ο χρήστης εισάγει τα στοιχεία του (ονοματεπώνυμο, τηλέφωνο, email) και το περιεχόμενο του μηνύματος.

Βήμα 2: Αποστολή μηνύματος.

Πατώντας το κουμπί “Submit”, τα στοιχεία ελέγχονται από το σύστημα. Σε περίπτωση που είναι έγκυρα ,το μήνυμα στέλνεται στον διαχειριστή και ο πελάτης ενημερώνεται για την επιτυχή ολοκλήρωση της διαδικασίας. Σε αντίθετη περίπτωση εμφανίζεται επεξηγηματικό μήνυμα που δείχνει στον πελάτη ότι πρέπει να διορθώσει τα μη έγκυρα στοιχεία.



Εικόνα 74. Activity Diagram – Αποστολή ηλεκτρονικού μηνύματος



Εικόνα 75. Διάγραμμα περίπτωσης χρήσης αποστολής ηλεκτρονικού ταχυδρομείου

6.9.4 Υλοποίηση συστήματος

Για την υλοποίηση του συστήματος επιλέχθηκε η αρχιτεκτονική τριών επιπέδων (three-tier architecture). Η αρχιτεκτονική των τριών επιπέδων, είναι μια αρχιτεκτονική client-server, στην οποία, η διεπαφή χρήστη (user interface), η επιχειρηματική λογική (business logic) και η αποθήκευση και η πρόσβαση δεδομένων αναπτύσσονται και συντηρούνται ως ξεχωριστές ενότητες ή πιο συχνά σε ξεχωριστές πλατφόρμες.

Η αρχιτεκτονική αυτή αποτελείται από τα εξής επίπεδα:

- Το επίπεδο παρουσίασης (presentation tier): Είναι το αρχικό επίπεδο της εφαρμογής και αυτό με το οποίο αλληλεπιδρά ο τελικός χρήστης μέσω των στοιχείων διαπαλών χρήστη (user interfaces). Το επίπεδο αυτό περιλαμβάνει μηχανισμούς ελέγχου των στοιχείων που εισάγει ο χρήστης.

- Το επιχειρηματικό επίπεδο (business tier): Το επίπεδο αυτό συντονίζει την εφαρμογή, επεξεργάζεται εντολές, αξιολογεί τα δεδομένα που δέχεται από το επίπεδο παρουσίασης και εκτελεί υπολογισμούς. Δέχεται αιτήσεις από το επίπεδο παρουσίασης και καλεί το επίπεδο δεδομένων ώστε να ανακτήσει πληροφορίες τις οποίες θα στείλει πίσω στο επίπεδο παρουσίασης.

- Το επίπεδο δεδομένων (data tier): Στο επίπεδο αυτό τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων ή σε έναν φάκελο του συστήματος. Οπως έχει ήδη αναφερθεί, η ζητούμενη πληροφορία ανακτάται από αυτό το επίπεδο και μεταβιβάζεται στο επιχειρηματικό επίπεδο το οποίο με τη σειρά του την προωθεί στο επίπεδο παρουσίασης.

Τα κυριότερα πλεονεκτήματα που προσφέρει η αρχιτεκτονική τριών επιπέδων σε σχέση με μια κλασική εφαρμογή client-serverδύο επιπέδων, είναι τα εξής:

- Συντηρητικότητα: Κάθε επίπεδο είναι ανεξάρτητο από τα υπόλοιπα με αποτέλεσμα οι ενημερώσεις και οι αλλαγές να μπορούν να πραγματοποιηθούν χωρίς να επηρεάζουν την εφαρμογή στο σύνολό της.

- Επεκτασιμότητα: Η δομή της αρχιτεκτονικής αυτής καθιστά εύκολη την αναβάθμιση της εφαρμογής, αφού κάθε επίπεδο αναπτύσσεται και συντηρείται ως ξεχωριστή ενότητα.

- Ευελιξία: Αυξάνεται η ευελιξία, επειδή κάθε επίπεδο διαχειρίζεται και αναπτύσσεται ανεξάρτητα από τα υπόλοιπα.

Παράδειγμα χρήσης της αρχιτεκτονικής τριών επιπέδων

Η δομή και η λειτουργία της αρχιτεκτονικής τριών επιπέδων μπορεί εύκολα να γίνει κατανοητή με την περιγραφή ενός απλού σεναρίου χρήσης του συστήματος. Το σύστημα επιτρέπει στον χρήστη να προσθέσει κάποιο προϊόν στο καλάθι αγορών πατώντας το κουμπί “Add to cart”.

