**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ**
**ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**
**ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ:**
**¨ΤΕΧΝΟΟΙΚΟΝΟΜΙΚΗ ΔΙΟΙΚΗΣΗ ΚΑΙ ΑΣΦΑΛΕΙΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ¨**
**ΚΑΤΕΥΘΥΝΣΗ:**
**¨ΤΕΧΝΟΟΙΚΟΝΟΜΙΚΗ ΔΙΟΙΚΗΣΗ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ¨**

# ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

## Ανάλυση απαιτήσεων για την ανάπτυξη πληροφοριακών συστημάτων. Μεθοδολογίες ανάλυσης απαιτήσεων στο πλαίσιο εναλλακτικών κύκλων ζωής έργων πληροφοριακών συστημάτων. Διενέργεια σχετικής μελέτης περίπτωσης.

**ΤΡΙΑΝΤΗΣ Α. ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΤΕ 1034**
**ΕΠΙΒΛΕΠΟΥΣΑ: ΜΑΛΑΜΑΤΕΝΙΟΥ ΦΛΩΡΑ**

**ΠΕΙΡΑΙΑΣ, 2013**

*Στους γονείς μου*

## Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάλυση των μεθοδολογιών της Μηχανικής των Απαιτήσεων σε έργα ανάπτυξης πληροφοριακών συστημάτων. Σκοπός της εργασίας είναι τόσο η παρουσίαση των διαδικασιών της Μηχανικής των Απαιτήσεων, όσο και η ανάδειξη της αναγκαιότητάς της για την ορθή δημιουργία ενός ολοκληρωμένου πληροφοριακού συστήματος. Αρχικά αναλύονται τα μοντέλα των κύκλων ζωής των πληροφοριακών συστημάτων. Στη συνέχεια δίνονται οι ορισμοί που περιγράφουν την ορολογία της Μηχανικής των Απαιτήσεων. Κατόπιν, παρουσιάζονται τα στάδια της Ανάπτυξης των Απαιτήσεων που αποτελούνται από την εκμαίευση, την ανάλυση, την προδιαγραφή και την επικύρωση των απαιτήσεων, ενώ, παράλληλα, εξηγούνται οι διαδικασίες που γίνονται σε κάθε ένα από τα στάδια αυτά. Ακολουθεί η επεξήγηση του σταδίου της διαχείρισης των αλλαγών των απαιτήσεων και παρουσιάζονται οι διαδικασίες και τα εργαλεία που χρησιμοποιούνται. Τέλος, παρουσιάζεται μία μελέτη περίπτωσης που αφορά στη διαδικασία της Μηχανικής των Απαιτήσεων στο χώρο της υγείας και, συγκεκριμένα, κατά τη δημιουργία ενός συστήματος κατ' οίκον νοσηλείας.




**Λέξεις κλειδιά**: Απαιτήσεις, Ανάπτυξη Απαιτήσεων, Διαχείριση Απαιτήσεων, Ανάλυση Απαιτήσεων, Εκμαίευση, Τεκμηρίωση, Επικύρωση, Κύκλος Ζωής , Μοντελοποίηση.

## Abstract

This thesis focuses on the analysis of the methodologies for Engineering Requirements in iformation systems' development projects. The purpose of this work is the presentation of the processes for defining Engineering Requirements, and emergence of the necessity of Engineering Requirements for the proper establishment of an integrated information system. Initially, the information system's life and development cycle models are analyzed. Afterwards, definitions and explanations for better understanding of the Requirements Engineering process are given. Further, the Requirements Engineering stages are described, such as requirements' elicitation, analysis, specification and validation. Then, the process of requirements change management is explained and the procedures and the tools that are used for this are presented. Finally, a case study which analyzes the requirements of home health care system, and utilizes the Requirements Engineering process is presented.



**Keywords**: Requirements, Requirements Development, Requirements Management, Requirements Analysis, Elicitation, Specification, Validation, Life Cycle Modeling.

## Πίνακας Περιεχομένων

# Περιεχόμενα Σχημάτων

# Περιεχόμενα Πινάκων

# Εισαγωγή

Οι απαιτήσεις των έργων ανάπτυξης πληροφοριακών συστημάτων διακρίνονται σε τρία πίπεδα: στις επιχειρηματικές, στις απαιτήσεις του χρήστη και στις λειτουργικές. Προέρχονται από διάφορες πηγές και χρονικές στιγμές κατά τη διάρκεια του έργου. Ενδιαφέρουν διαφορετικά άτομα, έχουν άλλους σκοπούς και πρέπει να τεκμηριωθούν με διαφορετικούς τρόπους. Οι επιχειρηματικές απαιτήσεις, που αναφέρονται στο πεδίο ορισμού του έργου, δεν θα πρέπει να έρχονται σε σύγκρουση και να αποκλείουν τις απαιτήσεις του χρήστη. Οι απαιτήσεις του χρήστη με την σειρά τους θα πρέπει θα πρέπει να ανιχνεύονται σε κάθε λειτουργική απαίτηση. Είναι επίσης αναγκαίο να εκμαιεύσουμε και τις μη λειτουργικές απαιτήσεις, αλλά κάτι τέτοιο θα πρέπει να γίνει από τις κατάλληλες πηγές.

Η Μηχανική των Απαιτήσεων είναι μία διαδικασία που ασχολείται με όλες τις φάσεις του συνόλου των απαιτήσεων για τη δημιουργία ενός Πληροφοριακού Συστήματος ή μίας εφαρμογής λογισμικού. Περιλαμβάνει τις διαδικασίες της εκμαίευσης, της ανάλυσης, της προδιαγραφής, της επικύρωσης και της διαχείρισης των απαιτήσεων. Είναι μία αρκετά χρονοβόρα διαδικασία αλλά αποτελεί επένδυση για την δημιουργία ενός σωστού συστήματος. Όσος περισσότερος χρόνος και πόροι δαπανώνται κατά την Μηχανική των Απαιτήσεων, τόσο πιο εύκολα, ορθά και γρήγορα θα προχωρήσει η σχεδίαση και η υλοποίηση του έργου. Στην συνέχεια παρατίθενται επιγραμματικά τα βήματα της Μηχανικής των Απαιτήσεων που πρέπει να ακολουθηθούν.

Πρώτα θα πρέπει να δημιουργηθεί ένας οδηγός για το πώς θα εκτελεστούν οι διαδικασίες της εκμαίευσης, της ανάλυσης, της τεκμηρίωσης και της επικύρωσης των απαιτήσεων , που θα βοηθήσει την δουλειά των αναλυτών, τον έλεγχο της όλης διαδικασίας και τον χρονοπρογραμματισμό του έργου και να δημιουργηθεί ένα έγγραφο οράματος και πεδίου ορισμού του έργου. Το έγγραφο αυτό είναι ένα εργαλείο αξιολόγησης στο οποίο θα αναφέρεται τι πρέπει και τι δεν πρέπει να παραδοθεί. Αφού πραγματοποιηθούν τα παραπάνω, θα πρέπει να οριστούν οι κλάσεις των χρηστών, όπως επίσης και τα χαρακτηριστικά τους, να περιγραφθούν οι δραστηριότητες τους, οι τοποθεσίες τους, η συχνότητα χρήσης του συστήματος από την κάθε ομάδα και τα επίπεδα πρόσβασης στα οποία χωρίζονται. Στην πορεία θα οριστεί ένας Project Champion, ένας εκπρόσωπος στην ουσία, για την κάθε ομάδα χρηστών ο οποίος θα είναι και ο υπεύθυνος να παίρνει μέρος στις αποφάσεις και να εξηγεί στην ομάδα του τι ακριβώς συμβαίνει ανά πάσα στιγμή. Ο ίδιος ή μία ομάδα πελατών θα πρέπει να ορίσουν τα ποιοτικά χαρακτηριστικά και την λειτουργικότητα που επιθυμούν να έχει το σύστημα.

Εφόσον περιγραφούν οι λειτουργίες και τα χαρακτηριστικά του έργου, προχωράμε στην διαδικασία της ανάλυσης των απαιτήσεων, η οποία περιλαμβάνει τον ορισμό τους έτσι ώστε να διασφαλιστεί ότι όλοι οι ενδιαφερόμενοι είναι σε θέση να τις κατανοήσουν. Στο στάδιο αυτό

γίνεται ανάλυση του υψηλού επιπέδου των απαιτήσεων σε λεπτομέρειες, δημιουργούνται πρωτότυπα, αξιολογείται η εφικτότητα και ορίζονται οι προτεραιότητες. Σκοπός είναι να δημιουργηθούν ορθές απαιτήσεις που θα βοηθήσουν τους εμπλεκόμενους να κάνουν σωστές εκτιμήσεις για τον σχεδιασμό του έργου. Η αναπαράσταση των απαιτήσεων είναι καλό να γίνεται τόσο σε έγγραφη μορφή, όσο και σε διαγραμματικές αναπαραστάσεις για να γίνονται εύκολα κατανοητές από όλους και να εντοπίζονται γρηγορότερα τα όποια σφάλματα. Όταν υπάρχει αβεβαιότητα σχετικά με τις απαιτήσεις θα πρέπει να δημιουργούνται πρωτότυπα, που έχουν σαν απώτερο σκοπό να κατανοηθεί και να λυθεί το οποιοδήποτε πρόβλημα. Ένα σημαντικό κομμάτι της ανάλυσης βασίζεται στον ορισμό προτεραιοτήτων στις απαιτήσεις, έτσι ώστε να γίνει πιο ομαλά και πιο γρήγορα η ανάπτυξή τους. Για να μπορέσουν να συνεννοηθούν σωστά όλοι οι συμμετέχοντες θα πρέπει να μιλούν με τους ίδιους όρους. Σε αυτό θα βοηθήσει το λεξικό που θα δημιουργηθεί για να διασφαλίσει την ακριβή και σωστή επικοινωνία μεταξύ των εμπλεκομένων μερών. Τέλος μία χρήσιμη τεχνική στην διαδικασία της ανάλυσης είναι η Quality Function Deployment. Αυτό που κάνει είναι να συσχετίσει τα στοιχεία και τα χαρακτηριστικά του προϊόντος με την αξία που θα πάρει ο χρήστης από αυτά. Με την τεχνική αυτή μεγιστοποιείται η ικανοποίηση του πελάτη από το παραγόμενο σύστημα.

Η ανάλυση δίνει την θέση της στο επόμενο στάδιο της Μηχανικής των Απαιτήσεων, αυτό της τεκμηρίωσης. Οι απαιτήσεις θα πρέπει να τεκμηριώνονται με τρόπο συνεπή, προσβάσιμο και αναγνώσιμο. Οι επιχειρηματικές απαιτήσεις συνήθως τεκμηριώνονται στο έγγραφο οράματος και πεδίου ορισμού του έργου, ενώ οι απαιτήσεις των χρηστών απεικονίζονται με την μορφή σεναρίων χρήσης και σε Πίνακες Δράσης – Απόκρισης (Event – Response Tables) . Οι λειτουργικές και μη λειτουργικές απαιτήσεις περιέχονται στο Έγγραφο Προδιαγραφής Απαιτήσεων (Software Requirements Specification ή SRS). Το Έγγραφο Τεκμηρίωσης Απαιτήσεων θα πρέπει να είναι προσαρμοσμένο σε ένα κυρίαρχο πρότυπο που θα προσδίδει μία συνεπή δομή για την καταγραφή τις λειτουργικότητας και των πληροφοριών που σχετίζονται με τις απαιτήσεις. Υπάρχουν διάφορα πρότυπα, άλλα πολλοί οργανισμοί χρησιμοποιούν το IEEE 830-1998. Η κάθε απαίτηση θα πρέπει να συνδέεται με την πηγή της, έτσι ώστε να υπάρχει μεγαλύτερη διασάφηση. Θα πρέπει να καταγράφονται οι ενδιαφερόμενοι, αλλά και οι επιχειρηματικοί κανόνες που σχετίζονται με την κάθε απαίτηση, για να γίνεται ευκολότερα η ενημέρωσή τους σε περίπτωση αλλαγής. Αυτές οι συσχετίσεις μπορούν να γίνουν με διασυνδέσεις ιχνηλασιμότητας. Για τον λόγο αυτό θα πρέπει να δίνεται μία ξεχωριστή ετικέτα ταυτοποίησης σε κάθε απαίτηση (Labeling). Πέρα όμως από την λειτουργικότητα θα πρέπει να οριστούν και τα ποιοτικά χαρακτηριστικά που θα οδηγήσουν το προϊόν να ικανοποιήσει τις προσδοκίες του πελάτη.

Αφού οι απαιτήσεις τεκμηριωθούν, θα πρέπει να επικυρωθούν για να διασφαλιστεί πως είναι ορθές και ικανοποιούν τους χρήστες. Πριν ξεκινήσει η διαδικασία της σχεδίασης και της υλοποίησης, θα πρέπει να σιγουρευτούμε πως οι απαιτήσεις του συστήματος είναι αξιόπιστες, υλοποιήσιμες και ελέγξιμες. Η επιθεώρηση είναι μία από τις σημαντικότερες τεχνικές εξακρίβωσης της ποιότητας του λογισμικού. Θα πρέπει να δημιουργηθεί μία μικρή ομάδα

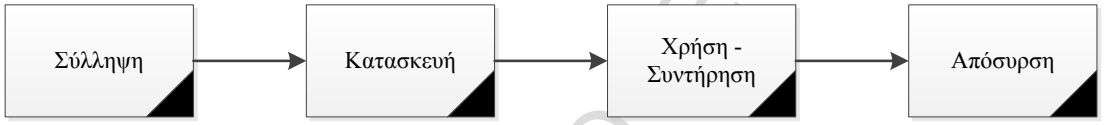
επιθεωρητών, που θα αντιπροσωπεύει τους διάφορους εμπλεκόμενους και να εξετάσει προσεκτικά το Έγγραφο Τεκμηρίωσης Απαιτήσεων, τα μοντέλα ανάλυσης και τις όποιες συμπληρωματικές πληροφορίες. Τέλος, θα πρέπει να διενεργηθούν δοκιμές ελέγχου (TestCases), που θα πιστοποιήσουν την αναμενόμενη συμπεριφορά του προϊόντος κάτω από ορισμένες συνθήκες. Θα πρέπει να γίνουν δοκιμές για όλες τις λειτουργικές απαιτήσεις και να διαπιστωθεί η ορθότητα των μοντέλων ανάλυσης και των πρωτοτύπων. Για να γίνει αυτό θα πρέπει οι χρήστες να περιγράψουν κατά πόσο το σύστημα ικανοποιεί τις ανάγκες τους και να δημιουργηθούν κριτήρια αποδοχής των σεναρίων χρήσης.

Το τελικό στάδιο είναι η διαχείριση των απαιτήσεων (Requirements Management ή RM). Είναι η διαδικασία που περιλαμβάνει τον έλεγχο των αλλαγών και τη διαπραγμάτευση των απαιτήσεων με τους εμπλεκόμενους φορείς (Stakeholders). Με τη διαδικασία αυτή αυξάνουμε την αξία των απαιτήσεων, αφού βεβαίως πρώτα έχει γίνει η εκμαίευσή τους. Για να διατηρηθεί ένα υψηλό επίπεδο όσον αφορά την ποιότητα των απαιτήσεων, οι αλλαγές τους θα πρέπει να αναλύονται σχολαστικά έτσι ώστε να καταλαβαίνουμε πόση προσπάθεια και πόσοι πόροι καταναλώθηκαν για να γίνουν. Είναι μία καλή μεθοδολογία να αντιμετωπιστούν οι αλλαγές στις απαιτήσεις αφού αποτρέπει την φθορά τους και προσπαθεί να τις κρατάει όσο το δυνατόν πιο ανέπαφες. Για να επιτύχουμε σωστή διαχείριση απαιτήσεων θα πρέπει να εφαρμόσουμε και κάποιες διαδικασίες μέτρησης. Ο αριθμός των απαιτήσεων, η συχνότητα των αλλαγών και τα ποσοστά ολοκλήρωσης θα μπορούσαν ήταν να είναι κάποιοι τέτοιοι παράγοντες. Έτσι οι Διαχειριστές του Έργου (Project Managers) θα μπορούν να έχουν μία πιο ολοκληρωμένη εικόνα του έργου, να ευθυγραμμίζονται καλύτερα με τις αναθεωρήσεις και τις αλλαγές που γίνονται και να συντονίζουν ευκολότερα την διαδικασία της ανάπτυξης. Καταλαβαίνουμε λοιπόν πόσο σημαντική είναι η διαχείριση, τόσο για την διευκόλυνση της δουλειάς των Διαχειριστών, όσο και για την γρηγορότερη ολοκλήρωση του έργου.

# 1 Πληροφοριακά Συστήματα Και Κύκλοι Ζωής

## 1.1 Μοντέλα Κύκλου Ζωής

Κάθε εφαρμογή Λογισμικού, από τη σύλληψη μέχρι την απόσυρσή της, διέρχεται από διάφορες φάσεις σε κάθε μία εκ των οποίων πρέπει να γίνονται ορισμένες εργασίες ώστε να επιτυγχάνεται το επιθυμητό αποτέλεσμα. Σε μακροσκοπικό επίπεδο οι πολύ γενικές φάσεις είναι η σύλληψη, η κατασκευή, η χρήση, η συντήρηση και απόσυρση. Το Σχήμα 1 παρουσιάζει μια εικόνα των γενικών αυτών φάσεων [1].




**Σχήμα 1.** Γενικές φάσεις του κύκλου ζωής του Λογισμικού

Πριν αναφερθούμε στον ορισμό του μοντέλου κύκλου ζωής και στα σημαντικότερα τέτοια μοντέλα που χρησιμοποιούνται σήμερα, είναι σκόπιμο να δοθούν ορισμένοι χρήσιμοι ορισμοί οι οποίοι θα χρησιμοποιηθούν εκτεταμένα στη συνέχεια.

**Δραστηριότητα ανάπτυξης Λογισμικού**. Μια δραστηριότητα ή διαδικασία ανάπτυξης λογισμικού καθορίζει ποιες ενέργειες πρέπει να γίνουν για να επιτευχθεί ένα επιθυμητό αποτέλεσμα σε κάποια από τις φάσεις του κύκλου ζωής. Μια δραστηριότητα μπορεί να αναλύεται σε περισσότερες από μία επιμέρους φάσεις.

**Μεθοδολογία ανάπτυξης**. Μια μεθοδολογία (software development methodology), καθορίζει το πώς θα πρέπει να εκτελούνται οι δραστηριότητες ανάπτυξης, δηλαδή ποιες επιμέρους ενέργειες περιλαμβάνουν, ποια βήματα γίνονται σε κάθε μία, ποια προϊόντα παράγονται, καθώς και πότε αυτές θεωρούνται περατωθείσες.

**Εργαλείο λογισμικού (CASE: Computer-Aided Software Engineering).** Είναι ένα σύστημα (συνήθως είναι και το ίδιο εφαρμογή Λογισμικού) το οποίο υποστηρίζει τη μερική ή ολική αυτοματοποίηση των εργασιών που λαμβάνουν χώρα κατά την εφαρμογή των μεθοδολογιών ανάπτυξης λογισμικού.

Με βάση τα προηγούμενα, μπορούμε να δώσουμε τον ορισμό του Μοντέλου Κύκλου Ζωής Λογισμικού:

**Μοντέλο Κύκλου Ζωής Λογισμικού**. Ένα μοντέλο κύκλου ζωής Λογισμικού είναι μια περιγραφή των δραστηριοτήτων και των επιμέρους φάσεων από τις οποίες διέρχεται μια εφαρμογή Λογισμικού από τη σύλληψη μέχρι την απόσυρσή της, καθώς και των εργασιών που λαμβάνουν χώρα σε κάθε μία από τις φάσεις αυτές.

Στο Σχήμα 2 φαίνεται η σχέση μεταξύ των εννοιών "μοντέλο κύκλου ζωής", "διαδικασία ανάπτυξης", "μεθοδολογία", καθώς και "εργαλείο" οι οποίες ορίστηκαν προηγουμένως. Η έννοιες που βρίσκονται στα χαμηλότερα επίπεδα της πυραμίδας , αποτελούν το υπόβαθρο πάνω στο οποίο βασίζονται οι έννοιες που βρίσκονται στα αμέσως υψηλοτέρα σημεία [1].


**Σχήμα 2.** Σχέσεις εννοιών στην ανάπτυξη του Λογισμικού

Τα μοντέλα κύκλου ζωής λογισμικού προσδιορίζουν τις διαδικασίες ανάπτυξης οι οποίες λαμβάνουν χώρα κατά τις γενικές φάσεις "κατασκευή" και "χρήση - συντήρηση", προσδιορίζοντας τις επιμέρους φάσεις στις οποίες αυτές αναλύονται, τα προϊόντα που παράγονται σε καθεμία από αυτές, καθώς και την σειρά εκτέλεσής τους. Σε κάθε διαδικασία ανάπτυξης μπορούμε να διακρίνουμε περισσότερες από μία επιμέρους φάσεις, ενώ σε κάθε επιμέρους φάση μπορούμε να διακρίνουμε περισσότερες από μία εργασίες. Οι διαδικασίες ανάπτυξης Λογισμικού μπορούν να ταξινομηθούν ως ακολούθως [1]:

- Προδιαγραφή, δηλαδή καθορισμός των εργασιών που θα επιτελεί το Λογισμικό, καθώς και των περιορισμών και των παραδοχών που ισχύουν.
- Ανάπτυξη, δηλαδή κατασκευή του Λογισμικού. Εδώ, σε όλα τα μοντέλα κύκλου ζωής, μπορούμε να διακρίνουμε τρεις επιμέρους φάσεις: την ανάλυση, τη σχεδίαση και την συγγραφή του πηγαίου κώδικα (Source Code) την οποία στη συνέχεια θα ονομάζουμε και κωδικοποίηση.
- Επαλήθευση, δηλαδή επιβεβαίωση της ικανοποίησης των προδιαγραφών και της μη ύπαρξης σφαλμάτων.
- Εξέλιξη, δηλαδή επαύξηση των λειτουργικών χαρακτηριστικών του Λογισμικού ή τροποποίηση υπαρχουσών, προκειμένου να ικανοποιούνται οι μεταβαλλόμενες ανάγκες.

Ένα μοντέλο κύκλου ζωής Λογισμικού στοχεύει στην καθοδήγηση του κατασκευαστή προκειμένου αυτός να επιτύχει την καλύτερη δυνατή υλοποίηση των διαδικασιών ανάπτυξης Λογισμικού. Λέγοντας "καλύτερη δυνατή" εννοούμε περισσότερο παραγωγική, με τα λιγότερα δυνατά σφάλματα και το μικρότερο δυνατό ρίσκο στις εκάστοτε συνθήκες. Τα παραπάνω μπορούν να διαφοροποιούνται ανάλογα με το μέγεθος και το θεματικό πεδίο κάθε εφαρμογής λογισμικού, με την εμπειρία και τα ιδιαίτερα χαρακτηριστικά του κάθε κατασκευαστή και ασφαλώς με το εκάστοτε περιβάλλον ανάπτυξης.

Μια σημαντική παράμετρος που καταδεικνύει τη σημασία των μοντέλων κύκλου ζωής είναι το κόστος, ιδωμένο με την ευρύτερη σημασία του. Το κόστος αναθεώρησης αποφάσεων και διόρθωσης σφαλμάτων είναι τόσο μεγαλύτερο, όσο μεγαλύτερη είναι και η απαιτούμενη οπισθοδρόμηση της διαδικασίας που αυτή συνεπάγεται. Το κόστος αυτό δεν αφορά μόνο οικονομικούς πόρους που αποδίδονται στο έργο, αλλά και χρόνο καθυστέρησης που δεν είναι πάντα διαθέσιμος σε πραγματικές συνθήκες. Επίσης είναι συχνό φαινόμενο οι παρενέργειες στο υπόλοιπο σύστημα λογισμικού (Sideeffects) οι οποίες μπορούν να μεταβάλλουν προς το χειρότερο τα ποιοτικά του χαρακτηριστικά και δεν είναι εύκολο να εντοπιστούν από την αρχή.

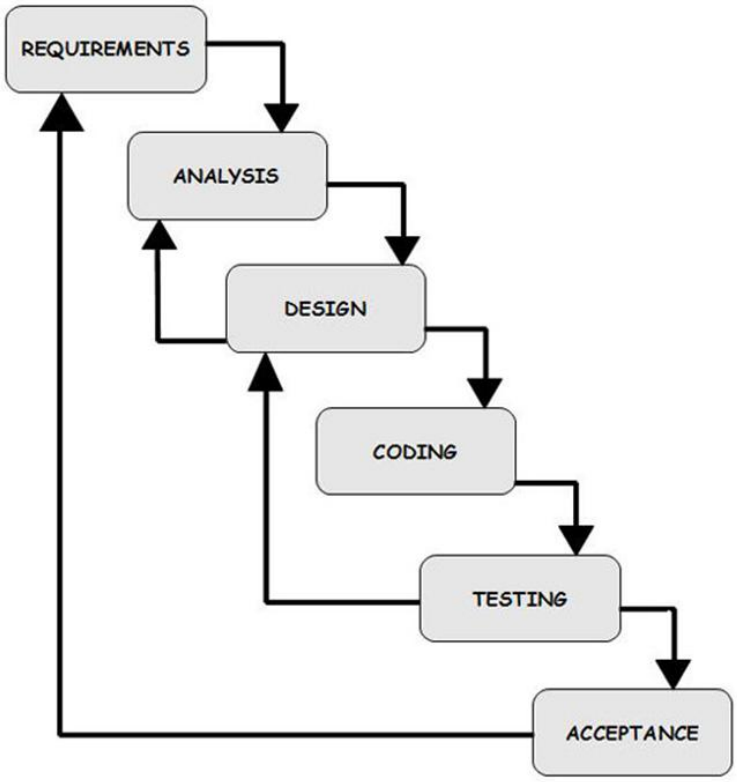
Υπάρχουν αρκετά μοντέλα κύκλου ζωής τα οποία διαφοροποιούνται ως προς τη σύλληψη της ιδέας του τρόπου κατασκευής αλλά και ως προς τις επιμέρους φάσεις που προτείνουν, την επαναληπτικότητα και την εμβέλεια των εργασιών αυτών, τα ενδιάμεσα προϊόντα – συστατικά Λογισμικού και την περιγραφή τους καθώς επίσης και στις οικονομικές και επιχειρηματικές πλευρές της χρήσης τους. Κάθε μία από τις ενέργειες που περιγράφεται σε ένα μοντέλο κύκλου ζωής είναι μια διαδικασία επίλυσης προβλημάτων (Problem Solving Process).

### 1.1.1 Το μοντέλο του καταρράκτη

Το πρώτο και γηραιότερο μοντέλο κύκλου ζωής λογισμικού, αυτό του καταρράκτη, αντιμετωπίζει την ανάπτυξη του λογισμικού σαν την μεταφορά ενός μεγάλου ογκόλιθου από ένα σημείο σε κάποιο άλλο περνώντας από ενδιάμεσες στάσεις, αλλά μεταφέροντας από τη μία στάση στην άλλη ολόκληρο τον ογκόλιθο. Επειδή οι εφαρμογές λογισμικού είναι περισσότερο εύπλαστες και ευμετάβλητες από τους ογκόλιθους, σύντομα παρουσιάστηκαν προβλήματα στην ιδέα, οπότε εμφανίστηκαν και άλλα μοντέλα κύκλου ζωής, τα οποία μεταφέρουν με διαφορετικούς και πιο ευέλικτους τρόπους μικρότερα τμήματα του "ογκόλιθου".

Αυτό, λοιπόν, που διαφοροποιεί τα διάφορα μοντέλα κύκλου ζωής λογισμικού είναι η εμβέλεια, δηλαδή η έκταση του υπό κατασκευή συστήματος Λογισμικού στην οποία αυτές οι διαδικασίες εφαρμόζονται, η επαναληπτικότητα των εργασιών, καθώς και οι ενδιάμεσες αποτιμήσεις από τον πελάτη ή τον κατασκευαστή. Έτσι, σε κάθε μοντέλο κύκλου ζωής, είναι με διαφορετικό τρόπο δυνατός ο εντοπισμός της ανάγκης και η ενσωμάτωση τροποποιήσεων στα χαρακτηριστικά του Λογισμικού προτού να ολοκληρωθεί πλήρως η κατασκευή του, οπότε μπορεί να είναι αργά από πλευράς χρόνου και κόστους [1,2].

Είναι ένα από τα πιο διαδεδομένα μοντέλα κύκλου ζωής. Η κεντρική ιδέα του μοντέλου του καταρράκτη είναι ότι το σύστημα λογισμικού αναπτύσσεται περνώντας ολόκληρο από διαδοχικές επιμέρους φάσεις, καθεμία από τις οποίες θεωρείται περατωμένη με την παραγωγή ορισμένων συστατικών Λογισμικού. Κάθε επιμέρους φάση ολοκληρώνεται με μια εργασία επαλήθευσης και επικύρωσης των προϊόντων της κατά την οποία αποφασίζεται η μετάβαση ή όχι στην επόμενη. Το λογισμικό εμφανίζεται πλήρες, δηλαδή με όλα τα λειτουργικά του χαρακτηριστικά, από την επιμέρους φάση της συνένωσης και μετά. Χαρακτηριστικό του μοντέλου του καταρράκτη είναι ότι για να ξεκινήσει μια φάση, πρέπει να έχει ολοκληρωθεί πλήρως η προηγούμενη. Η ανάπτυξη με τον τρόπο αυτό χαρακτηρίζεται ακολουθιακή διότι οι επιμέρους φάσεις από τις οποίες διέρχεται είναι διακριτές και ακολουθούν η μία την άλλη. Πρέπει να σημειωθεί ότι υπάρχουν τα μοντέλα **V** και **W** που αποτελούν παραλλαγές του μοντέλου του καταρράκτη.




**Σχήμα 3.** Διαγραμματική απεικόνιση του μοντέλου του καταρράκτη

Σύμφωνα με αυτό το μοντέλο αρχικά καθορίζονται οι απαιτήσεις από το σύστημα και το Λογισμικό. Ακολούθως, γίνεται η προκαταρτική και η λεπτομερής σχεδίαση του Λογισμικού, αντίστοιχα. Κατά την προκαταρτική σχεδίαση καθορίζονται οι μονάδες που θα αποτελούν το Λογισμικό, καθώς και οι συσχετίσεις μεταξύ τους. Ο καθορισμός αυτός μπορεί να γίνει σε περισσότερα από ένα επίπεδα λεπτομέρειας ανάλογα με το μέγεθος και την πολυπλοκότητα του

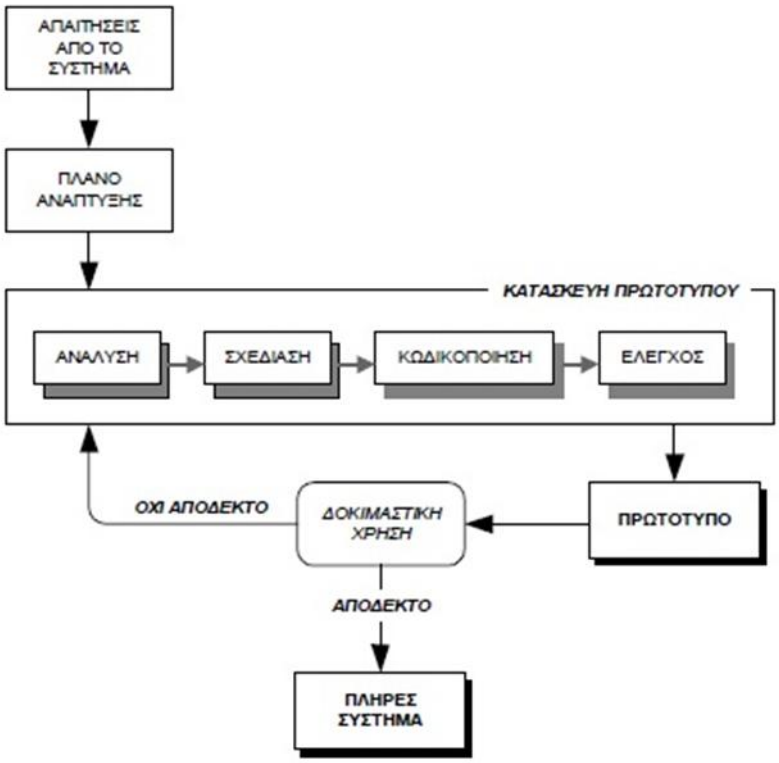
Λογισμικού. Στα πρώτα επίπεδα υπάρχει μικρός βαθμός λεπτομέρειας και όσο ανεβαίνουμε αυξάνεται.

Κατά την λεπτομερή σχεδίαση καθορίζεται η εσωτερική δομή κάθε μονάδας Λογισμικού η οποία αντιστοιχεί πρακτικά σε μονάδες πηγαίου κώδικα προγράμματος. Ο καθορισμός αυτός περιλαμβάνει όλα τα απαραίτητα στοιχεία (αλγόριθμοι, δομές δεδομένων, κλπ), ώστε η συγγραφή του πηγαίου κώδικα, που ακολουθεί να είναι μια διαδικασία διεκπεραίωσης και μόνο. Ακολουθεί η συνένωση των μονάδων σε σύστημα και ο έλεγχος του συστήματος, η ολοκλήρωση του οποίου επιτρέπει την παράδοση ολόκληρου του προϊόντος στον πελάτη και το πέρασμα στη φάση της λειτουργίας και συντήρησης.

Το μοντέλο του καταρράκτη υπήρξε για μεγάλο διάστημα το πιο διαδεδομένο μοντέλο κύκλου ζωής Λογισμικού. Είναι ιδιαίτερα χρήσιμο σε περιπτώσεις όπου οι απαιτήσεις από το λογισμικό είναι από την αρχή γνωστές και δεν μεταβάλλονται κατά την ανάπτυξη του λογισμικού και μπορεί να χρησιμοποιηθεί αποτελεσματικά για τη βιομηχανοποίηση της ανάπτυξης τέτοιων εφαρμογών. Για παράδειγμα τέτοιες είναι οι εφαρμογές που επιλύουν μεγάλα προβλήματα χρησιμοποιώντας μαθηματικούς υπολογισμούς. Σε πολλές, όμως, περιπτώσεις εφαρμογών, οι απαιτήσεις είτε δεν είναι από την αρχή και με σαφήνεια γνωστές, είτε ενδέχεται να μεταβληθούν κατά τη διάρκεια της ανάπτυξης.

### 1.1.2 Το μοντέλο της πρωτοτυποποίησης

Η κεντρική ιδέα του μοντέλου πρωτοτυποποίησης (Prototyping Model) είναι η ανάπτυξη του λογισμικού όχι εξ' ολοκλήρου, αλλά σε τμήματα που ονομάζονται πρωτότυπα. Οι διαδικασίες ανάπτυξης επαναλαμβάνονται για ένα τμήμα του συστήματος κάθε φορά και για το λόγο αυτό το μοντέλο χαρακτηρίζεται ως επαναληπτικό. Κάθε πρωτότυπο περιλαμβάνει τις βασικές από τις λειτουργίες που προορίζεται να εκτελεί το Λογισμικό και τίθεται σε δοκιμασία από τον πελάτη. Από εκεί συλλέγονται παρατηρήσεις και η διαδικασία κατασκευής νέου πρωτοτύπου επαναλαμβάνεται μέχρις ότου ένα πρωτότυπο να ικανοποιεί τις απαιτήσεις, δηλαδή εκτελεί τις επιθυμητές λειτουργίες του λογισμικού με τρόπο ικανοποιητικό και να γίνεται αποδεκτό από τον πελάτη. Από το σημείο αυτό και μετά μπορούν να προστεθούν και οι υπόλοιπες λειτουργίες, ώστε το λογισμικό να ολοκληρωθεί [1,3].



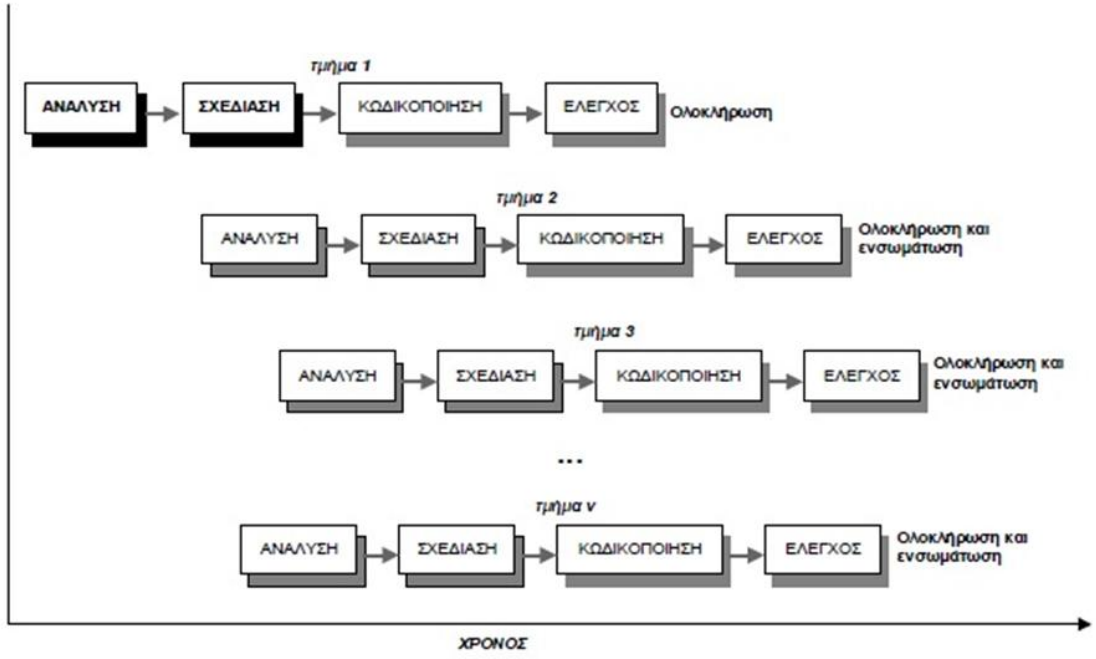

**Σχήμα 4.** Διαγραμματική απεικόνιση μοντέλου πρωτοτυποποίησης

Ένα σημαντικό πλεονέκτημα του μοντέλου αυτού, είναι η δυνατότητα απόκτησης άποψης για την εφαρμογή Λογισμικού νωρίτερα απ' ότι στο μοντέλο του καταρράκτη. Αυτό μπορεί να γλιτώσει την ανάπτυξη από καθυστερήσεις (και συνεπαγόμενα κόστη) ή ακόμη και από ολική αποτυχία, τα οποία θα επέρχονταν αν ο κατασκευαστής αναγκάζετε να οπισθοδρομήσει την ανάπτυξη ενώ αυτή είχε προχωρήσει πολύ. Παράλληλα, ιδιαίτερη σημασία αποκτά η διοίκηση του έργου η οποία πρέπει να εξασφαλίζει την υλοποιησιμότητα του πρωτοτύπου και την εύκολη τροποποίησή του. Κάθε κατασκευή πρωτοτύπου μπορεί να θεωρηθεί ως ένα μικρό έργο Λογισμικού το οποίο κατασκευάζεται με διαδικασίες που μπορούν να ακολουθούν άλλα μοντέλα κύκλου ζωής, όπως αυτό του καταρράκτη.

Με βάση τις παραπάνω παρατηρήσεις το μοντέλο πρωτοτυποποίησης χρησιμοποιείται στην ανάπτυξη εφαρμογών λογισμικού για τις απαιτήσεις από τις οποίες δεν υπάρχει βεβαιότητα στην αρχή της ανάπτυξης, οπότε δεν μπορούν να συμφωνηθούν και να παγιοποιηθούν. Τέτοιες είναι εφαρμογές που κατασκευάζονται για πρώτη φορά ή που είναι στενά εξαρτημένες από τον πελάτη, χωρίς να υπάρχει αποδεκτό προηγούμενο παράδειγμα. Ωστόσο, το μέγεθος των εφαρμογών αυτών δεν μπορεί να είναι ιδιαίτερα μεγάλο, διότι ο χρόνος ανάπτυξης κάθε πρωτοτύπου μεγαλώνει και η απαιτούμενη ευελιξία μειώνεται [2,4].

### 1.1.3 Το μοντέλο της λειτουργικής επαύξησης

Το μοντέλο της λειτουργικής επαύξησης (Incremental Model) συνδυάζει την ακολουθιακή ανάπτυξη του μοντέλου του καταρράκτη και την τμηματική ανάπτυξη του μοντέλου της πρωτοτυποποίησης. Κεντρική ιδέα είναι η κατάτμηση του υπό κατασκευή Λογισμικού σε τμήματα που αναπτύσσονται ανεξάρτητα, ακολουθώντας το καθένα ακολουθιακή ανάπτυξη σύμφωνα με το μοντέλο του καταρράκτη. Κατά την αρχική φάση ανάλυσης και σχεδίασης, αποφασίζονται τα τμήματα στα οποία θα κατατμηθεί η εφαρμογή, η ανάπτυξη των οποίων γίνεται στη συνέχεια ανεξάρτητα και παράλληλα. Όταν ολοκληρώνεται η ανάπτυξη κάθε τμήματος, αυτό ενσωματώνεται στο σύνολο της εφαρμογής, διαδικασία η οποία δικαιολογεί και τον τίτλο "λειτουργική επαύξηση".