Στο Βήμα 1, ο χρήστης κάνει κλικ στο κουμπί “Add to cart” για κάποιο συγκεκριμένο προϊόν. Στο Βήμα 2, το επίπεδο παρουσίασης (το οποίο περιέχει το κουμπί) προωθεί το αίτημα στο επίπεδο λογικής: *“Πρόσθεσε αυτό το προϊόν στο καλάθι αγορών”*. Στο Βήμα 3, το επιχειρηματικό επίπεδο δέχεται το αίτημα, αντιλαμβάνεται ότι ο χρήστης θέλει να προσθέσει το συγκεκριμένο προϊόν στο καλάθι αγορών, και χειρίζεται το αίτημα λέγοντας στο επίπεδο δεδομένων να ενημερώσει το καλάθι αγορών του χρήστη προσθέτοντας το επιλεγμένο προϊόν. Το επίπεδο δεδομένων πρέπει να κληθεί καθώς αποθηκεύει και διαχειρίζεται όλα τα δεδομένα της εφαρμογής, περιλαμβανομένων των πληροφοριών για το καλάθι αγοράς του χρήστη.

Στο Βήμα 4, το επίπεδο δεδομένων ενημερώνει τη βάση δεδομένων και επιστρέφει έναν κώδικα που υποδεικνύει την επιτυχή ολοκλήρωση της διαδικασίας στο επιχειρηματικό επίπεδο. Στο Βήμα 5, το επιχειρηματικό επίπεδο διαχειρίζεται τον επιστρεφόμενο κώδικα καθώς και πιθανά λάθη που μπορεί να έχουν συμβεί στο επίπεδο δεδομένων κατά την ενημέρωση της βάσης δεδομένων και επιστρέφει τα δεδομένα στο επίπεδο παρουσίασης. Στο Βήμα 6, το επίπεδο παρουσίασης παράγει μια ενημερωμένη προβολή του καλαθιού αγορών. Στο τελευταίο Βήμα, τα αποτελέσματα της εκτέλεσης εμφανίζονται δημιουργώντας μια Hypertext Markup Language (HTML) ιστοσελίδα στην οποία ο χρήστης μπορεί να δει το ανανεωμένο καλάθι αγορών μέσω του web browser που χρησιμοποιεί.

Να σημειωθεί ότι σε αυτό το απλό παράδειγμα το επιχειρηματικό επίπεδο δεν επεξεργάζεται πολλά δεδομένα και η επιχειρηματική λογική (business logic) δεν είναι πολύπλοκη. Ωστόσο, κάθε φορά που προστίθενται επιχειρηματικοί κανόνες (business rules) το επιχειρηματικό επίπεδο πρέπει να αναβαθμίζεται. Εάν, για παράδειγμα, η επιχειρηματική λογική καθορίζει ότι ένα προϊόν μπορεί να προστεθεί στο καλάθι χρήστη μόνο εάν η ποσότητά του στο stock είναι μεγαλύτερη από μηδέν, τότε θα έπρεπε να εκτελεστεί μια επιπλέον κλήση του επιχειρηματικού επιπέδου ώστε να προσδιοριστεί η ποσότητα. Τότε, το επίπεδο δεδομένων θα ζητούσε να ενημερωθεί το καλάθι αγορών μόνο αν υπάρχουν προϊόντα στο stock. Σε κάθε περίπτωση το επίπεδο παρουσίασης θα ενημερώνεται για το status του προϊόντος και θα παρέχει το κατάλληλο μήνυμα στον χρήστη.

6.9.5 Σχεδίαση και υλοποίηση της βάσης δεδομένων

Στη συνέχεια θα περιγραφεί η βάση δεδομένων του συστήματος καθώς η μορφή της καθορίζει σε μεγάλο βαθμό την συνολική οργάνωση του συστήματος. Η ανάπτυξη της βάσης δεδομένων έγινε με γνώμονα την απλότητα, την ευχρηστία, την πληρότητα αλλά και την έλλειψη επανάληψης πληροφορίας. Αρχικά, θα ερμηνευτεί η χρησιμότητα κάθε πίνακα και ακολούθως θα παρατεθεί το σχεσιακό μοντέλο στο οποίο θα φαίνονται όλοι οι πίνακες με τις επιμέρους ιδιότητές τους καθώς και οι σχέσεις μεταξύ των πινάκων.

Οι πίνακες της βάσης είναι οι εξής:

Πίνακας product

Πεδία:productID, productName, categoryID, manufacturerID, description, thumbnail, image, unitPrice, unitsInStock, RAM, HDD, screen, graphicCard, opticalDrive, wireless, sound, gates, OS, battery Length, weight, guarantee, front Page. Ο πίνακας product περιέχει τα χαρακτηριστικά κάθε προϊόντος. Τα πεδία categoryID και manufacturerID είναι εξωτερικά κλειδιά που συνδέουν τον παρόντα πίνακα με τους πίνακες category και manufacturer αντίστοιχα. Τα πεδία thumbnail και image υποδηλώνουν τα paths των φωτογραφιών που αντιπροσωπεύουν ένα προϊόν.

Τέλος, το πεδίο frontPage υποδηλώνει εάν το συγκεκριμένο προϊόν θα βρίσκεται στην αρχική σελίδα της εφαρμογής. Ως πρωτεύον κλειδί ορίζεται το πεδίο productID.

Πίνακας category

Πεδία:categoryID, categoryName, description.

Ο πίνακας category περιλαμβάνει όλες τις κατηγορίες των προϊόντων. Σημειώνεται ότι μπορεί να αλλάξει από τον διαχειριστή του συστήματος ο οποίος μπορεί να προσθέσει, να διαγράψει και να μεταβάλλει μια κατηγορία. Το πεδίο categoryName είναι το όνομα της κατηγορίας και το πεδίο description είναι μια σύντομη περιγραφή της κατηγορίας. Ως πρωτεύον κλειδί, ορίζεται το πεδίο categoryID.

Πίνακας manufacturer

Πεδία : manufacturerID, manufacturerName, description.

Ο πίνακας manufacturer περιλαμβάνει όλες τους κατασκευαστές προϊόντων και, όπως στον πίνακα category, μπορεί να αλλάξει από τον διαχειριστή του συστήματος ο οποίος μπορεί να προσθέσει, να διαγράψει και να μεταβάλλει ένα κατασκευαστή προϊόντων. Το πεδίο manufacturerName είναι το όνομα του κατασκευαστή και το πεδίο description είναι μια σύντομη περιγραφή της κατηγορίας. Ως πρωτεύον κλειδί ορίζεται το πεδίο manufacturerID.

Πίνακας shopping Cart

Πεδία:cartID, productID, quantity, dateAdded.

Ο πίνακας shopping Cartπεριέχει πληροφορίες για το συγκεκριμένο καλάθι χρήστη. Το πεδίοproductIDείναι ο κωδικός του προϊόντος που έχει προστεθεί στο καλάθι, το πεδίο quantityείναι η ποσότητα κάθε προϊόντος και το πεδίο dateAdded είναι η ημερομηνία προσθήκης στο καλάθι χρήστη. Ως σύνθετο πρωτεύον κλειδί ορίζονται τα πεδία cartID κα ιproductID.

Πίνακας order

Πεδία: orderID, customerID, dateCreated, dateShipped, verified, completed, canceled, comments.

Ο πίνακας orderπεριέχει όλες τις παραγγελίες προϊόντων που έχουν καταγραφεί. Το πεδίο orderID είναι ο κωδικός της παραγγελίας, το πεδίο customerID είναι ο κωδικός του πελάτη που πραγματοποίησε την παραγγελία, το πεδίο dateCreated είναι η ημερομηνία που καταγράφηκε η παραγγελία, και το πεδίο dateShipped είναι η ημερομηνία αποστολής των προϊόντων της παραγγελίας. Ο διαχειριστής της εφαρμογής μπορεί να θέσει tostatus της παραγγελίας μέσω των πεδίων verified, completed και canceled. Επίσης, μπορεί να προσθέσει κάποιο σχόλιο για την παραγγελία μέσω του πεδίου comments. Το πεδίοcustomerID είναι εξωτερικό κλειδί που συνδέει τον παρόντα πίνακα με τον πίνακα customer. Ως πρωτεύον κλειδί ορίζεται το πεδίο orderID.

Πίνακας orderDetail

Πεδία:orderID, productID, quantity, unitCost, subtotal.

Ο πίνακας orderDetail περιέχει πληροφορίες για τα προϊόντα που ανήκουν σε μια παραγγελία. Το πεδίο orderIDείναι εξωτερικό κλειδί που συνδέει τον παρόντα πίνακα με τον πίνακα order. Για κάθε προϊόν της παραγγελίας καταγράφεται ο κωδικός του (productID) , η ποσότητα (quantity), το κόστος μονάδας (unitCost) και το συνολικό κόστος του προϊόντος (subtotal). Εάν ένα προϊόν έχει παραγγελθεί kφορές, τότε το πεδίοquantityπαίρνει την τιμή k και το πεδίο subtotal υπολογίζεται από τον τύπο subtotal = quantity x subtotal.