**Σχήμα 5.** Διαγραμματικά το μοντέλο της λειτουργικής επαύξησης

Πλεονεκτήματα της ιδέας είναι η δυνατότητα παράλληλης ανάπτυξης, η οποία τελικά καταλαμβάνει μικρότερο χρόνο καθώς και ο διαδοχικός εμπλουτισμός των λειτουργικών χαρακτηριστικών του Λογισμικού. Τα βασικά μειονεκτήματα του μοντέλου είναι τα ακόλουθα:

- Η αρχική κατάτμηση και γενική σχεδίαση του συστήματος αποκτά ιδιαίτερη βαρύτητα. Σφάλματα σε αυτή μπορούν να έχουν σημαντικές επιπτώσεις στο Λογισμικό που θα κατασκευαστεί στη συνέχεια.
- Σε περίπτωση μεταβολής των λειτουργικών απαιτήσεων κατά τη χρήση του ημιτελούς συστήματος, μπορεί η αρχιτεκτονική αυτού να μεταβληθεί σε βαθμό που να κλονιστεί η ανάπτυξη των υπολοίπων τμημάτων αυτού.

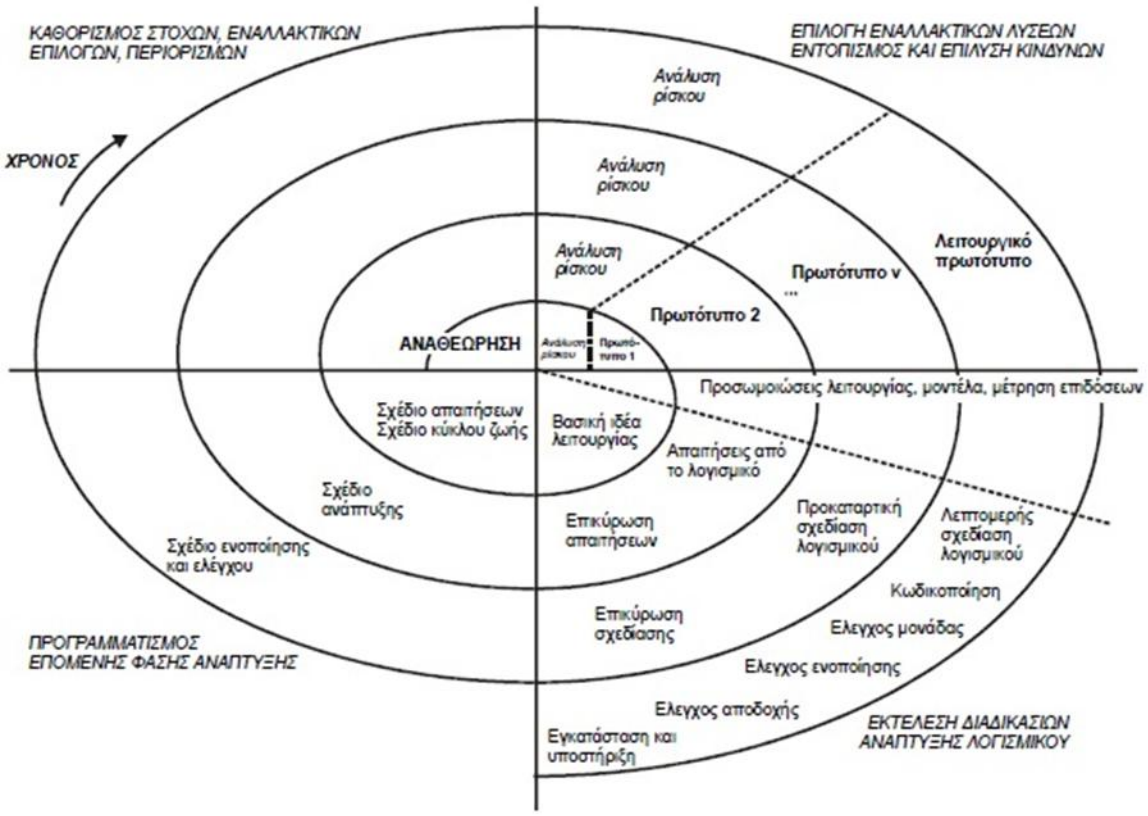
Το μοντέλο της λειτουργικής επαύξησης χρησιμοποιείται στην ανάπτυξη μεγάλων εφαρμογών λογισμικού για τις οποίες ισχύουν οι απαιτήσεις του μοντέλου του καταρράκτη, δηλαδή σαφής γνώση και μικρή ή καθόλου μεταβλητότητα των απαιτήσεων κατά την ανάπτυξη.

### 1.1.4 Το 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 προτυποποίησης 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 όρους, συχνά δε σε συνολική αποτυχία των έργων [1,3].

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 που πρόκειται για μια γενίκευση των μοντέλων της λειτουργικής επαύξησης και της προτυποποίησης, με σημαντικά νέα στοιχεία:

- 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.




**Σχήμα 6.** Διαγραμματική απεικόνιση του μοντέλου σπειροειδούς ανάπτυξης

Όπως φαίνεται και στο Σχήμα 6, στο σπειροειδές μοντέλο διακρίνονται τέσσερις κατηγορίες εργασιών:

**Προσδιορισμός στόχων**. 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.

**Εντοπισμός και επίλυση κινδύνων.** 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 και για την κατασκευή ή όχι πρωτοτύπου.

**Εκτέλεση διαδικασιών ανάπτυξης και επαλήθευση**. Η εκτέλεση των βημάτων της διαδικασίας ανάπτυξης λογισμικού που έχει επιλεγεί για το τμήμα εκείνο του συστήματος που αφορά η τρέχουσα επανάληψη

**Εργασίες προγραμματισμού**. Μετά την επαλήθευση των αποτελεσμάτων γίνεται προγραμματισμός της συνέχισης της ανάπτυξης.

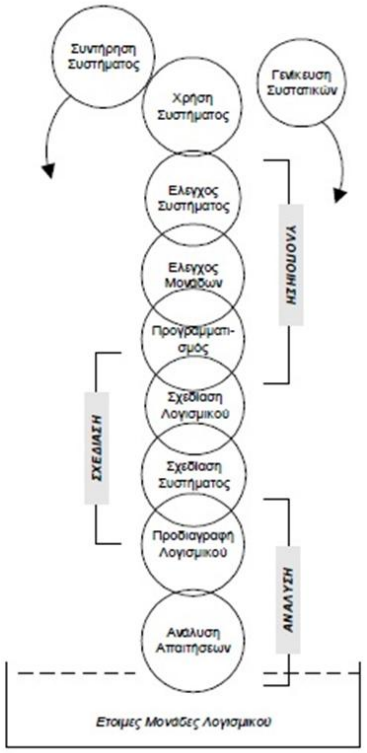
Το σπειροειδές μοντέλο δεν καθορίζει εκ των προτέρων ποιές ακριβώς είναι οι εργασίες ανάπτυξης λογισμικού που πρέπει να γίνουν, ούτε σε ποια έκταση του συστήματος αυτές θα εφαρμοστούν. Διαφορετικές διαδικασίες ανάπτυξης μπορεί να επιλεγούν για διαφορετικά τμήματα του Λογισμικού. Αυτό που προτείνει, είναι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

Το εξελικτικό και το σπειροειδές μοντέλο αποτελούν συνήθως μια μέση λύση ανάμεσα στο μοντέλο καταρράκτη και πρωτοτυποίησης [4].

### 1.1.5 Το μοντέλο του πίδακα

Αρκετά μοντέλα κύκλου ζωής που έχουν προταθεί αποτελούν παραλλαγές αυτών που αναφέρθηκαν, τα χαρακτηριστικά των οποίων υποβάλλονται από τις μεθοδολογίες ανάπτυξης. Οι πρώτες προσεγγίσεις του θέματος με βάση την αντικειμενοστραφή (object-oriented) τεχνολογία διαφοροποίησαν το παραπάνω σχήμα βασιζόμενες σε δύο ιδιαίτερα γνωρίσματά της: πρώτον, ότι οι έννοιες ανάλυση-σχεδίαση-κωδικοποίηση έρχονται στο αντικειμενοστραφές παράδειγμα πολύ πιο κοντά και δεύτερον, ότι το αποτέλεσμα κάθε διαδικασίας κατασκευής Λογισμικού είναι όχι μόνο ένα σύστημα, αλλά και επαναχρησιμοποιήσιμες μονάδες οι οποίες μπορούν να χρησιμοποιηθούν από τις πρώτες φάσεις της ανάπτυξης μελλοντικών συστημάτων. Με τον τρόπο αυτό προέκυψε το μοντέλο του πίδακα (fountain model) που φαίνεται στο Σχήμα 7.




**Σχήμα 7.** Το μοντέλο κύκλου ζωής του πίδακα το οποίο βασίζεται στην αντικειμενοστραφή τεχνολογία ανάπτυξης Λογισμικού

Κατά την ανάπτυξη παρατηρούνται επικαλύψεις των φάσεων ανάλυση – σχεδίαση – κωδικοποίηση, οι οποίες φαίνονται με την επικάλυψη των κύκλων στο σχήμα. Κατά το τέλος της ανάπτυξης, ορισμένα από τα συστατικά Λογισμικού που έχουν παραχθεί ενσωματώνονται σε μια "δεξαμενή" συστατικών και διατίθενται για να χρησιμοποιηθούν στην ανάπτυξη και νέων συστημάτων. Η ιδέα του μοντέλου κύκλου ζωής του πίδακα τονίζει περισσότερο τα επιθυμητά χαρακτηριστικά της μεθοδολογίας κατασκευής του Λογισμικού σύμφωνα με την αντικειμενοστραφή λογική, ήταν δε αρκετά επίκαιρη κατά την έκρηξη ενδιαφέροντος για την αντικειμενοστραφή τεχνολογία στα τέλη της δεκαετίας του 80 και στις αρχές της δεκαετίας του 90.

### 1.1.6 Γενικά μοντέλα κύκλου ζωής λογισμικού

Μεταγενέστερα μοντέλα κύκλου ζωής Λογισμικού προσπαθούν να δώσουν μια γενική κατεύθυνση εφαρμογής των υπαρχουσών ιδεών, αφήνοντας σημαντικούς βαθμούς ελευθερίας στον κατασκευαστή που τα ακολουθεί. Αυτό είναι ιδιαίτερα επιθυμητό διότι η αυστηρή πειθαρχία που επιχειρήθηκε να εισαχθεί τα πρώτα χρόνια της έκρηξης της χρήσης του Λογισμικού δε συμβάδιζε με την ωριμότητα σκέψης που διέθετε η τεχνική κοινότητα την εποχή εκείνη, ούτε και μπορούσε να παρακολουθήσει τους υψηλούς ρυθμούς εξελίξεων στο χώρο της πληροφορικής. Χαρακτηριστικό είναι ότι συχνά ένα πολύ μεγάλο μέρος ογκωδέστατων παραδοτέων (σχεδίων και προδιαγραφών) δεν ήταν παρά λευκές σελίδες με την μόνη ένδειξη "this page has been intentionally left blank" οι οποίες όμως ήταν υποχρεωτικό να υπάρχουν

σύμφωνα με το ακολουθούμενο μοντέλο ανάπτυξης. Η πειθαρχία αυτή τελικά δεν οδήγησε στην κατασκευή Λογισμικού αναμενόμενης ποιότητας.

Μια περιγραφή ενός σύγχρονου μοντέλου κύκλου ζωής Λογισμικού περιέχει μόνο γενικές κατευθύνσεις οι οποίες εξειδικεύονται στο εκάστοτε περιβάλλον ανάπτυξης, πρόβλημα, κλπ. Επίσης, δεν είναι άρρηκτα συνδεδεμένο με κάποια μεθοδολογία ανάπτυξης Λογισμικού, αλλά μπορεί να εξειδικευτεί για την πρακτική του κάθε κατασκευαστή. Ένα τέτοιο μοντέλο φαίνεται στο Σχήμα 8 και μπορεί να χαρακτηριστεί ως απόγονος πολλών από τα μοντέλα που προαναφέρθηκαν.


**Σχήμα 8.** Γενικό μοντέλο κύκλου ζωής το οποίο ενσωματώνει χαρακτηριστικά πολλών από τα μοντέλα που αναφέρθηκαν

Το γενικό πλαίσιο του μοντέλου αυτού περιλαμβάνει τις φάσεις σύλληψης, κατασκευής και λειτουργίας. Κάθε μια από αυτές αναλύεται σε επιμέρους εργασίες, σύμφωνα με τα χαρακτηριστικά του εκάστοτε περιβάλλοντος. Ιδιαίτερα η γενική φάση της κατασκευής αναλύεται σε "κύκλους ανάπτυξης" καθένας εκ των οποίων προσθέτει νέα χαρακτηριστικά και λειτουργίες στο υπό κατασκευή Λογισμικό.

Τα επιμέρους βήματα μέσα σε κάθε κύκλο ανάπτυξης μοιάζουν με τα βήματα του μοντέλου του καταρράκτη, μόνο που δεν εφαρμόζονται για ολόκληρο το σύστημα, αλλά για το μικρό μέρος του που κατασκευάζεται στον εν λόγω κύκλο, όπως στο μοντέλο της προτυποποίησης. Για την εκκίνηση κάθε κύκλου ανάπτυξης μπορεί να έχει προηγηθεί ανάλυση ρίσκου και σκοπιμότητας όπως στο σπειροειδές μοντέλο. Ζητήματα όπως η αλληλουχία των ενεργειών και ο ακριβής καθορισμός των κύκλων ανάπτυξης αφήνονται στη διακριτική ευχέρεια του κάθε κατασκευαστή από τον οποίο και καθορίζονται

σύμφωνα με τις ιδιαιτερότητες κάθε περίπτωσης. Ένα πραγματικό τέτοιο πλαίσιο ανάπτυξης προτείνεται από την μεθοδολογία Objectory η οποία είναι το προϊόν σύγκλισης των επικρατέστερων αντικειμενοστραφών μεθοδολογιών ανάπτυξης Λογισμικού.

Τα μοντέλα κύκλου ζωής Λογισμικού που έχουν παρουσιαστεί διακρίνονται σε ακολουθιακά και σε επαναληπτικά. Στα ακολουθιακά μοντέλα η ανάπτυξη γίνεται σε διαδοχικές διακριτές φάσεις και για ολόκληρο το σύστημα λογισμικού, ενώ στα επαναληπτικά η ανάπτυξη του λογισμικού γίνεται σε τμήματα. Χαρακτηριστικότερο ακολουθιακό μοντέλο είναι αυτό του καταρράκτη, ενώ το γενικότερο από τα επαναληπτικά είναι το σπειροειδές. Πρακτικά χρησιμότερα στην πράξη είναι τα μοντέλα κύκλου ζωής που αφήνουν ελευθερία εξειδίκευσης στις εκάστοτε συνθήκες και δεν προσδιορίζουν με αυστηρότητα τις ενέργειες που πρέπει να γίνουν, τα προϊόντα κλπ. Δεν υπάρχει ένα "καλύτερο" μοντέλο κύκλου ζωής, αλλά ένα καταλληλότερο στις εκάστοτε συνθήκες τόσο του κατασκευαστή, όσο και του θεματικού πεδίου της εφαρμογής λογισμικού. Στον Πίνακα 1 φαίνονται συνοπτικά ορισμένα χαρακτηριστικά των μοντέλων κύκλου ζωής λογισμικού [1].

| **Μοντέλο** | **Μέγεθος εφαρμογών** | **Μεταβολές στις απαιτήσεις** | **Προσαρμοστικότητα στον κατασκευαστή** | **Διάσοση** |
|---|---|---|---|---|
| **Καταρράκτη** | Μικρό έως μεσαίο | Ανεπιθύμητες | Καμμία | Μεγάλη τάση μείωσης |
| **Πρωτυποποίησης** | Μικρό έως μεσαίο | Δεκτές | Μικρή | Μικρή με τάση αύξησης |
| **Λειτουργικής επαύξησης** | Μεσαίο ως μεγάλο | Ανεπιθύμητες | Καμία | Μικρή με τάση μείωσης |
| **Σπειροειδές** | Μεσαίο ως μεγάλο | Δεκτές | Αρκετή | Μικρή με τάση μείωσης |
| **Πίδακα** | Οποιοδήποτε | Δεκτές | Αρκετή | Μικρή |
| **Γενικό** | Οποιοδήποτε | Δεκτές | Μεγάλη | Μικρή με ισχυρές τάσεις αύξησης |

**Πίνακας 1**.Συνοπτική παράθεση χαρακτηριστικών των μοντέλων κύκλου ζωής λογισμικού

Ενας συστηματικός τρόπος περιγραφής των ενεργειών και των προϊόντων Λογισμικού διακρίνει τρία επίπεδα λεπτομέρειας καθένα από τα οποία περιέχει διαφορετικές πληροφορίες για τις ενέργειες ανάπτυξης Λογισμικού και τα προϊόντα τους. Με αυτό τον τρόπο είναι δυνατή μια συστηματική περιγραφή των δραστηριοτήτων και των συστατικών λογισμικού.

### 1.1.7 Εναλλακτικά Μοντέλα Ανάπτυξης

Τα εναλλακτικά μοντέλα έχουν προταθεί ως μια εναλλακτική λύση στα συμβατικά με σκοπό να ελαττώσουν τα προβλήματα και τις αδυναμίες που έχουν τα συμβατικά μοντέλα. Τα εναλλακτικά μοντέλα ανάπτυξης είναι τα εξής [3]:

***Λειτουργικό Μοντέλο***

Το Λειτουργικό Μοντέλο (Operational Model) χρησιμοποιεί αποκλειστικά τις προδιαγραφές που είναι λειτουργικές και περιγράφουν τι θα κάνει το σύστημα έμμεσα. Έτσι, δημιουργείται μια περιγραφή η οποία δείχνει πως αυτό θα λειτουργεί η οποία υλοποιείται σε κάποια γλώσσα ώστε να μπορεί να εκτελεστεί. Έτσι μπορεί να γίνει φανερή η συμπεριφορά του τελικού συστήματος και ως εκ τούτου να αξιολογηθεί και να βελτιωθεί εάν απαιτείται.

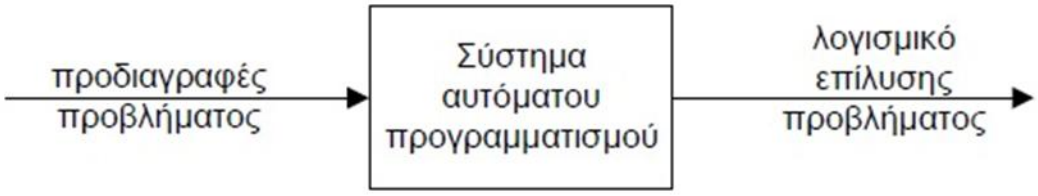
Οι λειτουργικές προδιαγραφές που χρησιμοποιούνται σύμφωνα με το μοντέλο αυτό είναι ένα είδος πρωτότυπου στο οποίο είναι εμφανής όλη η λειτουργική συμπεριφορά του συστήματος, χρησιμοποιώντας διαφορετικά μέσα από αυτά που θα χρησιμοποιούσε το τελικό σύστημα.

Αξιολογώντας τη συμπεριφορά του συστήματος, οι χρήστες μπορούν να κάνουν παρατηρήσεις και αλλαγές στις λειτουργικές προδιαγραφές. Ο κύκλος αυτός αξιολόγησης – αλλαγών επαναλαμβάνεται έως ότου θεωρηθεί ότι το σύστημα έχει την επιθυμητή λειτουργικότητα. Έτσι ολοκληρώνεται η φάση των απαιτήσεων και στην συνέχεια μπορεί να χρησιμοποιηθεί για παράδειγμα το μοντέλο του καταρράκτη από την φάση της σχεδίασης και κάτω.

Βασικό πρόβλημα του μοντέλου αποτελεί το γεγονός ότι χρησιμοποιεί εκτελέσιμες γλώσσες προδιαγραφών (executable specification languages) που είναι αυστηρά τυπικές (formal) και απαιτούν ιδιαίτερες γνώσεις από την ομάδα ανάπτυξης αλλά και εργαλεία υλοποίησης (compilers – interpreters, specification languages).