Ως σύνθετο πρωτεύον κλειδί ορίζονται τα πεδία orderIDκαιproductID.

Πίνακας customer

Πεδία: customerID, contactName, shippingAddress, city, state, postalCode, country, phone, email.

πίνακας customer περιέχει πληροφορίες για τα στοιχεία του πελάτη που έχει πραγματοποιήσει μια παραγγελία. Το πεδίο customerID είναι ο κωδικός του πελάτη, το πεδίο contactName είναι το ονοματεπώνυμό του, το πεδίο shippingAddress είναι η διεύθυνση αποστολής, το πεδίο postalCode είναι ο ταχυδρομικός κώδικας, το πεδίο country είναι η χώρα αποστολής, το πεδίο phone είναι το τηλέφωνο του πελάτη και το πεδίο email είναι το e-mail του. Ως πρωτεύον κλειδί ορίζεται το πεδίο customerID.

Πίνακας comments

Πεδία: commentID, productID, addedBy, addedByDate, addedByIP, addedByEmail, body.

Ο πίνακας comments περιέχει πληροφορίες για τα σχόλια που έχουν υποβάλλει οι χρήστες για τα προϊόντα του ηλεκτρονικού καταστήματος. Συγκεκριμένα, αποθηκεύεται ο κωδικός του σχολίου (commentID), ο κωδικός του προϊόντος (productID) για τα οποίο έχει γίνει το σχόλιο, το ψευδώνυμο του χρήστη(addedBy), η ημερομηνία υποβολής (addedByDate), η IPτου χρήστη (addedByIP), το e-mail του (addedByEmail) και το περιεχόμενο του μηνύματος (body). Το πεδίο productID είναι εξωτερικό κλειδί που συνδέει τον συγκεκριμένο πίνακα με τον πίνακα product. Ως πρωτεύον κλειδί ορίζεται το πεδίο commentID.

Πίνακας users

Πεδία:userID, userName, lowered UserName, mobileAlias, is Anonymous, last ActivityDate.

Ο πίνακας users περιλαμβάνει όλους τους εγγεγραμμένους χρήστες της εφαρμογής, οι οποίοι, στην παρούσα έκδοση της εφαρμογής, είναι οι administrators. Ως πρωτεύον κλειδί ορίζεται τοπεδίο userID.

Πίνακας membership

Πεδία: userID, password, password Format, password Salt, mobile Pin, Email, lowered Email, password Question, password Answer, is Approved, is locked Out, create Date, last Login Date, last Password Changed Date, last Lockout Date, failed Password Attemp Count, failed Password Answer Attemp Count, comment.

Ο πίνακας membership περιέχει πληροφορίες για κάθε εγγεγραμμένο χρήστη της εφαρμογής. Το πεδίο userID είναι εξωτερικό κλειδί που συνδέει τον συγκεκριμένο πίνακα με τον πίνακα users. Ως πρωτεύον κλειδί ορίζεται το πεδίο userID.

Πίνακας roles

Πεδία: roleID, roleName, lowered RoleName, description.

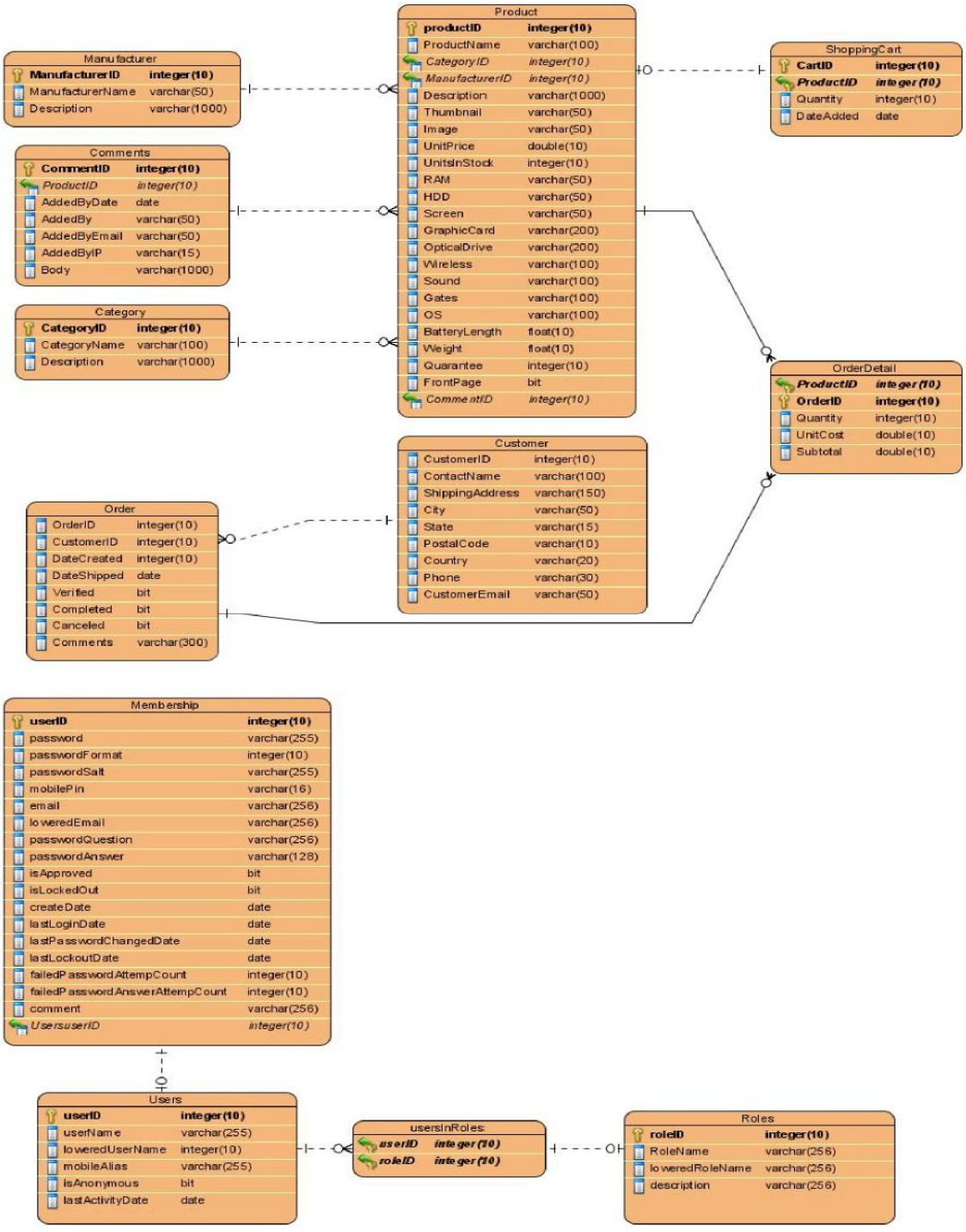
Ο πίνακας roles περιέχει πληροφορίες για όλους τους ρόλους που περιλαμβάνει η εφαρμογή. Να σημειωθεί ότι στην παρούσα έκδοση υποστηρίζεται μόνο ο ρόλος του administrator. Για κάθε ρόλο καταγράφεται ο κωδικός του (roleID), το όνομα του (roleName) και μια σύντομη περιγραφή (description).