***Μοντέλο Αυτόματου Προγραμματισμού***

Το μοντέλο αυτόματου προγραμματισμού (Automatic Programming Model) βασίζεται στην ιδέα της δημιουργίας ενός συστήματος που να μπορεί να δημιουργήσει λογισμικό αυτόματα αφού πρώτα του δοθούν οι προδιαγραφές του προβλήματος.




**Σχήμα 9.** Σύστημα αυτόματου προγραμματισμού

Η αυτοματοποιημένη δημιουργία λογισμικού είναι μια παλιά ιδέα που χρησιμοποιήθηκε και για την δημιουργία μεταφραστών γλωσσών προγραμματισμού (μετα-μεταφραστές - meta-translators). Το μοντέλο αυτόματου προγραμματισμού ενδείκνυται για τις περιπτώσεις που οι απαιτήσεις του συστήματος είναι σαφώς καθορισμένες ώστε να μπορούν να περιγραφούν με ένα

πολύ τυπικό τρόπο όπως είναι οι γραμματικές χωρίς συμφραζόμενα. Τα εργαλεία που χρησιμοποιούνται (μετα-μεταφραστές και οι αντίστοιχες γραμματικές) απαιτούν υψηλές γνώσεις πληροφορικής και επομένως πολύ εξειδικευμένη και έμπειρη ομάδα ανάπτυξης.

***Μοντέλο Επαναχρησιμοποίησης Λογισμικού***

Με το μοντέλο επαναχρησιμοποίησης λογισμικού (software reusability model) γίνεται χρήση ήδη υπάρχοντος και δοκιμασμένου λογισμικού, σχεδίων και κώδικα. Οι υπάρχουσες ψηφίδες λογισμικού (με ελεγμένη ορθότητα) ενσωματώνονται σε νέα προϊόντα λογισμικού [3].

Η διαδικασία αυτή δεν είναι εύκολη, αφού παρουσιάζονται δυσκολίες, λόγω της ανυπαρξίας εργαλείων και τεχνικών καταλλήλων για αυτή τη δουλειά, αλλά και της έλλειψης προτύπων κατασκευής ψηφίδων λογισμικού που να μπορούν να επαναχρησιμοποιηθούν.

Τα βασικά πλεονεκτήματα του μοντέλου είναι η συντόμευση του χρονικού διαστήματος κατασκευής λογισμικού αλλά και η βελτίωση της αξιοπιστίας του αφού στηρίζεται σε έτοιμα, δοκιμασμένα και άρα αξιόπιστα τμήματα λογισμικού.

Τα συστατικά λογισμικού που θα επαναχρησιμοποιηθούν μπορεί να είναι πολλών και διαφόρων μεγεθών, όπως για παράδειγμα :

- Επαναχρησιμοποίηση ολόκληρων συστημάτων εφαρμογών (Application System Reuse) τα οποία είτε ενσωματώνονται στο καινούριο σύστημα χωρίς αλλαγή είτε δημιουργούνται ολόκληρες οικογένειες εφαρμογών που μπορούν να τρέξουν σε διαφορετικές πλατφόρμες ώστε να ικανοποιήσουν συγκεκριμένες ανάγκες.
- Επαναχρησιμοποίηση ψηφίδων (componentreuse) όπου διάφορα "συστατικά" μιας εφαρμογής από ένα υποσύστημα μέχρι ένα μεμονωμένο αντικείμενο μπορούν να επαναχρησιμοποιηθούν.
- Επαναχρησιμοποίηση κάποιας λειτουργίας ή συνάρτησης (FunctionReuse) όπου συστατικά (συναρτήσεις, διαδικασίες) που υλοποιούν μια απλή λειτουργία όπως μια μαθηματική συνάρτηση επαναχρησιμοποιούνται σε άλλες εφαρμογές. Αυτού του είδους η επαναχρησιμοποίηση που βασίζεται στις βιβλιοθήκες είναι και η πιο διαδεδομένη.

Το προφανές κέρδος από την επαναχρησιμοποίηση είναι η μείωση του κόστους ανάπτυξης δεδομένου ότι λιγότερα συστατικά του συστήματος χρειάζεται να προσδιοριστούν, να σχεδιαστούν, να υλοποιηθούν και να αξιολογηθούν.

***Αντικειμενοστραφές Μοντέλο***

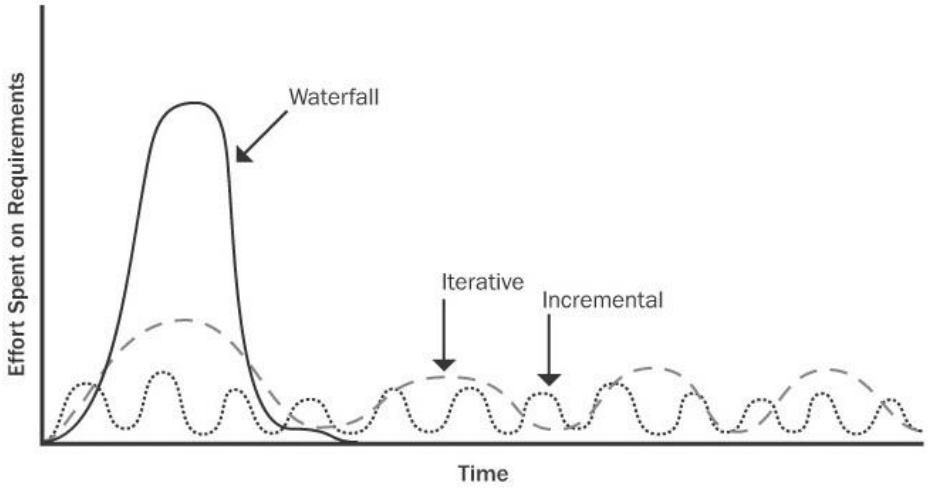
Το αντικειμενοστραφές μοντέλο (Object - Oriented Model) βασίζεται στον αντικειμενοστραφή προγραμματισμό. Αναπτύσσεται με τρόπο παρόμοιο με το μοντέλο του καταρράκτη, αλλά διαφέρει σε δύο βασικά σημεία:

- Οι διάφορες φάσεις υπερκαλύπτονται μεταξύ τους.

- Η ανάπτυξη 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.

Η ανάπτυξη των απαιτήσεων δεν γίνεται πάντα στην αρχή του κύκλου ζωής του έργου. Τα έργα που ακολουθούν ένα επανληπτικό μοντέλο κύκλου ζωής αφιερώνουν χρόνο στις απαιτήσεις κατά τη διάρκεια κάθε επανάληψης. Στα επαυξητικά μοντέλα γίνονται μικρές αλλά συχνές προσπάθειες ανάπτυξης απαιτήσεων. Στο Σχήμα 10 βλέπουμε πώς τα διαφορετικά μοντέλα κύκλου ζωής αφιερώνουν προσπάθεια στην ανάπτυξη απαιτήσεων κατά τη διάρκεια ανάπτυξης του προϊόντος.




**Σχήμα 10**. Κατανομή της προσπάθειας ανάπτυξης απαιτήσεων με την πάροδο του χρόνου, στα διαφορετικά μοντέλα κύκλου ζωής της ανάπτυξης ενός λογισμικού

## 1.2 Μεθοδολογίες Ανάπτυξης Πληροφοριακών Συστημάτων

Το πρόβλημα της σχεδίασης λογισμικού μπορεί να αντιμετωπιστεί με διάφορες στρατηγικές προσεγγίσεις. Οι διάφορες μεθοδολογίες που έχουν παρουσιαστεί μπορούν να ενταχθούν σε δύο μεγάλες κατηγορίες: ¨τις προσανατολισμένες στις διαδικασίες¨ (Function - Oriented) και τις ¨προσανατολισμένες στα αντικείμενα¨ (Object - Oriented). Μια σύντομη αναφορά στα χαρακτηριστικά των δύο κατηγοριών δίνεται στην ενότητα αυτή [5].

Η απάντηση στην ερώτηση "τι θα κάνει το λογισμικό" για μεγάλο χρονικό διάστημα δινόταν με τη μορφή μιας ιεραρχίας διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού, η οποία επιδρούσε σε ένα σύνολο ανεξάρτητων δεδομένων. Αν και από την δεκαετία του '60 ακόμα είχαν παρουσιαστεί και άλλες απόψεις, η προσέγγιση αυτή επικράτησε και πάνω της γεννήθηκαν διάφορες μεθοδολογίες σχεδίασης λογισμικού, οι οποίες συγκρότησαν την οικογένεια μεθοδολογιών δομημένης σχεδίασης.

Η "άλλη άποψη", προσπαθούσε να απαντήσει στο ίδιο ερώτημα χωρίς να διακρίνει τα δεδομένα από τις διαδικασίες, προτείνοντας ένα σύνολο συνεργαζόμενων οντοτήτων που περιλαμβάνουν στο ίδιο κέλυφος και δεδομένα και διαδικασίες. Οι οντότητες αυτές ονομάστηκαν "αντικείμενα". Οι μεθοδολογίες που γεννήθηκαν πάνω σε αυτή την προσέγγιση ονομάστηκαν "προσανατολισμένες-στα αντικείμενα". Και στις δύο περιπτώσεις, ο σχεδιαστής καταπιάνεται και με τις διεργασίες και με τα δεδομένα, αποδίδοντάς τους, όμως, διαφορετική βαρύτητα σε κάθε περίπτωση. Οι δύο οικογένειες φαίνονται και στο Σχήμα 10.


**Σχήμα 11.**Ταξινόμηση των τεχνοτροπιών σχεδίασης λογισμικού

### 1.2.1 Η μεθοδολογία της αφαίρεσης

Η μεθοδολογία της αφαίρεσης είναι ένας από τους τρόπους διαχείρισης της πολυπλοκότητας. Επιχειρούμε να αγνοήσουμε πληροφορίες που προς στιγμήν μας ενδιαφέρουν και να εστιάσουμε σε εκείνες τις πληροφορίες που κρίνουμε σημαντικές. Η μεθοδολογία αυτή θα πρέπει να χρησιμοποιείται με τρόπο που δεν αλλοιώνει το πρόβλημα μας. Αλλά θα πρέπει να εστιάσουμε μόνο στο υποσύνολο εκείνο της πολυπλοκότητας που μας ενδιαφέρει κάθε στιγμή.

### 1.2.2 Η μεθοδολογία της Μοντελοποίησης

Η μοντελοποίηση είναι ένας άλλος τρόπος διαχείρισης της πολυπλοκότητας. Προσφέρει έναν αποδοτικό τρόπο για να κατανοήσουν οι εμπλεκόμενοι στην ανάπτυξη και την ανάλυση τι και πώς θα αναπτύξουν και θα συνεννοηθούν μεταξύ τους. Στα θετικά της μεθοδολογίας εντάσσεται το γεγονός ότι οι εμπλεκόμενοι επιβεβαιώνουν ότι το σύστημα που θα αναπτυχθεί ανταποκρίνεται στις απαιτήσεις τους και το ότι επιτρέπει μελλοντικές αλλαγές απαιτήσεων. Έχουν αναπτυχθεί πολλές προσεγγίσεις μοντελοποίησης οι οποίες κατατάσσονται σε δύο μεγάλες κατηγορίες. Σε εκείνες που βλέπουν το λογισμικό ως ένα σύνολο δεδομένων και λειτουργιών που επεξεργάζονται αυτά τα δεδομένα, οι λεγόμενες και Δεδομένο – Λειτουργικές προσεγγίσει; και σε εκείνες που βλέπουν το λογισμικό ως ένα σύνολο αντικειμένων που αλληλεπιδρούν μεταξύ τους μέσω μηνυμάτων, οι Αντικειμενοστραφείς προσεγγίσεις.

***Δομημένη σχεδίαση***

Οι μεθοδολογίες που ακολουθούν αυτή την προσέγγιση προτείνουν τρόπους αποσύνθεσης του συστήματος από πάνω προς τα κάτω (Top - Down) σε μια ιεραρχία διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού. Όσο κατεβαίνει κανείς στην ιεραρχία αυτή, τόσο μεγαλύτερη λεπτομέρεια συναντά, μέχρις ότου φτάσει στις απλές δομικές μονάδες, δηλαδή τις εντολές της γλώσσας προγραμματισμού.

Γνωστές μεθοδολογίες που ανήκουν στην οικογένεια αυτή έχουν προταθεί από πολλούς συγγραφείς και σχετικά μπορείτε να ανατρέξετε στη βιβλιογραφία. Οι περισσότερες των προσεγγίσεων αυτών επικεντρώνουν την προσοχή τους στις διαδικασίες πρώτα και μετά στα δεδομένα. Οι πιο σύγχρονες καθορίζουν τη δομή των διαδικασιών που επιδρούν πάνω στα δεδομένα με βάση τη δομή των δεδομένων αυτών και για τον λόγο αυτό χαρακτηρίζονται ως "βασισμένες στα δεδομένα" και συγγενεύουν εν μέρει με τις προσανατολισμένες στα αντικείμενα τεχνοτροπίες [1].


**Σχήμα 12.** Προϊόντα της δομημένης ανάλυσης και σχεδίασης

Στη δομημένη ανάλυση, οι απαιτήσεις από το λογισμικό περιγράφονται με τη βοήθεια δομημένου κειμένου, καθώς και με ένα διάγραμμα (μοντέλο παράστασης λογισμικού), το οποίο ονομάζεται διάγραμμα ροής δεδομένων. Το διάγραμμα αυτό είναι κεντρικό στοιχείο όλων των προσεγγίσεων της οικογένειας της δομημένης ανάλυσης και σχεδίασης και χρησιμεύει ως βάση για την αποκάλυψη της ιεραρχίας μονάδων λογισμικού που αναφέραμε. Από το διάγραμμα ροής δεδομένων, ακολουθώντας κάποια βήματα, φτάνουμε στο διάγραμμα δομής προγράμματος. Το επίπεδο λεπτομέρειας του διαγράμματος δομής προγράμματος, είναι αντίστοιχο με εκείνο του διαγράμματος ροής δεδομένων από το οποίο προήλθε. Στο διάγραμμα αυτό, περιγράφεται η

ιεραρχία των μονάδων που αποτελούν την εφαρμογή λογισμικού, δηλαδή το ποιες είναι οι μονάδες αυτές, καθώς και η επικοινωνία μεταξύ τους, δηλαδή οι κλήσεις που κάνει κάθε μονάδα σε άλλες και οι παράμετροι που ανταλλάσσονται κατά τις κλήσεις αυτές.

Προκειμένου να μπορεί να κατασκευαστεί ο πηγαίος κώδικας, πρέπει να γίνει μια κατά το δυνατόν λεπτομερής περιγραφή κάθε μονάδας λογισμικού που περιέχεται στο διάγραμμα δομής προγράμματος. Η περιγραφή αυτή γίνεται με τη βοήθεια κάποιας υποθετικής και άτυπης "γλώσσας προγραμματισμού", η οποία δε διαθέτει αυστηρότητα στη σύνταξη και τη γραμματική και αναφέρεται ως ψευδοκώδικας. Τα ενδιάμεσα αυτά προϊόντα έχουν ιδιαίτερη σημασία στη δομημένη ανάλυση και σχεδίαση, αποτελώντας τη ραχοκοκαλιά της σχεδίασης του λογισμικού, όπως φαίνεται στο παραπάνω σχήμα.

Στο θεωρητικό επίπεδο, η δομημένη ανάλυση και σχεδίαση παρουσιάζει μια εγγενή αδυναμία στην απεικόνιση των οντοτήτων του πραγματικού κόσμου σε συστατικά λογισμικού. Επιχειρεί να αποσυνθέσει ένα πρόβλημα σε μια ιεραρχία συστατικών λογισμικού, καθένα εκ των οποίων επιτελεί ένα μικρό μέρος της λύσης του. Αυτή η εφαρμογή της αρχαίας ρήσης "διαίρει και βασίλευε", έχει νόημα όταν η λύση του προβλήματος είναι μόνο υπόθεση αριθμητικών υπολογισμών. Στον πραγματικό κόσμο, όμως, και ιδιαίτερα στο πεδίο εφαρμογών λογισμικού που σχετίζονται με την επιχειρηματική δραστηριότητα, αυτή η προσέγγιση υστερεί για δύο λόγους:

- Πρώτον, δε λαμβάνει υπόψη τα δεδομένα, τα οποία έχουν τη δική τους πολύπλοκη δομή και εξαρτήσεις. Τα δεδομένα στη δομημένη ανάλυση και σχεδίαση είναι ανεξάρτητα από τις λειτουργικές μονάδες που επιδρούν σε αυτά, πράγμα που δεν ισχύει στο επίπεδο του πραγματικού κόσμου, όπου η διαχείριση δεδομένων δεν είναι ανεξάρτητη από αυτά.
- Δεύτερον, το υπολογιστικό μοντέλο που αποτελείται από το δίδυμο "συστατικά λογισμικού" και "ανεξάρτητα δεδομένα", δεν αντιστοιχεί σε οντότητες του πραγματικού κόσμου, δηλαδή δεν παριστά οντότητες που είναι αντιληπτές στον πραγματικό κόσμο. Με μια άλλη διατύπωση, δεν μοντελοποιεί εύκολα και φυσικά την επιχειρησιακή λογική (businesslogic). Ως εκ τούτου, είναι ανεπαρκές ως μεθοδολογικό εργαλείο για τη δόμηση λογισμικού.

Σε πρακτικό επίπεδο μπορούμε να εντοπίσουμε τα ακόλουθα προβλήματα:

- Ο προσδιορισμός των απαιτήσεων είναι το δυσκολότερο από τα βήματα της ανάπτυξης Λογισμικού. Η ακριβής περιγραφή των πολλών, πολύπλοκων και αλληλοσχετιζόμενων απαιτήσεων με χρήση μετασχηματισμών και ανεξάρτητων δεδομένων, καθώς και η αντιμετώπιση των μεταβολών ακόμα και κατά τη διάρκεια της ανάπτυξης του λογισμικού, φέρουν τη δομημένη προσέγγιση στα όριά της.
- Η διαχείριση των μοντέλων παράστασης λογισμικού, είναι μια δύσκολη και επιρρεπής σε σφάλματα εργασία. Το πλήθος, η πολυπλοκότητα και οι συσχετίσεις των συστατικών

αυτών, η μεταβολή τους με το χρόνο, αλλά και οι παρενέργειες κατά την πραγματοποίηση μεταβολών καθιστούν την εργασία αυτή ακόμα δυσκολότερη.

- Η συντήρηση του Λογισμικού έχει εξελιχθεί σε μια πολύ δύσκολη διαδικασία, που δυσκολεύει περισσότερο, καθώς αυξάνεται το μέγεθος του Λογισμικού. Έχει αναφερθεί ότι η συνεισφορά της συντήρησης στο συνολικό κόστος κατά τον κύκλο ζωής λογισμικού, μπορεί να ξεπεράσει το 50%.
- Μολονότι κατά την ανάπτυξη μιας νέας εφαρμογής ενδέχεται να αναγνωριστούν ομοιότητες με τα χαρακτηριστικά μιας υπάρχουσας, η επαναχρησιμοποίηση συστατικών λογισμικού που έχουν κατασκευαστεί με τη δομημένη ανάλυση και σχεδίαση δεν ενθαρρύνεται. Αυτό ισχύει στη γενική περίπτωση, διότι η πολυπλοκότητα και η φύση των συσχετίσεων ενός συστατικού λογισμικού – μέρους μιας εφαρμογής που κατασκευάστηκε με τη δομημένη φιλοσοφία, δεν επιτρέπουν τη γενίκευση και εύκολη επαναχρησιμοποίησή του. Ο κανόνας αυτός ισχύει ιδιαίτερα για λογισμικό που χρησιμοποιείται σε επιχειρηματικές εφαρμογές και γενικά σε εφαρμογές που σχετίζονται με τη διαχείριση δεδομένων, ενώ εξαίρεση αποτελούν οι βιβλιοθήκες μαθηματικών συναρτήσεων που αφορούν καθαρά υπολογιστικές εργασίες.

Όσα αναφέρθηκαν, με κανένα τρόπο δε σημαίνουν δύο πράγματα: πρώτον, ότι η δομημένη ανάλυση και σχεδίαση είναι "άχρηστη", "εσφαλμένη" ή κάτι τέτοιο και δεύτερον, ότι η αντικειμενοστραφής φιλοσοφία έχει τη λύση σε όλα τα προβλήματα της ανάπτυξης του λογισμικού. Όπως εξάλλου αναφέρθηκε, η συζήτηση αυτή έχει το νόημα της αντίληψης των εξελίξεων που εισήγαγαν μια νέα φιλοσοφία προσέγγισης του λογισμικού. Πολλά από τα προβλήματα που υπήρχαν στην ανάπτυξη του λογισμικού εξακολουθούν να υπάρχουν με την μία ή άλλη έννοια, ενδεχομένως σε μικρότερο βαθμό. Η αντικειμενοστραφής τεχνολογία είναι μόνο ένα καλύτερο εργαλείο στα χέρια του ανθρώπου για τη λύση των προβλημάτων, δεν είναι η ίδια η λύση των προβλημάτων.

***Αντικειμενοστραφής σχεδίαση***

Η αντικειμενοστραφής προσέγγιση ακολουθεί ένα διαφορετικό δρόμο: αντί το σύστημα να θεωρείται ως μια ιεραρχία διαδικασιών, ανεξαρτήτων από τα δεδομένα, θεωρείται ως μια συλλογή οντοτήτων, καθεμία εκ των οποίων περικλείει και διαδικασίες και δεδομένα. Η προσέγγιση βασίζεται στην ιδέα ότι στον πραγματικό κόσμο δεδομένα και διαδικασίες μπορούν να ιδωθούν ενιαία με βάση το πεδίο ευθύνης κάποιων οντοτήτων που ονομάζονται αντικείμενα. Κάθε αντικείμενο παρέχει στο περιβάλλον του ένα σύνολο υπηρεσιών της ευθύνης του. Η συνεργασία του συνόλου των αντικειμένων του πεδίου μιας εφαρμογής λογισμικού, παράγει το επιθυμητό αποτέλεσμα. Το σύστημα θεωρείται ως ένα σύνολο αντικειμένων που αλληλεπιδρούν μεταξύ τους με σκοπό την εξυπηρέτηση των χρηστών του συστήματος. Η αλληλεπίδραση αυτή γίνεται μέσω μηνυμάτων που ανταλλάσουν τα αντικείμενα μεταξύ τους. Στο μοντέλο αυτό κάθε οντότητα του χώρου του προβλήματος παριστάνεται από ένα αντικείμενο επιτυγχάνοντας έτσι μια αντιστοίχηση των οντοτήτων του χώρου του χώρου του προβλήματος με τα αντικείμενα που συνθέτουν το μοντέλο.

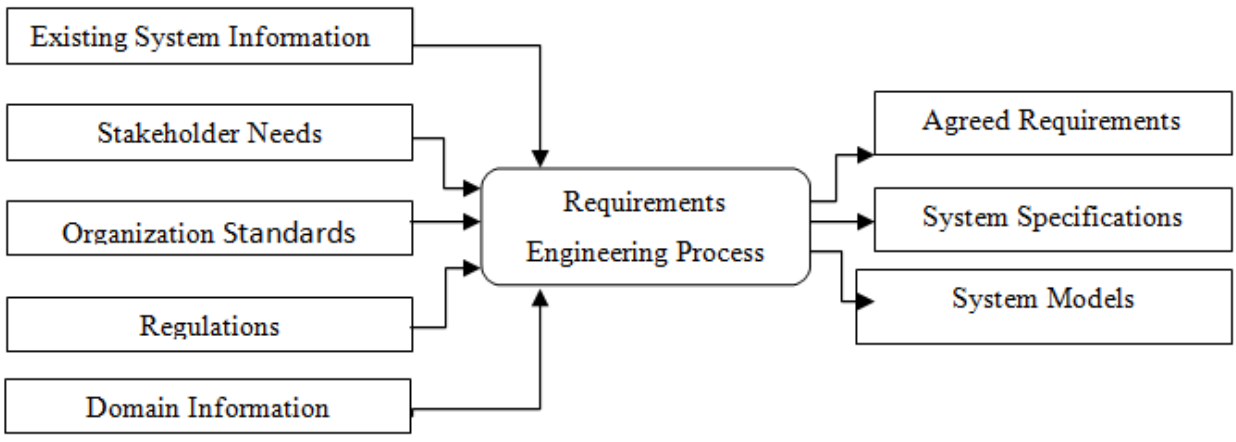
Μερικές από τις πιο γνωστές προσεγγίσεις που ανήκουν στην κατηγορία αυτή προτάθηκαν από τους Meyer, Booch, Jacobson, Rumbaugh. Για μεγάλο χρονικό διάστημα επικρατούσε μια σύγχυση σε επίπεδο ορολογίας και συμβολισμών στην οικογένεια της αντικειμενοστραφούς ανάλυσης και σχεδίασης. Τα τελευταία χρόνια η σύγχυση αυτή φαίνεται να περιορίζεται με την εμφάνιση "συγχωνευμένων" μεθοδολογιών και ενοποιημένων συμβολισμών [1,4].

# 2 Μηχανική των Απαιτήσεων

Πολλά προβλήματα λογισμικού προκύπτουν από τις ελλείψεις των τρόπων με τους οποίους οι άνθρωποι συγκεντρώνουν, τεκμηριώνουν και επικυρώνουν τις απαιτήσεις του προϊόντος. Οι περιοχές του προβλήματος μπορεί να περιλαμβάνουν την άτυπη συλλογή πληροφοριών, την αφανή λειτουργικότητα, τις εσφαλμένες υποθέσεις και τις ανεπαρκώς καθορισμένες. Σύμφωνα με μία έρευνα που έγινε από την Standish Group Study, το 13,1% των έργων αποτυγχάνουν λόγω των ελλιπών απαιτήσεων και το 8,8% λόγω των ταχέως μεταβαλλόμενων απαιτήσεων [7]. Τα δύο πιο συχνά αναφερθέντα προβλήματα σχετίζονται με τον προσδιορισμό και την διαχείριση των απαιτήσεων των πελατών. Παρόλα αυτά πολλοί οργανισμοί εφαρμόζουν μη αποδοτικές μεθόδους για αυτές τις θεμελιώδεις εργασίες του έργου. Το σύνηθες αποτέλεσμα είναι μία απόκλιση των προσδοκιών, μία διαφορά μεταξύ αυτού που η ομάδα ανάπτυξης (Developers) θεωρεί ότι πρέπει να χτίσει και του τι πραγματικά οι πελάτες έχουν ανάγκη.

¨Η Μηχανική των Απαιτήσεων είναι ο κλάδος της τεχνολογίας λογισμικού που ασχολείται με την αντιστοίχηση των στόχων των λειτουργιών και των περιορισμών των συστημάτων από τον πραγματικό κόσμο σε ένα λογισμικό. Ασχολείται επίσης με την επιρροή των παραπάνω παραγόντων στην συμπεριφορά και τις προδιαγραφές του λογισμικού [6].

Όπως φαίνεται και στο Σχήμα 13 [7], κατά την Μηχανική των Απαιτήσεων υπάρχουν ορισμένα δεδομένα εισόδου στα οποία γίνεται επεξεργασία και στην συνέχεια εξάγονται συγκεκριμένα αποτελέσματα. Τα δεδομένα εισόδου έχουν να κάνουν με πληροφορίες του υπάρχοντος συστήματος, ανάγκες ενδιαφερόμενων μερών, προτύπων του οργανισμού που θα λειτουργήσει το νέο σύστημα, κανονισμούς και πληροφορίες σχετικά με τον τομέα εφαρμογής. Αφού γίνει λοιπόν η εν λόγω επεξεργασία θα εξαχθούν οι συμφωνημένες απαιτήσεις, οι προδιαγραφές και τα μοντέλα του συστήματος [8].



**Σχήμα 13.** Είσοδος και Έξοδος της Διαδικασίας της Μηχανικής των Απαιτήσεων

Στην διαδικασία της μηχανικής των απαιτήσεων διασταυρώνονται τα συμφέροντα όλων των εμπλεκόμενων μερών σε ένα έργο λογισμικού ή συστήματος. Τα εμπλεκόμενα αυτά μέρη περιλαμβάνουν τους εξής [9]:

Πελάτες που χρηματοδοτούν ένα έργο ή αποκτούν ένα προϊόν για την ικανοποίηση των επιχειρηματικών τους στόχων.

- Χρήστες που αλληλεπιδρούν έμμεσα ή άμεσα με το προϊόν.
- Αναλυτές απαιτήσεων που καταγράφουν τις απαιτήσεις.
- Ομάδα ανάπτυξης που σχεδιάζει, τροποποιεί και συντηρείτο προϊόν.
- Τα μέλη της ομάδας ελέγχου (Testers) που αποφασίζουν αν το προϊόν συμπεριφέρεται όπως θα έπρεπε.
- Οι συγγραφείς τεκμηριώσεων παράγουν εγχειρίδια χρήσης και εκπαιδευτικό υλικό ως βοήθεια των συστημάτων.
- Διαχειριστές των έργων που θέτουν ένα πλάνο για το έργο και καθοδηγούν την ομάδα ανάπτυξης σε επιτυχή παράδοση.
- Νομικό προσωπικό που διαβεβαιώνει ότι το προϊόν συμμορφώνεται με όλες τις σχετικές νομικές και κανονιστικές διατάξεις.
- Κατασκευαστές που πρέπει να αναπτύξουν το προϊόν που περιέχει το λογισμικό.

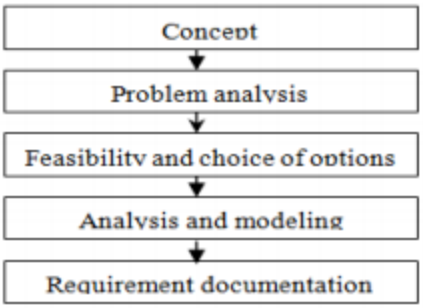
Επειδή οι απαιτήσεις είναι το θεμέλιο και της ανάπτυξης λογισμικού και των διεργασιών διαχείρισης έργου όλα τα εμπλεκόμενα μέρη θα πρέπει να δεσμευτούν σε μία αποτελεσματική διαδικασία απαιτήσεων. Η ανάπτυξη και διαχείριση των απαιτήσεων είναι δύσκολη. Δεν υπάρχουν εύκολες διαδρομές και μαγικές λύσεις [10,11].

## 2.1 Μοντέλα Διεργασιών της Μηχανικής των Απαιτήσεων

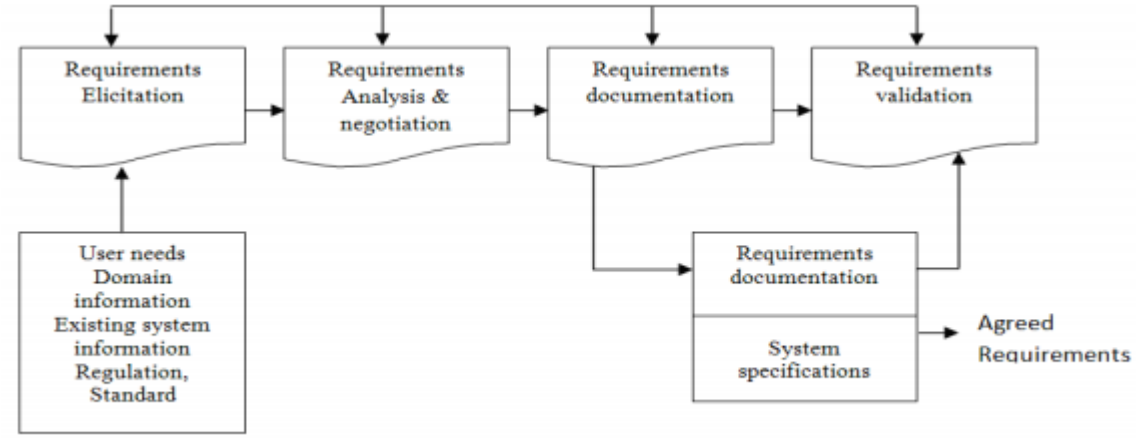
Έχουν προταθεί διάφορα μοντέλα που περιγράφουν την σειρά των εκτελούμενων διεργασιών κατά την Μηχανική των Απαιτήσεων. Οι διεργασίες και η σειρά που χρησιμοποιούνται είναι σχεδόν ίδιες σε κάθε ένα από τα παρακάτω μοντέλα, ωστόσο υπάρχουν κάποιες διαφορές, οι οποίες και περιγράφονται.

### 2.1.1 Μοντέλο Γραμμικής Διαδικασίας

Το μοντέλο της γραμμικής διαδικασία της μηχανικής των απαιτήσεων προτάθηκε από την Linda Macaulay το 1996. Υπάρχουν πέντε δραστηριότητες που εκτελούνται διαδοχικά: ο ορισμός του έργου, η ανάλυση του προβλήματος, η σκοπιμότητα και οι επιλογές, η ανάλυση και η μοντελοποίηση και η τεκμηρίωση των απαιτήσεων. Είναι ένα απλό μοντέλο και χρησιμοποιείται κυρίως για έργα με μικρό συντελεστή πολυπλοκότητας, αλλά δεν είναι ιδιαίτερα χρήσιμο όταν η πολυπλοκότητα είναι μεγάλη [12].



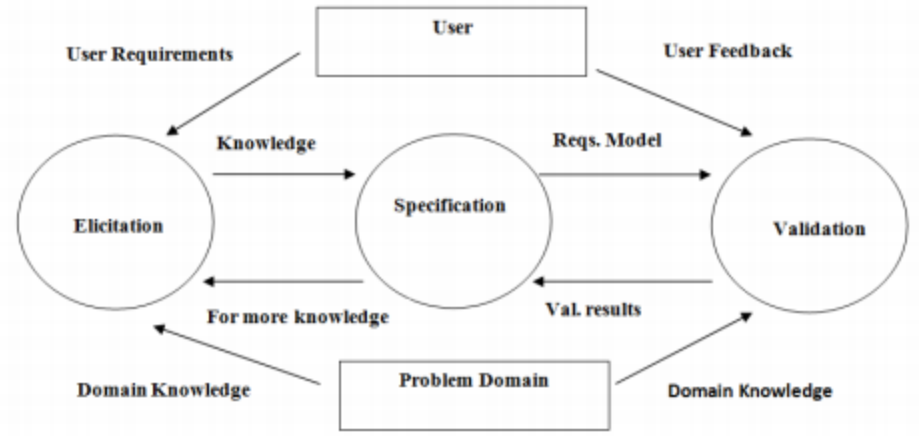

**Σχήμα 14.** Μοντέλο Γραμμικής Διαδικασίας Μηχανικής Απαιτήσεων




**Σχήμα 15**. Μοντέλο Γραμμικής Διαδικασίας Μηχανικής Απαιτήσεων

### 2.1.2 Μοντέλο Γραμμικής Επαναληπτικότητας της Μηχανικής των Απαιτήσεων

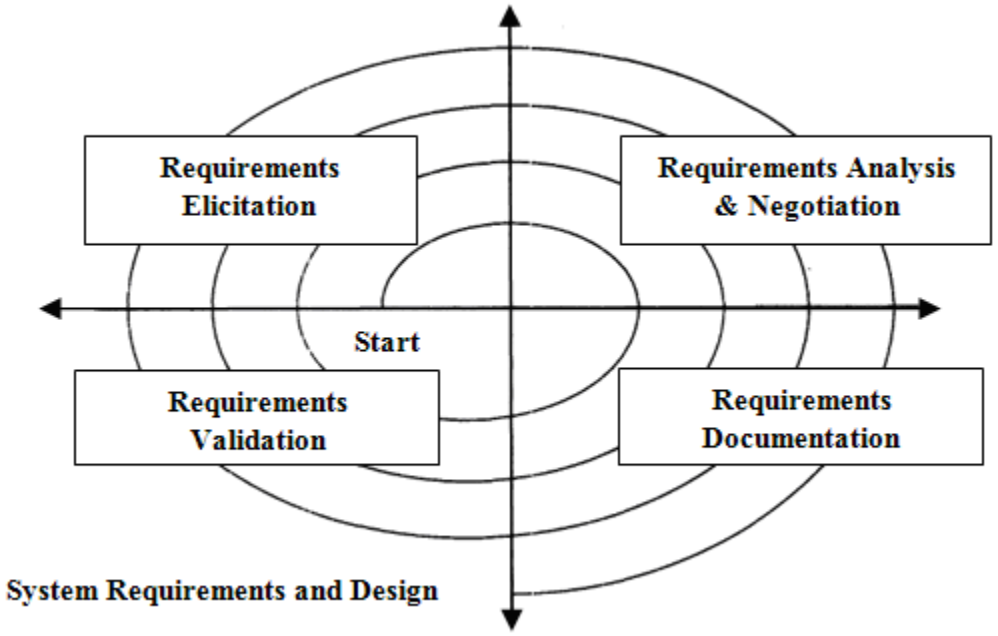
Η γραμμική επαναληπτικότητα προτάθηκε από τους Λουκόπουλο και Καρακώστα και στηρίζεται στις πολλές επαναλήψεις. Είναι κατάλληλο για έργα που δίνονται στην αγορά ανά εκδόσεις. Στο μοντέλο αυτό υπάρχουν τρεις απλές φάσεις: η εκμαίευση, η προδιαγραφή και οι επικύρωση. Οι απαιτήσεις που αναπτύσσονται προέρχονται από τους χρήστες και τα προβλήματα του χώρου για τον οποίο δημιουργείται το σύστημα [12].




**Σχήμα 16.** Μοντέλο Γραμμικής Επαναληπτικότητας της Διαδιακασίας της Μηχανικής των Απαιτήσεων

### 2.1.3 Σπειροειδές Μοντέλο

Το Σπειροειδές Μοντέλο προτάθηκε από τους Kotonya και Sommerville το 1998 και εκτελείται ανά σπείρες. Η κάθε σπείρα αντιπροσωπεύει την πλήρη έκδοση των απαιτήσεων με τις οποίες θα αναπτυχθεί το σύστημα. Κάθε σπείρα χωρίζεται σε τέσσερα τεταρτημόρια: εκμαίευση, ανάλυση και διαπραγμάτευση, τεκμηρίωση και επικύρωση. Το κύριο χαρακτηριστικό αυτού του μοντέλου είναι ότι διαχειρίζεται τις ανεπιθύμητες συνέπειες, τους κινδύνους δηλαδή όπως τις καθυστερήσεις, τις αλλαγές των απαιτήσεων και την χαμηλή απόδοση των επενδύσεων, οι οποίες μπορούν να επηρεάσουν το χρονοδιάγραμμα και την ποιότητα του έργου [7].




**Σχήμα 17.** Σπειροειδές Μοντέλο

## 2.2 Προβλήματα κατά την Μηχανική των Απαιτήσεων

Μερικά από τα συνηθισμένα προβλήματα που δημιουργούνται κατά την διάρκεια της Μηχανικής των Απαιτήσεων και οδηγούν σε μη επιτυχή παράδοση του έργου είναι:

- Δεν προσδιορίζονται σαφώς ο σκοπός του έργου και το πεδίο εφαρμογής του.
- Οι πελάτες δεν επενδύουν τον απαραίτητο χρόνο με τους αναλυτές ή την ομάδα ανάπτυξης για την ανάλυση των απαιτήσεων.
- Δεν περιγράφονται οι απαιτήσεις με ακρίβεια.
- Οι πελάτες ισχυρίζονται πως όλες οι απαιτήσεις είναι σημαντικές και δεν τις ιεραρχούν.
- Τα μάλη της ομάδας ανάπτυξης αντιμετωπίζουν ασάφειες και έτσι πραγματοποιούνελλιπή κωδικοποίηση πληροφορίας αφού πρέπει να μαντέψουν την ερμηνεία των μη σωστά τεκμηριωμένων απαιτήσεων.
- Οι πελάτες αλλάζουν συχνά τις ανάγκες τους.
- Τα μέλη της ομάδας ανάπτυξης δημιουργούν περιττή και άχρηστη λειτουργικότητα.

## 2.3 Ορισμός Απαιτήσεων Λογισμικού

Ένα πρόβλημα στην βιομηχανία λογισμικού είναι η έλλειψη κοινών ορισμών για τους όρους που χρησιμοποιούνται για την περιγραφή των πτυχών του έργου. Διαφορετικοί αναλυτές περιγράφουν την ίδια δήλωση σαν απαίτηση χρηστών, απαίτηση λογισμικού, λειτουργική απαίτηση, απαίτηση συστήματος, τεχνική απαίτηση, επιχειρηματική απαίτηση, ή απαίτηση του προϊόντος. Ο ορισμός των απαιτήσεων από τον πελάτη είναι μία υψηλού επιπέδου έννοια στο έργο. Καταλαβαίνουμε λοιπόν ότι δημιουργείται σύγχυση και δεν οδηγούμαστε με ακρίβεια στο επιθυμητό αποτέλεσμα.