Πίνακας users InRoles

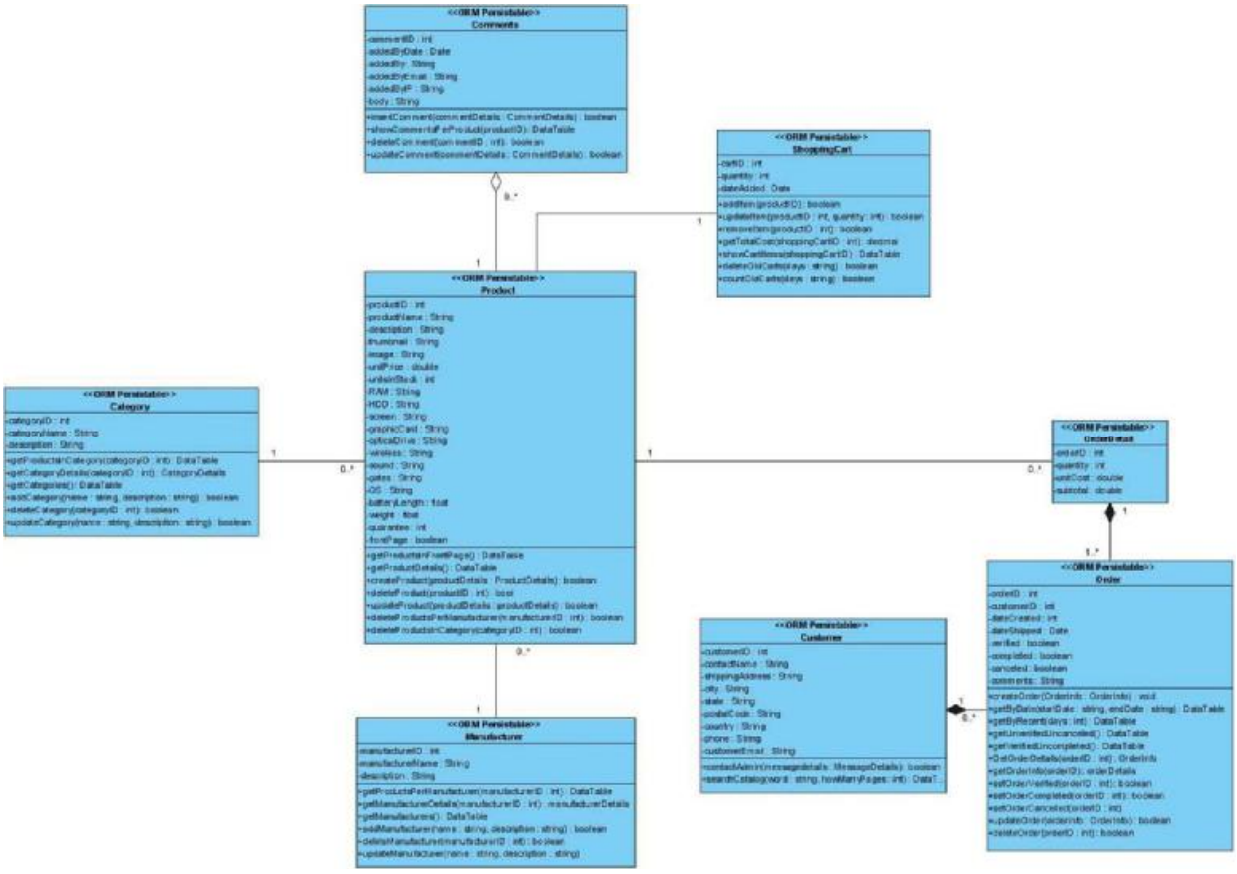
Πεδία: userID, roleID.

Ο πίνακας Users InRoles συνδέει κάθε εγγεγραμμένο χρήστη με κάποιο ρόλο. Ως σύνθετο πρωτεύον κλειδί ορίζονται τα πεδία userID και roleID.

Στην επόμενη σελίδα ακολουθεί το σχεσιακό μοντέλο που περιγράφει πλήρως την βάση δεδομένων του συστήματος.



Εικόνα 76. Βάση συστήματος



Εικόνα 77. Οι κύριες κλάσεις του συστήματος

Επίλογος

Με την παρούσα πτυχιακή πετύχαμε τον αρχικό μας στόχο, ασχοληθήκαμε με την καταγραφή και την παρουσίαση μεθοδολογιών ανάπτυξης έργων λογισμικού και μοντέλων ανάλυσης απαιτήσεων, επικεντρωθήκαμε στην μοντελοποίηση και την μεθοδολογική ανάπτυξη έργων λογισμικού στο πλαίσιο της επιχείρησης, ενώ αναπτύξαμε μια περίπτωση μελέτης ενός Ηλεκτρονικού Καταστήματος, ώστε να καλύψουμε στην πράξη το θεωρητικό μας μέρος.

Στο πρώτο κεφάλαιο, κάναμε μια εισαγωγή στο θέμα, παρουσιάζοντας τις βασικές αρχές της Τεχνολογίας Λογισμικού, την Ιστορία και την εξέλιξή της και τη σχέση της με το επιχειρηματικό περιβάλλον. Στο δεύτερο κεφάλαιο, είδαμε με περισσότερη λεπτομέρεια την Ανάλυση Απαιτήσεων, ένα πολύ νευραλγικό κομμάτι του κύκλου ζωής ανάπτυξης λογισμικού που μπορεί να επηρεάσει θετικά ή αρνητικά, ανάλογα με πόσο προσεκτική δουλειά έχει γίνει, όλη την αλυσίδα της επιλογής μοντέλου, της επιλογής μεθοδολογίας και εν τέλει όλη την ανάπτυξη και το τελικό αποτέλεσμα. Στο τρίτο κεφάλαιο, μιλήσαμε αναλυτικά για τα μοντέλα ανάπτυξης Πληροφοριακών Συστημάτων. Στο τέταρτο κεφάλαιο, αναλύσαμε τις πιο διαδεδομένες μεθοδολογίες ανάπτυξης λογισμικού δίνοντας έμφαση στα δυνατά και αδύναμα σημεία της κάθε μεθοδολογίας όπως επίσης και στην υπόδειξη κάποιων κριτηρίων, τα οποία μπορούν να βοηθήσουν στο να εντοπιστεί η καταλληλότερη μεθοδολογία για συγκεκριμένο project που αναλαμβάνεται προς ανάπτυξη. Στο πέμπτο κεφάλαιο, πραγματοποιήσαμε μια εισαγωγή των εννοιών Μοντελοποίηση και Γλώσσας Μοντελοποίησης, και αναλύσαμε την πιο διαδεδομένη από αυτές(Unified Modelling Language - UML). Τέλος, αφού αναλύσαμε όλο το θεωρητικό πλαίσιο, προχωρήσαμε σε μια περίπτωση μελέτης ενός Ηλεκτρονικού Καταστήματος, για την υλοποίηση του οποίου, εφαρμόσαμε την μεθοδολογία ICONIX με τη χρήση των διαγραμμάτων της Ενοποιημένης Γλώσσας Μοντελοποίησης.