***Ερμηνείες των Απαιτήσεων***

Οι απαιτήσεις είναι μία προδιαγραφή για το τί θα πρέπει να εφαρμοστεί και οδηγούν στις επιλογές του σχεδιασμού. Σύμφωνα με το πρότυπο IEEE ,το πρότυπο γλωσσάριο ορολογίας της τεχνολογίας λογισμικού, η απαίτηση ορίζεται σαν:

- Μία κατάσταση ή ικανότητα που απαιτείται από έναν χρήστη, για να λύσει ένα πρόβλημα ή να πετύχει έναν στόχο.
- Μία κατάσταση ή ικανότητα που πρέπει να πληρείται από ένα σύστημα ή από ένα στοιχείο του συστήματος για να ικανοποιήσει μία σύμβαση, ένα πρότυπο, μία προδιαγραφή.
- Μία τεκμηριωμένη αναπαράσταση μίας κατάστασης ή δυνατότητας, αυτών που αναφέρονται παραπάνω [13].

### 2.3.1 Επίπεδα Απαιτήσεων

Η ενότητα αυτή παρουσιάζει τους ορισμούς που θα χρησιμοποιηθούν για κάποιους όρους που συναντώνται συνήθως στον τομέα της μηχανικής των απαιτήσεων. Οι απαιτήσεις λογισμικού περιλαμβάνουν τρία διαφορετικά επίπεδα:

- Επιχειρηματικές απαιτήσεις
- Απαιτήσεις χρηστών
- Λειτουργικές απαιτήσεις

Οι επιχειρηματικές απαιτήσεις αντιπροσωπεύουν τους υψηλού επιπέδου στόχους του οργανισμού ή των πελατών που ζητούν το σύστημα. Συνήθως προέρχονται από τον χρηματοδότη του έργου, το τμήμα marketing ή από το όραμα του προϊόντος. Περιγράφουν γιατί ο οργανισμός θέλει να εφαρμόσει το σύστημα, πιστεύοντας πως θα επιτύχει. Οι επιχειρηματικές απαιτήσεις θα πρέπει να καταγράφονται σε ένα έγγραφο οράματος και πεδίου εφαρμογής του έργου.

Οι απαιτήσεις των χρηστών περιγράφουν τους στόχους των χρηστών ή τα καθήκοντα που θα έχουν οι χρήστες στο προϊόν. Τρόποι να εκπροσωπηθούν οι απαιτήσεις των χρηστών είναι οι περιπτώσεις χρήσης, οι περιγραφές σεναρίων και οι πίνακες απόκρισης γεγονότων. Οι ανάγκες των χρηστών περιγράφουν τι θα κάνουν οι χρήστες μέσα στο σύστημα. Θα πρέπει να ευθυγραμμίζονται με τις επιχειρηματικές απαιτήσεις

Οι λειτουργικές απαιτήσεις καθορίζουν την λειτουργικότητα του λογισμικού που πρέπει να φτιάξουν τα μέλη της ομάδας ανάπτυξης έτσι ώστε το προϊόν να επιτρέπει στους χρήστες να εκπληρώνουν τα καθήκοντά τους καλύπτοντας έτσι τις επιχειρησιακές απαιτήσεις. Μερικές φορές ονομάζονται απαιτήσεις συμπεριφοράς (Behavioural Requirements).

Ο όρος απαιτήσεις του συστήματος περιγράφει τις κύριες απαιτήσεις ενός προϊόντος που περιέχει πολλαπλά υποσυστήματα, δηλαδή ένα σύστημα. Ένα σύστημα μπορεί να είναι μόνο ένα λογισμικό ή μπορεί να περιλαμβάνει ένα συνδυασμό λογισμικού και hardware.

Οι επιχειρηματικοί κανόνες περιλαμβάνουν τις εταιρικές πολιτικές, τους κρατικούς κανονισμούς και τους διάφορους περιορισμούς που μπορεί να έχει ένας συγκεκριμένος κλάδος. Δεν αποτελούν ένα υποσύνολο απαιτήσεων αλλά περιορίζουν τα πρόσωπα που θα μπορούν να εκτελούν μία διεργασία και υπαγορεύουν την λειτουργικότητα που θα πρέπει να έχει το σύστημα ώστε να συμμορφώνεται με τις σχετικές διατάξεις. Μερικές φορές αυτοί οι κανόνες είναι αυτοί οι οποίοι ορίζουν τα χαρακτηριστικά της ποιότητας και της λειτουργικότητας. Όπως είπαμε και πριν δεν είναι απαιτήσεις, αλλά οδηγούν στην γέννηση απαιτήσεων [14,15].

Οι λειτουργικές απαιτήσεις είναι τεκμηριωμένες σε ένα έγγραφο προδιαγραφών λογισμικού. Μπορεί να είναι ένα απλό έγγραφο, μία βάση δεδομένων ή υπολογιστικά φύλλα τα οποία περιέχουν απαιτήσεις και πληροφορίες που είναι αποθηκευμένες σε ένα εμπορικό εργαλείο

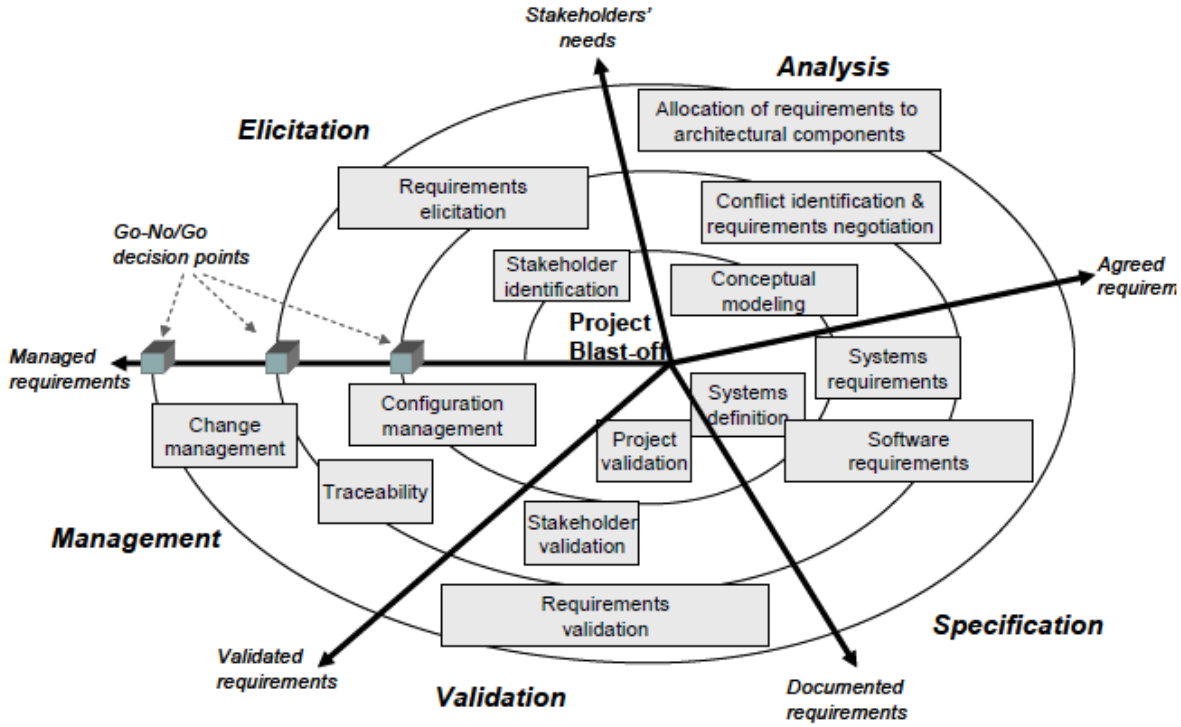
διαχείρισης απαιτήσεων. Χρησιμοποιείται στην ανάπτυξη, την δοκιμή, την διασφάλιση ποιότητας, την διαχείριση του έργου και τα συναφή.

Στο έγγραφο προδιαγραφών λογισμικού περιγράφονται και οι μη λειτουργικές απαιτήσεις. Αυτές περιλαμβάνουν τους στόχους της απόδοσης και τις περιγραφές των χαρακτηριστικών ποιότητας του λογισμικού. Τα χαρακτηριστικά ποιότητας αυξάνουν την περιγραφή της λειτουργικότητας του προϊόντος και περιλαμβάνουν αυτά την χρηστικότητα, την φορητότητα, την ακεραιότητα, την αποτελεσματικότητα και την ισχύ του συστήματος. Άλλες μη λειτουργικές απαιτήσεις περιγράφουν τις εξωτερικές διασυνδέσεις μεταξύ του συστήματος και των εξωτερικών στοιχείων που αλληλεπιδρούν με αυτό.

Ένα χαρακτηριστικό είναι ένα σύνολο από λειτουργικές απαιτήσεις που παρέχει μία δυνατότητα στον χρήστη και επιτρέπει την ικανοποίηση ενός στόχου της επιχείρησης. Στο τομέα του λογισμικού ένα χαρακτηριστικό γνώρισμα είναι μία ομάδα απαιτήσεων που βοηθά τους ενδιαφερόμενους να πάρουν μία απόφαση. Είναι μία λίστα με τα επιθυμητά χαρακτηριστικά του προϊόντος και μία περιγραφή των καθηκόντων που σχετίζονται με τους χρήστες. Μπορεί να περιλαμβάνει πολλαπλές περιπτώσεις χρήσης και πολλές απαιτήσεις που θα πρέπει να εφαρμοστούν ώστε να μπορέσει ο χρήστης να εκτελέσει μία διεργασία [10,14,16].

Υπάρχει και ένας τελευταίος τύπος απαιτήσεων, οι αντίστροφες απαιτήσεις. Περιγράφουν στην ουσία τι δεν πρέπει να κάνει το σύστημα και τους περιορισμούς στην αναμενόμενη συμπεριφορά του συστήματος. Εγγυώνται την αναμενόμενη συμπεριφορά του σε κάθε ακραία περίπτωση. Τέτοιες απαιτήσεις είναι οι απαιτήσεις ασφάλειας [17].

Παρά το γεγονός ότι πολλά μοντέλα του κύκλου ζωής παρουσιάζουν την φάση της ανάπτυξης των απαιτήσεων σαν μία εκ των προτέρων διαδικασία, η αλήθεια είναι πως αυτή επαναλαμβάνεται και συνεχίζεται καθ΄ όλη την διάρκεια του έργου [18]. Οι απαιτήσεις δεν χρησιμεύουν μόνο στον αρχικό σχεδιασμό και στην επικύρωση του συστήματος, αλλά είναι απαραίτητες και στις διαδικασίες της διαχείρισης, της αλλαγής και του ελέγχου. Η Μηχανική των Απαιτήσεων αποτελείται από πέντε βασικές διαδικασίες, την εκμαίευση, την ανάλυση, την τεκμηρίωση, την επικύρωση και την διαχείριση, στις οποίες υπάρχει ένας βαθμός επικάλυψης αλλά θα περιγραφούν στην συνέχεια σαν μεμονωμένες φάσεις.




**Σχήμα 18**. Η διαδικασία της Μηχανικής των Απαιτήσεων

Οι δραστηριότητες σε αυτές τις φάσεις είναι στενά συνδεδεμένες μεταξύ τους και συνεπώς υπάρχει μία επικάλυψη και μία επαναλαμβανόμενη διαδικασία μεταξύ τους. Το Σχήμα 18 παρουσιάζει ακριβώς αυτή την επαναληπτικότητα που παρατηρείται στις διάφορες φάσεις των [19].

## 2.4 Ανάπτυξη απαιτήσεων

Η ανάπτυξη των απαιτήσεων διαιρείται σε τέσσερις κατηγορίες. Την εκμαίευση, την ανάλυση, την προδιαγραφή και την επικύρωση. Αυτές οι υποδιαιρέσεις περιλαμβάνουν όλες τις διεργασίες που εμπλέκονται στην συλλογή, την αξιολόγηση και τεκμηρίωση των απαιτήσεων ενός λογισμικού. Περιλαμβάνει τις εξής διαδικασίες:

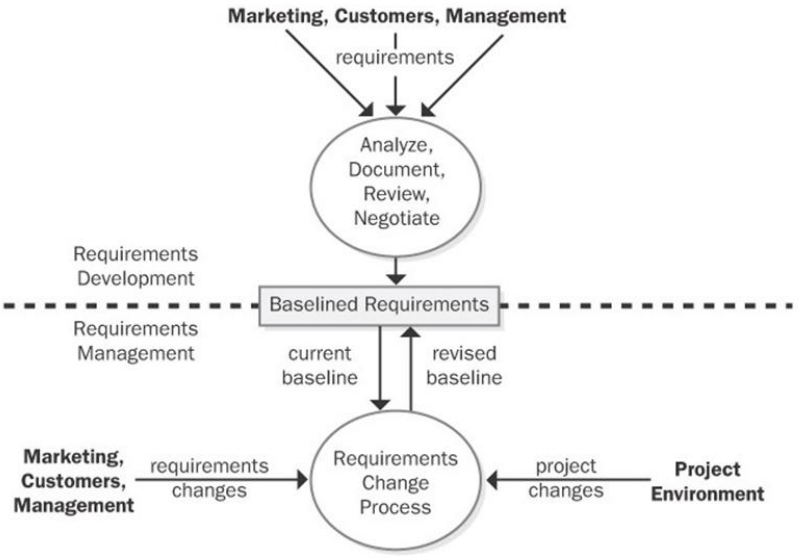
- Προσδιορισμό των κατηγοριών των χρηστών του προϊόντος.
- Εκμαίευση απαιτήσεων από άτομα που αντιπροσωπεύουν κάθε μία κατηγορία χρηστών.
- Κατανόηση των διεργασιών των χρηστών και των στόχων της επιχείρησης στην οποία θα υλοποιηθεί το σύστημα λογισμικού.
- Ανάλυση της πληροφορίας που λαμβάνεται από τους χρήστες για να διακριθούν οι λειτουργικές απαιτήσεις, οι επιχειρηματικοί κανόνες και οι προτεινόμενες λύσεις υλοποίησης.
- Προσδιορισμός των ποιοτικών χαρακτηριστικών του συστήματος.
- Ορισμός προτεραιοτήτων στις απαιτήσεις.

- Καταγραφή των προδιαγραφών των απαιτήσεων και δημιουργία μοντέλων ανάλυσης.
- Επανεξέταση, διόρθωση και επικύρωση των τεκμηριωμένων απαιτήσεων, έτσι ώστε να ξεκινήσει η υλοποίηση.

## 2.5 Διαχείριση Απαιτήσεων

Η διαχείριση απαιτήσεων είναι η θέσπιση και η διατήρηση της συμφωνίας με τον πελάτη σχετικά με τις απαιτήσεις του έργου λογισμικού. Αυτή η συμφωνία συμπεριλαμβάνεται στις γραπτές προδιαγραφές απαιτήσεων και τα μοντέλα. Η αποδοχή του πελάτη είναι μόνο η μισή εξίσωση για την αποδοχή απαιτήσεων. Θα πρέπει να αποδεχθούν τις προδιαγραφές και οι αναλυτές να συμφωνήσουν να τις χτίσουν μέσα στο προϊόν. Οι διεργασίες της διαχείρισης απαιτήσεων συμπεριλαμβάνουν:

- Καθορισμό των βασικών απαιτήσεων.
- Επανεξέταση των προτεινόμενων αλλαγών των απαιτήσεων και αξιολόγηση των επιπτώσεων κάθε πιθανής αλλαγής πριν την έγκρισή της.
- Ενσωμάτωση των επικυρωμένων αλλαγών απαιτήσεων στο έργο με ελεγχόμενο τρόπο.
- Ενημέρωση των των σχεδίων του έργου σύμφωνα με τις απαιτήσεις.
- Διαπραγμάτευση νέων δεσμεύσεων που βασίζονται στην εκτιμώμενηεπιρροή των αλλαγών των απαιτήσεων.
- Ανίχνευση των απαιτήσεων στους αντίστοιχους σχεδιασμούς, πηγαίους κώδικες και περιπτώσεις δοκιμών.
- Παρακολούθηση της κατάστασης των απαιτήσεων και των διεργασιών αλλαγής σε όλο το έργο.




**Σχήμα 19.** Η διαχωριστική γραμμή μεταξύ της ανάπτυξης και της διαχείρισης των απαιτήσεων

Το πιο δύσκολο μέρος της δημιουργίας ενός συστήματος λογισμικού είναι να αποφασίσεις τι θέλεις να δημιουργήσεις. Κανένα άλλο μέρος του έργου δεν είναι τόσο δύσκολο όσο ο καθορισμός των λεπτομερών τεχνικών προδιαγραφών, συμπεριλαμβανομένων όλων των διεπαφών των ανθρώπων, των μηχανών και των άλλων συστημάτων λογισμικού. Ο χρόνος κατανόησης των αναγκών των χρηστών είναι πολύ σημαντικός για την επιτυχία του έργου. Αν τα μέλη της ομάδας ανάπτυξης δεν έχουν γραπτές απαιτήσεις με τις οποίες να συμφωνούν οι πελάτες, τότε δεν θα μπορέσουν να τους ικανοποιήσουν [2,20].

## 2.6 Λόγοι που οδηγούν σε λανθασμένες απαιτήσεις

Συχνά είναι αδύνατο να προσδιοριστούν πλήρως οι απαιτήσεις πριν την έναρξη της κατασκευής. Σε αυτές τις περιπτώσεις χρησιμοποιούμε επαναληπτική και επαυξητική προσέγγιση για την υλοποίηση ενός μέρους των απαιτήσεων και λαμβάνουμε υπόψη την ανάδραση των πελατών πριν προχωρήσουμε στον επόμενο κύκλο. Ο μη σωστός προσδιορισμός των απαιτήσεων είναι ένα φαινόμενο που προκύπτει από τους παρακάτω λόγους:

***Ανεπαρκής συμμετοχή των χρηστών.*** Είναι ζωτικής σημασίας για ένα έργο οι χρήστες να συμμετέχουν και να κατανοούν πλήρως τις απαιτήσεις πριν αρχίσει ο σχεδιασμός και η υλοποίηση. Η όποια αλλαγή θα πρέπει να γίνεται κατά την διάρκεια της Μηχανικής των Απαιτήσεων γιατί σε διαφορετική περίπτωση θα κοστίσει πολύ περισσότερο τόσο σε πόρους, όσο και σε χρόνο. Εάν, τέλος, δεν έχουν αποσαφηνιστεί και προσδιοριστεί πλήρως οι απαιτήσεις τότε δεν μπορούμε να είμαστε σίγουροι ότι το έργο έχει τελειώσει.

***Συχνές αλλαγές των προτιμήσεων των χρηστών***. Καθώς οι απαιτήσεις αυξάνονται και εξελίσσονται κατά τη διάρκεια της ανάπτυξης τα έργα συχνά υπερβαίνουν τα προϋπολογισθέντα προγράμματα και τους προϋπολογισμούς τους. Οι συνεχής τροποποιήσεις των απαιτήσεων κάνουν το πρόβλημα ακόμα χειρότερο. Αυτό έγκειται εν μέρη στις συχνές αλλαγές των απαιτήσεων των χρηστών και στον τρόπο που ανταποκρίνεται η ομάδα ανάπτυξης σε αυτές. Θα πρέπει να γίνει σαφής δήλωση των επιχειρηματικών στόχων, του στρατηγικού οράματος και του πεδίου εφαρμογής του έργου. Η όποια αίτηση αλλαγής θα πρέπει γίνεται σύμφωνα με συγκεκριμένες διαδικασίες, να εκτιμούνται οι επιπτώσεις της στα υπόλοιπα στοιχεία του συστήματος και μετά λαμβάνεται η απόφαση για την αποδοχή ή την απόρριψη της.

***Διφορούμενες απαιτήσεις***. Ένα σύμπτωμα της αμφισημίας των απαιτήσεων είναι ότι ο αναγνώστης μπορεί να τις ερμηνεύσει με πολλούς τρόπους. Προκύπτει από ασαφείς και μη λεπτομερείς απαιτήσεις που αναγκάζουν τα μέλη της ομάδας ανάπτυξης να δώσουν την δική του ερμηνεία, που συνήθως δεν είναι και η επιθυμητή. Η αμφισημία οδηγεί σε ένα χάσμα μεταξύ των προσδοκιών των πελατών και αυτού που τελικά θα παραδοθεί.

***Πρόσθεση περιττής λειτουργικότητας (Gold Planting).*** Το φαινόμενο αυτό συμβαίνει όταν η ομάδα ανάπτυξης προσθέτει λειτουργικότητα που δεν είχε οριστεί στις προδιαγραφές απαιτήσεων. Συχνά οι χρήστες δεν ενδιαφέρονται για την παραπάνω λειτουργικότητα και ο χρόνος που χρειάστηκε για να υλοποιηθεί πηγή χαμένος. Αντί λοιπόν οι αναλυτές και η ομάδα

ανάπτυξης να εισάγουν νέα χαρακτηριστικά, θα έπρεπε να παρουσιάζουν στους πελάτες δημιουργικές ιδέες και εναλλακτικές λύσεις υπό εξέταση. Η υλοποίηση περισσότερηςλειτουργικότητας θα πρέπει να γίνεται έπειτα από την συναίνεση των πελατών. Η κάθε νέα λειτουργικότητα θα πρέπει να εμπίπτει πρώτα από όλα στο πεδίο εφαρμογής του έργου και να μην έρχεται σε σύγκρουση με τους περιορισμούς που έχουν τεθεί.

***Ελάχιστες προδιαγραφές.***Καμιά φορά η διαδικασία της τεκμηρίωσης των απαιτήσεων μπαίνει σε δεύτερη μοίρα, καθώς θεωρείται ότι απλά σπαταλά χρόνο από την υλοποίηση. Κάτι τέτοιο όμως δεν ισχύει και όπως θα δούμε παρακάτω όσο καλύτερα τεκμηριώνονται οι απαιτήσεις, τόσο γρηγορότερα υλοποιείται ένα σύστημα λογισμικού.

***Παράβλεψη Κατηγορίας χρηστών***. Τα περισσότερα προϊόντα έχουν διάφορες ομάδες χρηστών και η κάθε μία έχει διαφορετικές ανάγκες. Θα πρέπει να μην παραλειφθεί καμία κατηγορία από την διαδικασία της ανάπτυξης των απαιτήσεων και να σιγουρευτεί ότι έχουν υπολογιστεί όλες οι απαιτούμενες λειτουργικότητες.

## 2.7 Πλεονεκτήματα από μία υψηλής ποιότητας διαδικασίας Απαιτήσεων

Οι οργανισμοί που εφαρμόζουν αποτελεσματικές διαδικασίες μηχανικής απαιτήσεων μπορούν να αποκομίσουν πολλαπλά οφέλη. Μία μεγάλη ανταμοιβή έρχεται από την μείωση των επαναλήψεων κατά το τελευταίο στάδιο της διαδικασίας ανάπτυξης και από την μακρά περίοδο συντήρησης. Μία υψηλού επιπέδου έρευνα απαιτήσεων δίνει συνολική αξία στα πρώιμα στάδια του έργου.

Οι διαδικασίες απαιτήσεων σε μία συνεργατική προσέγγιση για την ανάπτυξη του προϊόντος περιλαμβάνει πολλούς φορείς που εμπλέκονται στο έργο. Συλλέγοντας απαιτήσεις επιτρέπεται στην ομάδα ανάπτυξης να καταλάβει καλύτερα την χρησιμότητα του έργου στην αγορά και αυτό αποτελεί κρίσιμο παράγοντα επιτυχίας του έργου. Είναι πολύ φτηνότερο να φτάσεις στην κατανόηση πριν το δημιουργήσεις παρά αφού το παραδώσεις στους πελάτες [19].

Κανείς δεν μπορεί να υποσχεθεί μία συγκεκριμένη απόδοση της επένδυσης από την βελτίωση της διαδικασίας των απαιτήσεων, αλλά υπάρχουν ορισμένα οφέλη που γίνονται εύκολα αντιληπτά. Αυτά είναι:

- Λιγότερες ελαττωματικές απαιτήσεις.
- Μειωμένη επανάληψη ανάπτυξης.
- Λιγότερα άσκοπα χαρακτηριστικά.
- Χαμηλότερο κόστος βελτίωσης.
- Ταχύτερη ανάπτυξη.
- Λιγότερες παρεξηγήσεις.
- Μειωμένος αλλαγές των απαιτήσεων των χρηστών.
- Πιο ακριβές σύστημα ελέγχου και εκτιμήσεων.

- Υψηλότερη ικανοποίηση πελατών και ομάδας.

## 2.8 Χαρακτηριστικά Σωστών Απαιτήσεων

Θα πρέπει πριν ξεκινήσουμε την διαδικασία της σχεδίασης να έχουμε εξασφαλίσει ότι οι απαιτήσεις που υπάρχουν στο έγγραφο προδιαγραφών λογισμικού είναι [17]:

**Ολοκληρωμένες**. Θα πρέπει να έχει οριστεί σαφώς η λειτουργικότητα της κάθε απαίτησης και τα χαρακτηριστικά που θα βοηθήσουν την ομάδα ανάπτυξης να την υλοποιήσει με ορθό τρόπο. Αν για κάποιο λόγο μία απαίτηση είναι αβέβαιη θα πρέπει να οριστεί ως TBD (To Be Determined - Να Καθοριστεί), έτσι ώστε να μην προχωρήσει η υλοποίησή της.

**Ορθές.** Θα πρέπει να βεβαιώνεται ότι οι απαιτήσεις δεν έρχονται σε σύγκρουση με τους περιορισμούς και τους επιχειρηματικούς κανόνες. Για τον λόγο αυτό είναι σημαντικό να συμμετέχουν οι χρήστες στην ανάλυσή τους, αφού είναι οι μοναδικοί που γνωρίζουν τον τρόπο λειτουργίας της επιχείρησης.

**Εφικτές.** Πρέπει να είναι δυνατό να εφαρμοστεί κάθε απαίτηση εντός των γνωστών δυνατοτήτων και περιορισμών του συστήματος και του λειτουργικού περιβάλλοντος. Για να αποφευχθεί ο καθορισμός ανέφικτων απαιτήσεων θα πρέπει να εμπλακεί ένας αναλυτής απαιτήσεων κατά την διαδικασία εκμαίευσης.

**Απαραίτητες**. Η κάθε απαίτηση θα πρέπει πρώτα να έχει πάρει την έγκριση των πελατών και μετά να υλοποιείται. Η δημιουργία περιττής λειτουργικότητας θα δημιουργήσει αύξηση του χρονοδιαγράμματος παράδοσης του έργου και άσκοπη σπατάλη πολύτιμων πόρων.

**Να δηλώνονται προτεραιότητες.** Εάν υπάρχει μεγάλος αριθμός απαιτήσεων, θα πρέπει να ορίζονται προτεραιότητες έτσι ώστε να υλοποιούνται πρώτα οι πιο αναγκαίες και μετά οι υπόλοιπες. Με τον τρόπο αυτό χρονοπρογραμματίζεται καλύτερα το έργο και η υλοποίηση γίνεται πιο ομαλά.

**Σαφής.** Οι απαιτήσεις θα πρέπει να είναι γραμμένες με σαφήνεια και με τρόπο κατανοητό από τους χρήστες. Θα πρέπει να προτιμάται η φυσική γλώσσα και τηρούνται τα διάφορα πρότυπα τεκμηρίωσης, έτσι ώστε να μην υπάρχουν προβλήματα ανάγνωσης και κατανόησης.

**Επαληθεύσιμες.** Εάν οι απαιτήσεις δεν είναι επαληθεύσιμες τότε δεν μπορούν να γίνουν δοκιμές ελέγχου, με αποτέλεσμα να μην μπορούμε να εξακριβώσουμε εάν έχουν υλοποιηθεί σωστά.

**Τροποποιήσιμες**. Είναι ένας σημαντικός παράγοντας στην περίπτωση που χρειαστεί για οποιοδήποτε λόγο να γίνει κάποια αλλαγή.

**Ανιχνεύσιμες.** Μία ανιχνεύσιμη απαίτηση μπορεί να συνδεθεί με την καταγωγή της και να συνδεθεί στα στοιχεία σχεδιασμού και πηγαίου κώδικα που την εφαρμόζουν. Επίσης θα πρέπει να συνδεθεί με τις δοκιμές ελέγχου που την αφορούν. Έτσι κάθε φορά που θα χρειαστεί να γίνει μία αλλαγή, η διαδικασία εκτίμησης των επιπτώσεων θα γίνεται πιο εύκολα αφού θα ξέρουμε ακριβώς τι κομμάτια θα πρέπει να αλλάξουν [2,11].

## 2.9 Αναγκαιότητα Συνεργασίας Αναλυτών και Πελατών

Οι πελάτες που ζητούν ένα πληροφοριακό σύστημα, συχνά δεν καταλαβαίνουν πόσο σημαντικό είναι να συμμετέχουν σε αυτό δίνοντας πολύτιμες πληροφορίες. Η επιτυχία όμως του προϊόντος βασίζεται στην συνεργασία πελατών και προγραμματιστών. Υπάρχουν διαφορετικοί τύποι απαιτήσεων. Οι επιχειρηματικές, οι απαιτήσεις των χρηστών και οι λειτουργικές. Ως εκ τούτου οι προγραμματιστές δεν μπορούν να περιγράψουν τις απαιτήσεις των χρηστών, αφού δεν είναι οι ίδιοι χρήστες. Οι χρήστες από την άλλη μπορούν απλά να περιγράψουν τις διεργασίες που επιθυμούν το νέο σύστημα να κάνει, αλλά δεν μπορούν αν περιγράψουν τις λειτουργικές απαιτήσεις που θα κάνουν υλοποιήσιμες τις παραπάνω διεργασίες.

Καταλαβαίνουμε πόσο σημαντική είναι η συνεργασία αυτών των δύο ομάδων για την επιτυχία ενός Πληροφοριακού Συστήματος. Αν κάτι τέτοιο δεν γίνει οι απαιτήσεις των χρηστών δεν θα ικανοποιηθούν, θα χρειαστεί να γίνουν αλλαγές και έτσι η εταιρεία θα χάσει πολύτιμους πόρους τόσο σε χρόνο όσο και σε χρήμα [2,10,18,20].

***Ποιοι είναι οι πελάτες;***

Πελάτης, γενικότερα, είναι ένα άτομο ή ένας οργανισμός που επωφελείται έμμεσα ή άμεσα από ένα προϊόν. Οι ενδιαφερόμενοι ενός προϊόντος λογισμικού είναι εκείνοι που ζητούν , πληρώνουν, επιλέγουν, χρησιμοποιούν και δέχονται τα αποτελέσματα που παράγει το σύστημα. Στην κατηγορία των ενδιαφερόμενων επίσης ανήκουν οι αναλυτές των απαιτήσεων, η ομάδα ανάπτυξης, η ομάδα δημιουργίας δοκιμών ελέγχου, η ομάδα τεκμηρίωσης, οι Διαχειριστές του έργου, η ομάδα υποστήριξης, η νομική ομάδα και η ομάδα Marketing [9].

Ο Senior Manager μίας εταιρείας αντιπροσωπεύει το είδος του πελάτη που πληρώνει για ένα έργο λογισμικού. Είναι υπεύθυνος να ορίσει τις επιχειρηματικές απαιτήσεις και να προσδιορίσει τον σκοπό του προϊόντος και την δράση της εταιρείας με αυτό. Οι επιχειρηματικές απαιτήσεις περιγράφουν τους επιχειρηματικούς στόχους που οι πελάτες, η εταιρεία και οι υπόλοιποι ενδιαφερόμενοι θέλουν να επιτύχουν. Δημιουργούν με άλλα λόγια ένα πλαίσιο στο οποίο θα πρέπει να κινηθεί το έργο. Όλα τα υπόλοιπα χαρακτηριστικά του έργου θα πρέπει να συμβαδίζουν και να ικανοποιούν τις επιχειρηματικές απαιτήσεις, που από μόνες τους όμως δεν δίνουν λεπτομέρειεςγια το τι θα πρέπει να αναπτυχθεί.

Το επόμενο στάδιο των απαιτήσεων , οι απαιτήσεις των χρηστών, προέρχονται από τα άτομα που στην ουσία θα χρησιμοποιήσουν το προϊόν είτε έμμεσα είτε άμεσα. Αυτοί οι χρήστες, που αποκαλούνται και σαν End Users – Τελικοί χρήστες, αποτελούν ένα άλλο είδος πελάτη.

Μπορούν να περιγράψουν τις διεργασίες που θα πρέπει να κάνει το σύστημα, όπως επίσης και τα ποιοτικά χαρακτηριστικά που περιμένουν από αυτό.

Οι διάφοροι ενδιαφερόμενοι θα πρέπει να συμβιβαστούν έτσι ώστε οι επιχειρηματικές απαιτήσεις να συμφωνούν με τις απαιτήσεις των χρηστών και το αντίστροφο. Θα πρέπει επίσης να εργάζονται μαζί με τους αναλυτές κατά την διαδικασία της ανάπτυξης των απαιτήσεων και όχι απλά να περιμένουν από εκείνους να βρουν τις ανάγκες τους.

Η κατάσταση είναι κάπως διαφορετική στην ανάπτυξη εμπορικού λογισμικού, όπου ο πελάτης και ο χρήστης είναι το ίδιο άτομο. Στην περίπτωση αυτή το τμήμα Marketing ή ο ProductManager αναλαμβάνει να καθορίσει τι θα ήταν δελεαστικό για τον πελάτη. Αλλά ακόμα και σε εδώ θαείναι καλό να εμπλέκονται και οι πελάτες, ένα αντιπροσωπευτικό τους δείγμα, στην διαδικασία της ανάπτυξης των απαιτήσεων.

Καμιά φορά ανάμεσα στις απαιτήσεις των χρηστών και στις επιχειρηματικές δημιουργούνται διάφορες συγκρούσεις. Αυτό συμβαίνει γιατί οι επιχειρηματικές απαιτήσεις αντανακλούν κάποιες επιχειρηματικές στρατηγικές ή οικονομικούς περιορισμούς που συνήθως δεν γίνονται αντιληπτοί από τους χρήστες. Θα πρέπει λοιπόν να ξεκαθαριστούν και να οριστούν όλοι περιορισμοί για να μειωθούν οι παραπάνω συγχύσεις. Με τον τρόπο αυτό μπορεί μεν να μην μείνουν όλοι ικανοποιημένοι, αλλά τουλάχιστον θα ξέρουν τον λόγο που οδήγησε να παρθούν οι όποιες αποφάσεις [21].

### 2.9.1 Η σχέση πελατών και αναλυτών

Οι υψηλού επιπέδου απαιτήσεις είναι αποτέλεσμα της άριστης συνεργασίας και επικοινωνίας μεταξύ των πελατών και των αναλυτών. Μία συλλογική προσπάθεια μπορεί να επιτύχει όταν όλα τα εμπλεκόμενα μέρη γνωρίζουν τι ακριβώς μπορεί και πρέπει να επιτευχθεί. Παρακάτω υπάρχουν δέκα κανόνες δικαιωμάτων και υποχρεώσεων που έχουν και οι δύο πλευρές κατά τη διάρκεια του έργου [22,23].

**Δικαιώματα πελατών (Υποχρεώσεις αναλυτών)**

- Ο αναλυτής θα πρέπει να "μιλάει την γλώσσα των πελατών" , να γίνεται δηλαδή κατανοητός. Οι συζητήσεις σχετικά με τις απαιτήσεις θα πρέπει να γίνεται σύμφωνα με τους όρους και τις ανάγκες τις επιχείρησης. Θα πρέπει να δημιουργηθεί ένα λεξικό της ορολογίας που θα ακολουθηθεί, έτσι ώστε να μην χάνεται πολύτιμος χρόνος στην "μετάφραση".
- Ο αναλυτής θα πρέπει να μάθει τους επιχειρηματικούς στόχους της επιχείρησης και του συστήματος. Αυτό θα βοηθήσει τους αναλυτές να φτιάξουν ένα σύστημα που θα ικανοποιεί τις ανάγκες των χρηστών. Σε διαφορετική περίπτωση θα αναγκαστεί να επαναπροσδιοριστεί η σχεδίαση του συστήματος.
- Ο αναλυτής θα πρέπει να δομήσει τις πληροφορίες που του παρέχονται κατά τη διάρκεια της εκμαίευσης σε ένα έγγραφο τεκμηρίωσης. Τέτοιες πληροφορίες είναι οι περιπτώσεις

χρήσης, οι επιχειρηματικοί κανόνες, οι λειτουργικές απαιτήσεις και οι ποιοτικοί στόχοι. Το παραδοτέο αυτής της ανάλυσης είναι το Έγγραφο Προδιαγραφής Απαιτήσεων, που στην ουσία είναι μία συμφωνία μεταξύ των πελατών και των αναλυτών αναφορικά με τις λειτουργίες, τα χαρακτηριστικά και την ποιότητα του έργου που πρόκειται να παραδοθεί. Το Έγγραφο Προδιαγραφής Απαιτήσεων θα πρέπει να γράφεται με απλό και κατανοητό τρόπο για να διασφαλιστεί η πλήρης κάλυψη των αναγκών των πελατών.

- Ο αναλυτής θα πρέπει να εξηγεί όλα τα προϊόντα που δημιουργούνται από την διαδικασία των απαιτήσεων. Θα πρέπει να παρουσιάζει τις απαιτήσεις χρησιμοποιώντας διάφορα διαγράμματα, ως συμπληρωματικά τουΕγγράφου Προδιαγραφής Απαιτήσεων. Τα διαγράμματα αυτά θα βοηθήσουν στην καλύτερη κατανόηση και έκφραση των στοιχείων του συστήματος. Είναι ναι μεν άγνωστα προς τους πελάτες, αλλά είναι εύκολο κατανοητά. Θα πρέπει να εξηγηθεί ο σκοπός του κάθε διαγράμματος και να εξηγηθεί πώς μέσα από αυτά μπορούν οι πελάτες να αξιολογούν το έργο.
- Θα πρέπει οι αναλυτές να συμπεριφέρονται με σεβασμό και επαγγελματισμό κατά την διάρκεια του έργου. Οι πελάτες που συμμετέχουν στην διαδικασία της ανάλυσης θα πρέπει να αντιμετωπίζονται με σεβασμό και να αναγνωρίζεται ο χρόνος που αφιερώνουν. Θα πρέπει όμως να δίνεται και ο αντίστοιχος σεβασμός προς τους αναλυτές, αφού στην ουσία προσπαθούν να δημιουργήσουν ένα σύστημα προς όφελος των πελατών.
- Θα πρέπει οι αναλυτές να προτείνουν εναλλακτικές ιδέες και προτάσεις όσον αφορά τις απαιτήσεις και την εφαρμογή του συστήματος. Προτείνοντας διαφορετικές δυνατότητες στο έργο που υλοποιείται, αυτόματα δίνεται μεγαλύτερη αξία στο παραδοτέο προϊόν.
- Οι αναλυτές θα πρέπει να περιγράφουν τα χαρακτηριστικά του προϊόντος που το κάνουν πιο ευκολότερο και πιο ευχάριστο στην χρήση. Τα χαρακτηριστικά αυτά κάνουν το έργο πιο ευχάριστο και έτσι οι πελάτες το αποδέχονται ευκολότερα.
- Οι αναλυτές θα πρέπει να καθορίζουν με σαφήνεια τις απαιτήσεις. Όσος περισσότερος χρόνος επενδύεται στην τεκμηρίωση των απαιτήσεων, τόσο πιο εύκολα γίνεται η σχεδίαση και η υλοποίηση.
- Θα πρέπει να παρέχουν σωστές εκτιμήσεις όσον αφορά το κόστος και τις επιπτώσεις των αλλαγών των απαιτήσεων. Η εκτίμηση της επίδρασης μίας προτεινόμενης αλλαγής είναι απαραίτητη για να ληφθεί η απόφαση έγκρισης ή απόρριψής της. Οι εκτιμήσεις που παρέχονται θα πρέπει να είναι καλά μελετημένες και ρεαλιστικές.
- Οι αναλυτές θα πρέπει να παραδίδουν το έργο με την συμφωνημένη λειτουργικότητα και ποιότητα. Για να γίνει αυτό θα πρέπει οι πελάτες να δίνουν όλες οι απαραίτητες πληροφορίες στους αναλυτές, αλλιώς το σύστημα δεν θα μπορέσει να ικανοποιήσει τις ανάγκες τους.

**Υποχρεώσεις των πελατών (Δικαιώματα αναλυτών)**

- Πρέπει να ενημερώνουν τους αναλυτές σχετικά με την επιχείρηση να και να ορίσουν την επιχειρησιακή ορολογία. Σκοπός δεν είναι να γίνουν οι αναλυτές "ειδικοί" στην

επιχειρησιακή ορολογία, αλλά να κατανοήσουν τα προβλήματα που τους ζητείται να λύσουν.