Συμπεράσματα

Τα πλεονεκτήματα για τις εταιρίες που αποφασίζουν την ανάθεση του σχεδιασμού, της υλοποίησης και της συντήρησης των εφαρμογών ηλεκτρονικών προμηθειών τους είναι πολλά και σημαντικά, και αρκεί η ανάλυση των πλεονεκτημάτων για να στραφεί το ενδιαφέρον των επιχειρήσεων στην εξέλιξη αυτή.

Η μείωση του κόστους απόκτησης και εγκατάστασης των εφαρμογών, το κόστος της φιλοξενίας, συντήρησης και αναβάθμισης του εξοπλισμού είναι μηδαμινό εάν αναλυθεί νωρίτερα, ο λόγος που η εταιρεία προχωράει στην ανάθεση ενός έργου για την υλοποίηση ενός έργου που μπορεί να επιφέρει κέρδη είτε, μειώνοντας τον χρόνο εργασίας και διεκπεραίωσης εργασιών που με τις παλαιότερες μεθόδους εργασίας θα ήταν μια πολυδάπανη σε εργατοώρες διαδικασία, είτε ανοίγοντας ένα νέο τρόπο πώλησης στους τελικούς καταναλωτές.

Αναλυτικότερα, η επιχείρηση επικεντρώνεται στις κύριες δραστηριότητές της. Οι δαπάνες του λογισμικού προβλέπονται περισσότερο, ενώ είναι εφικτή η καλύτερη διαχείριση του λογισμικού, διατίθενται περισσότερα κεφάλαια για παραγωγικές επενδύσεις, η αποδοτικότητα είναι μεγαλύτερη και όσον αφορά τις επενδύσεις του λογισμικού, είναι εύκολη η κλιμάκωση σύμφωνα με τις ανάγκες τις εταιρίας, είναι δυνατή η διαρκής πρόσβαση σε τεχνολογίες αιχμής, που είναι απαραίτητες για την ομαλή λειτουργία της επιχείρησης χωρίς ρίσκο εφαρμογής και κόστος εγκατάστασης και διαχείρισης. Παρέχεται η δυνατότητα χρήσης του υπάρχοντος υπολογιστικού εξοπλισμού της εταιρίας, καθώς όλες οι εφαρμογές υποστηρίζονται από το υπολογιστικό σύστημα των εταιριών παροχής λύσεων λογισμικού και υποδομής.

Τέλος, είναι δυνατή η χρήση υπολογιστικής υποδομής, η οποία διαθέτει ασφάλεια, αυτοματοποιημένη δημιουργία αντιγράφων ασφαλείας (actor), αποκατάσταση καταστροφών και υπηρεσίες υποστήριξης, τα οποία θα κόστιζαν αρκετά σε χρήμα και ανθρώπινο δυναμικό εάν τα υλοποιούσε η ίδια η επιχείρηση.

Βέβαια, η υλοποίηση απαιτεί ένα χρηματικό ποσό το οποίο, οι σημερινές επιχειρήσεις, εν μέσω οικονομικής κρίσης, είναι δύσκολο να διαθέσουν, ενώ το μέσο διάστημα μέχρι την υλοποίηση του λογισμικού απαιτείται χρόνος από τους υπαλλήλους για να αναλύσουν όλες τις ανάγκες και τα προβλήματα που έχουν με τον ήδη υπάρχοντα τρόπο εργασίας τους, γεγονός που επιφέρει στον εργοδότη έμμεσο κόστος.

Τέλος, η λανθασμένη επικοινωνία με τους υπαλλήλους, ή η ελλείπεις ανάλυση του έργου μπορεί να επιφέρει τραγική εξέλιξη στην επιχείρηση, καθώς θα έχουν δαπανηθεί χρήματα, τα οποία δεν θα λειτουργήσουν έτσι ώστε να απλοποιήσουν και να τροποποιήσουν τον τρόπο της εσωτερικής συνεργασίας των υπαλλήλων, αλλά είναι δυνατόν να δημιουργηθούν πολλά προβλήματα.

Βιβλιογραφία – Πηγές

1. Booch G., Rumbaugh J. and Jacobson I., the Unified Modelling Language User Guide, Addison Wesley, Boston, MA, 1999.

2. Eriksson E., PenkerM. UML Toolkit, John Wiley.

3. Χατζηγεωργίου Α., Αντικειμενοστρεφής Σχεδίαση, Εκδόσεις Κλειδάριθμος 2005.

4. Wuyts R, UML: History and Overview

   http://www.ulb.ac.be/di/rwuhttp://www.ulb.ac.be/di/rwuyts/INFO025_2004/04-UML-Overview.pdfyts/INFO025_2004/

5. Γερογιάννης Β., Καραρόντζας Γ., Καμέας Α., Σταμέλος Γ., Φιτσιλής Π., Αντικειμενοστρεφής Ανάπτυξη Λογισμικού με τη UML, Κλειδάριθμος 2006.

6. J. Warmer, A. Kleppe, The Object Constraint Language: Precise Modeling with UML, Object Technology Series, Addison-Wesley, 1999.

7. Chen R., Sgro1 M., Lavagno L., Martin G., Sangiovanni- Vincentelli A., Rabaey J., UML and Platform-based Design, IEEE Computer Society 2002

8. B. Selic, "A Generic Framework forModelling Resources with UML", *IEEE Computer*, June 2000.

9. Brenda M. Michelson, Business Process Execution Language (BPEL) Primer, Understanding an Important Component of SOA and Integration Strategies. PDF, Patricia Seybold Group, www.psgroup.com

10. Kenneth C. Laudon, Jane P. Laudon, Management Information Systems, Seventh Edition, Prentice Hall,p.50-55

11. Matthias Kloppmann, Dieter Koenig, Frank Leymann, Gerhard Pfau, Alan Rickayzen, Claus von Riegen, Patrick Schmidt, Ivana Trickovic, WS-BPEL Extension for Sub-processes – BPEL-SPE, White Paper by IBM and SAP, September 2005

12. Practical UML: A Hands-On Introduction for Developers, Randy

Miller,http://bdn.borland.com/article/

13. Santosh S. Venkatraman, Web-Services – The Next Evolutionary Stage of E-Business. PDF, Tennessee State University, Journal of International Technology and Information Management, Volume 13, 2004

14. Soon Huat Lim, Neal Juster, Alan de Pennington, The Seven Major Aspects of Enterprise Modelling and Integration: A Position Paper.pdf, SIGGROUP Bulletin, Vol. 18, No. 1 (April 1997).

15. Wikipedia, http://en.wikipedia.org/wiki/Unified_Modeling_Language.

16. Royce, Winston, "Managing the Development of Large Software Systems», Proceedings of IEEE WESCON 26, 1970.

17. Εμμανουήλ Σκορδαλάκης. *Εισαγωγή στην Τεχνολογία Λογισμικού*. Συμμετρία, 1991.

18. Ian Sommerville. Software Engineering. Addison-Wesley, sixth edition, 2001.

19. Βεσκούκης, Β. Αρχές Τεχνολογίας Λογισμικού- Τεχνολογία Λογισμικού Ι, Πάτρα, Εκδόσεις ΕΑΠ. 2000.

20. http://www.dmst.aueb.gr/louridas/lectures/dais/uml/ar01s10.html

21. http://dtps.unipi.gr/files/notes/2009-2010/eksamino_3/texnologia_logismikoy/02_-_montela_kykloy_zwhs_logismikoy.pdf

22. http://dtps.unipi.gr/files/notes/2008-2009/eksamino_3/texnologia_logismikoy/eisagwgh_sthn_texnologia_logismikoy_-_montela_kykloy_zwhs_logismikoy.pdf

23. http://dtps.unipi.gr/files/notes/2007-2008/eksamino_3/texnologia_logismikoy/22_1192185501.pdf

24. http://www.syros.aegean.gr/users/gaviotis/is05/SAD.pdf