- Πρέπει να αφιερώνουν όσο χρόνο χρειαστεί για να αναδειχθούν και να αποσαφηνιστούν οι απαιτήσεις. Συνήθως οι πελάτες δεν αφιερώνουν χρόνο στην διαδικασία της ανάλυσης γιατί είναι μία χρονοβόρα διαδικασία. Τα επιθυμητά όμως αποτελέσματα θα έρθουν μόνο εάν επενδυθεί χρόνος και από τις δύο μεριές σε αυτή τη διαδικασία. Οι πελάτες θα πρέπει να απαντάνε ακόμα και στις πιο απλές ερωτήσεις.
- Πρέπει να είναι συγκεκριμένοι σχετικά με τα στοιχεία εισόδου του συστήματος. Δεν θα πρέπει να μένουν οι απαιτήσεις αβέβαιες και ελλιπείς. Εάν για κάποιο λόγο κάποια απαίτηση δεν αποσαφηνιστεί πλήρως θα πρέπει να οριστεί ως TBD (To Be Determined), για να οριστεί στην πορεία.
- Να παίρνουν όποιες αποφάσεις τους ζητηθούν και στον καθορισμένο χρόνο. Ο αναλυτής θα ζητήσει από τους πελάτες να κάνουν κάποιες αποφάσεις και να κάνουν τις αντίστοιχες αλλαγές. Από την μεριά τους οι εξουσιοδοτούμενοι πελάτες θα πρέπει να ενεργήσουν στα χρονικά πλαίσια που τους έχουν δοθεί. Οι αναλυτές από την μεριά τους δεν μπορούν να συνεχίσουν με ακρίβεια εάν δεν έχουν μία ξεκάθαρη απάντηση και όσο εκείνη αργεί, τόσο αργεί και η ανάπτυξη του έργου.
- Να σέβονται τις εκτιμήσεις κόστους και εφικτότητας που κάνουν οι αναλυτές. Όλες οι λειτουργίες του λογισμικού έχουν κόστος και οι αναλυτές είναι οι υπεύθυνοι να κάνουντην σωστή εκτίμηση. Κάποια από τα στοιχεία που οι πελάτες θέλουν να συμπεριληφθούν στο σύστημα μπορεί να μην είναι τεχνικά εφικτά ή να είναι αρκετά δαπανηρά ως προς τον προϋπολογισμό του έργου. Επίσης κάποιες απαιτήσεις μπορεί να δημιουργούν αστάθεια ή να απαιτούν πρόσβαση σε δεδομένα που δεν είναι συμβατά με το σύστημα. Γίνεται λοιπόν κατανοητό ότι οι εκτιμήσεις μπορεί να μην είναι πάντα αρεστές στους πελάτες, αλλά θα πρέπει να γίνονται κατανοητές.
- Να ορίζουν προτεραιότητες στις απαιτήσεις και στα άλλα στοιχεία του συστήματος. Λίγα έργα έχουν τους κατάλληλους πόρους να συμπεριλάβουν κάθε επιθυμητή λειτουργικότητα. Για τον λόγο αυτό θα πρέπει να δίνονται προτεραιότητες στις απαιτήσεις και εκείνες που δεν είναι αναγκαίες να μην συμπεριλαμβάνονται καθόλου. Ο ρόλος των πελατών στον προσδιορισμό των προτεραιοτήτων είναι σημαντικός, αφού οι αναλυτές δεν γνωρίζουν την σημαντικότητα της κάθε απαίτησης στο σύστημα. Οι αναλυτές, από τη μεριά τους, θα πρέπει να προσδιορίζουν το κόστος και το ρίσκο της κάθε απαίτησης για να βοηθούν τους πελάτες να καθορίσουν τις προτεραιότητες. Έτσι το έργο θα έχει την μεγαλύτερη εφικτή αξία, στο χαμηλότερε κόστος και στον λιγότερο χρόνο.
- Να διαβάζουν τις αναφορές και να αξιολογούν τα πρωτότυπα. Η συμμετοχή των πελατών στην αξιολόγηση των αναφορών είναι ένας σημαντικός παράγοντας ελέγχου των εργασιών του έργου. Αν οι αντιπρόσωποι των πελατών δεν πειστούν από την τεκμηρίωση θα πρέπει να ενημερώσουν τους υπεύθυνους για να δρομολογηθούν οι

απαραίτητες αλλαγές. Το ίδιο ακριβώς θα πρέπει να γίνεται και στα πρωτότυπα (τα πιλοτικά κομμάτια λογισμικού).

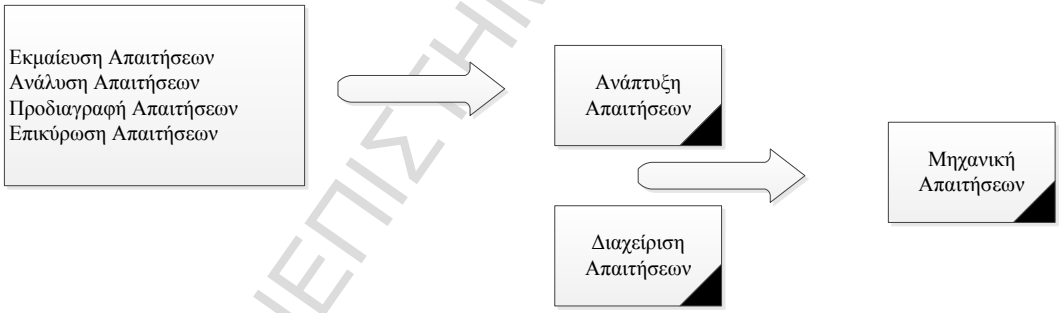
- Να ενημερώνουν γρήγορα σχετικά με τις αλλαγές που επιθυμούν να γίνουν. Η συχνή αλλαγή των απαιτήσεων ελλοχεύει το κίνδυνο να μην παραδοθεί τού έργο στην συμφωνημένη ποιότητα. Οι αλλαγές, κατά την διάρκεια της ανάπτυξης, είναι αναπόφευκτες αλλά όταν γίνονται αργά στον κύκλο ζωής κοστίζουν πολύ περισσότερο. Όσο πιο γρήγορα, λοιπόν, γίνουν τόσο το καλύτερο για την πορεία του έργου.
- Να ακολουθούν την διαδικασία που έχει οριστεί σχετικά με μία αίτηση αλλαγής απαίτησης. Για να μειωθεί η επίδραση των αλλαγών θα πρέπει να ακολουθείται η καθορισμένη ,από τους συμμετέχοντες, διαδικασία αίτησης αλλαγής απαιτήσεων. Η διαδικασία αυτή εξασφαλίζει ότι τα αιτήματα αλλαγών δεν χάνονται, πως αναλύονται οι επιδράσεις των προτεινόμενων αλλαγών και γίνεται σωστή εκτίμηση. Έτσι οι ενδιαφερόμενοι θα μπορούν πάρουν τις κατάλληλες αποφάσεις έγκρισης ή απόρριψης των αλλαγών.
- Να σέβονται την διαδικασία που ακολουθούν οι αναλυτές σχετικά με την διαδικασία της Μηχανικής των Απαιτήσεων. Υπάρχει μία λογική πίσω από την προσέγγιση που χρησιμοποιούν οι αναλυτές και για τον λόγο αυτό θα πρέπει οι πελάτες να σέβονται και να ακολουθούν τις τεχνικές των αναλυτών.

Μόνο εάν γίνουν όλα τα παραπάνω θα είναι όλοι πεπεισμένοι ότι το έργο δεν θα παρεκκλίνει από το πεδίο ορισμού και ότι θα ικανοποιεί όλες τις αναγκαίες για αυτό απαιτήσεις. Η ανάλυση των απαιτήσεων είναι μία χρονοβόρα και πολλές φορές επαναληπτική διαδικασία, αλλά θα γίνει πολύ πιο απλή εάν όλοι οι συμμετέχοντες ακολουθήσουν τους κανόνες.

# 3 Ανάπτυξη Απαιτήσεων

Παραπάνω είδαμε τα προβλήματα που συνήθως δημιουργούνται κατά την διαδικασία της Μηχανικής των Απαιτήσεων, ορίστηκαν οι απαιτήσεις του λογισμικού και έγινε ο διαχωρισμός τους στα διάφορα επίπεδα. Στην συνέχεια επισημάνθηκαν τα πλεονεκτήματα μίας σωστής διαδικασίας ανάλυσης απαιτήσεων και η σημαντικότητα της αγαστής συνεργασίας των πελατών και των αναλυτών για την επιτυχή ολοκλήρωση της διαδικασίας. Η Μηχανική των Απαιτήσεων είναι ο συνδυασμός της Ανάπτυξης και της Διαχείρισης των απαιτήσεων. Η ανάπτυξη των απαιτήσεων, η οποία αναλύεται στο παρακάτω κεφάλαιο, χωρίζεται σε τέσσερα επιμέρους βήματα:

- Εκμαίευση των Απαιτήσεων.
- Ανάλυση των Απαιτήσεων.
- Προδιαγραφή των Απαιτήσεων.
- Επικύρωση των Απαιτήσεων.




**Σχήμα 20**. Διαγραμματικά τα βήματα της Μηχανικής των Απαιτήσεων



## 3.1 Εκμαίευση Απαιτήσεων

Η κατανόηση των απαιτήσεων του χρήστη αποτελεί ένα αναπόσπαστο μέρος των πληροφοριακών συστημάτων και είναι πολύ χρήσιμο για την επιτυχή ανάπτυξη του έργου. Ωστόσο, η τεκμηρίωση των απαιτήσεων δεν είναι και τόσο εύκολη. Παρακάτω περιγράφονται κάποιες γενικές μέθοδοι που μπορούν αν υποστηρίξουν την ανάλυση των απαιτήσεων των χρηστών, οι οποίες μπορούν να υιοθετηθούν και εφαρμοστούν σε διάφορες καταστάσεις. περιγράφονται κάποιες μελέτες περίπτωσης που μπορούν και πρακτικά να εφαρμοστούν [9].

### 3.1.1 Όραμα και σκοπός συστήματος

Οι επιχειρηματικές απαιτήσεις αντιπροσωπεύουν το ανώτατο επίπεδο της αλυσίδας των απαιτήσεων. Ορίζουν τον σκοπό και το πεδίο εφαρμογής του νέου συστήματος λογισμικού. Οι απαιτήσεις των χρηστών και οι λειτουργικές απαιτήσεις θα πρέπει να ευθυγραμμίζονται με το πλαίσιο και τους στόχους που θέτουν οι επιχειρηματικές απαιτήσεις. Όσες απαιτήσεις δεν συμμορφώνονται με αυτήν την ιδέα δεν θα πρέπει να συμπεριλαμβάνονται στο έργο.

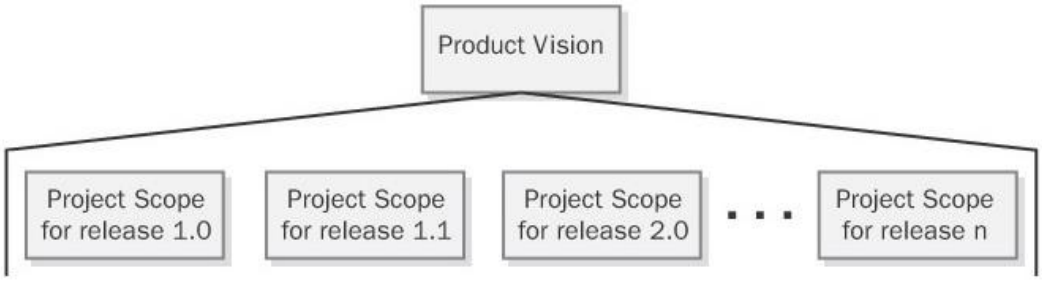
Ένα έργο που δεν έχει μία σαφή και καθορισμένη κατεύθυνση μπορεί να οδηγηθεί στην αποτυχία. Απαραίτητη προϋπόθεση για να καταλάβουν οι ενδιαφερόμενοι τις απαιτήσεις είναι πρώτα να έχουν κατανοήσει τους επιχειρηματικούς στόχους του προϊόντος. Στα αναπτυξιακά έργα που υπάρχει γεωγραφικός διαχωρισμός των συμμετεχόντων είναι ιδιαίτερα κρίσιμο να έχουν οριστεί με σαφήνεια τόσο ο σκοπός όσο και το όραμα του έργου.

Ένα σημάδι ότι οι επιχειρηματικές απαιτήσεις δεν έχουν οριστεί σαφώς είναι όταν ορισμένα στοιχεία του συστήματος που στην αρχή είχαν περιληφθεί αργότερα διαγράφονται και ορισμένα που δεν είχαν οριστεί τελικά συμπεριλαμβάνονται. Το όραμα και ο σκοπός θα πρέπει να ξεκαθαρίζονται πριν οριστούν με λεπτομέρεια οι λειτουργικές απαιτήσεις.Ο ορισμός του πεδίου εφαρμογής και των περιορισμών του έργου θα μας βοηθήσει στην επιλογή των προτεινόμενων στοιχείων που θα περιληφθούν τελικά σε αυτό. Βοηθάει επίσης στην λήψη αποφάσεων για την αποδοχή ή την απόρριψη των προτεινόμενων αλλαγών ή βελτιώσεων των απαιτήσεων [24].

***Ορισμός Οράματος μέσω των Επιχειρηματικών Απαιτήσεων***

Το όραμα του προϊόντος ευθυγραμμίζει όλα τα εμπλεκόμενα μέρη σε μία κοινή κατεύθυνση. Περιγράφει το ρόλο του προϊόντος και το γιατί αυτό υπάρχει. Το Project Scope προσδιορίζει πώς το έργο θα εκπληρώσει τον σκοπό του. Διαχωρίζει τα όρια του τι θα υπάρχουν και τι όχι μέσα σε αυτό, θέτει δηλαδή τους περιορισμούς που θα ισχύουν.

Το όραμα εφαρμόζεται και ισχύει για το όλο έργο. Μπορεί με την πάροδο του χρόνου να αλλάξει, εάν αλλάξει η στρατηγική τοποθέτηση ή οι επιχειρηματικοί στόχοι του προϊόντος. Το πεδίο εφαρμογής του έργου μπορεί να αλλάξει αν αυξηθεί ή μειωθεί η λειτουργικότητα του προϊόντος όπως φαίνεται στο Σχήμα 14. Το πεδίο εφαρμογής είναι πιο δυναμικό από το όραμα επειδή ο διαχειριστής του έργου καθορίζει τα περιεχόμενα της κάθε έκδοσης σύμφωνα με το χρονοδιάγραμμα, τους διαθέσιμους οικονομικούς πόρους, τις πηγές και τους περιορισμούς της ποιότητας. Στόχος των αναλυτών κα των σχεδιαστών είναι να αντιμετωπίσουν το πεδίο εφαρμογής σαν ένα υποσύνολο του οράματος του έργου. Το πεδίο του κάθε έργουεμφανίζεται στο SRS και όχι σε ένα ξεχωριστό έγγραφο οράματος και πεδίου εφαρμογής. Τα μεγάλα έργα ωστόσο θα πρέπει να έχουν πλήρη εικόνα του οράματος, του πεδίου ορισμού και του Εγγράφου Προδιαγραφής Απαιτήσεων.




**Σχήμα 21.** Το όραμα του προϊόντος περιλαμβάνει το πεδίο εφαρμογής για κάθε προγραμματισμένη έκδοση

### 3.1.2 Αντικρουόμενες επιχειρηματικές απαιτήσεις

Οι επιχειρηματικές απαιτήσεις που συλλέγονται από τις διάφορες πηγές μπορεί να είναι αντίθετες μεταξύ τους. Υπάρχουν περιπτώσεις που οι απαιτήσεις των χρηματοδοτών του έργου έρχονται σε αντίθεση με τις απαιτήσεις των πελατών (όταν μιλάμε για ένα εμπορικό σύστημα λογισμικού). Οι διαφορετικές ανάγκες των παραπάνω μπορεί να οδηγήσουν σε διαφορετικούς στόχους, σε διαφορετικούς περιορισμούς και διαφορετικό κόστος υλοποίησης πράγμα το οποίο θα κάνει μη συνεπείς τις επιχειρηματικές απαιτήσεις. Οι συγκρούσεις αυτές θα πρέπει να επιλυθούν αρχίσει η διαδικασία της ανάλυσης των απαιτήσεων. Θα πρέπει να δοθεί έμφαση στις απαιτήσεις εκείνες που θα εκπληρώνουν τους θεμελιώδεις στόχους του προϊόντος και προσδίδουν την μέγιστη επιχειρηματική αξία στο προϊόν.

Όσο περισσότεροι είναι οι ενδιαφερόμενοι, τόσο περισσότερα είναι τα συμφέροντα που εκείνοι εξυπηρετούν και τόσο περισσότερο αυξάνει το Scopecreep του έργου. Ένα ανεξέλεγκτο ScopeCreep στο που οι ενδιαφερόμενοι προσπαθούν να γεμίσουν το έργο με απαιτήσεις και χαρακτηριστικά που ικανοποιούν τις ανάγκες τους μπορεί να οδηγήσει σε αποτυχία παράδοσης του έργο τόσο στον χρόνο που έχει καθοριστεί όσο και στην ποιότητα που έχει συμφωνηθεί [14,25].

***Επιχειρηματικές Απαιτήσεις και Περιπτώσεις Χρήσης***

Οι επιχειρηματικές απαιτήσεις καθορίζουν τόσο το σύνολο των περιπτώσεων χρήσης (Use Cases) όσο και το βάθος ή το επίπεδο που υλοποιείται το κάθε use case. Αν μέσα από τις επιχειρηματικές απαιτήσεις εντοπίσουμε κάποια περίπτωση χρήσης είναι μεν σημαντικό αλλά είναι εκτός του πεδίου εφαρμογής του έργου, μπορούμε να αυξήσουμε λίγο το εύρος του έτσι ώστε να το συμπεριλάβουμε στην υλοποίηση. Οι επιχειρηματικές απαιτήσεις αναδεικνύουν ποια usecase απαιτούν μία ισχυρή και ολοκληρωμένη λειτουργική εφαρμογή και ποιά απαιτούν μία πιο επιφανειακή εφαρμογή, τουλάχιστον σε ένα πρώιμο επίπεδο.

Οι επιχειρηματικές απαιτήσεις επηρεάζουν τον τρόπο που εφαρμόζονται γενικά οι απαιτήσεις. Ορίζουν προτεραιότητες στις περιπτώσεις χρήσης και τις σχετιζόμενες με αυτές απαιτήσεις. Αν για παράδειγμα σε ένα e-shop θέλουμε σαν πρωταρχικό στόχο να αυξήσουμε τα έσοδα, τότε θα δώσουμε προτεραιότητα στην εφαρμογή των στοιχείων και των χαρακτηριστικών που

επιτρέπουν την πώληση περισσότερων αγαθών στον πελάτη. Κατά συνέπεια τα χαρακτηριστικά που αφορούν την εμφάνιση θα μπουν σε δεύτερη μοίρα.

***Έγγραφο Οράματος και Πεδίου Εφαρμογής του Έργου***

Το έγγραφο του οράματος και του πεδίου εφαρμογής συλλέγει όλες τις επιχειρηματικές απαιτήσεις και θέτει τις βάσεις για την ανάπτυξη του έργου. Για τον ίδιο σκοπό κάποιοι οργανισμοί δημιουργούν ένα Πλάνο του Έργου (Project Charter) ή ένα Κείμενο Επιχειρησιακών Απαιτήσεων(Business Case Document). Κάποιοι άλλοι που αναπτύσσουν κυρίως εμπορικές εφαρμογές δημιουργούν το Market Requirements Document (MRD). Το MRD περιέχει περισσότερες λεπτομέρειες από ότι τα έγγραφα οράματος και πεδίου ορισμού όσον αφορά τα τμήματα της αγοράς που απευθύνεται το προϊόν.

Η δημιουργία αυτού του εγγράφου οράματος και πεδίου ορισμού μπορεί να γίνει από τον αναλυτή του συστήματος σε συνεργασία με έναν ή μία ομάδα χρηματοδοτών του έργου. Οι επιχειρηματικές απαιτήσεις θα πρέπει να δίνονται από άτομα που έχουν μία σαφή αίσθηση του γιατί εκτελείται το έργο. Τα άτομα αυτά μπορεί να είναι οι πελάτες, ανώτερα διοικητικά στελέχη, ο Διαχειριστής του Έργου, ένας εμπειρογνώμονας ή το τμήμα Marketing του έργου.

1. **Business Requirements**
   - 1.1 Background
   - 1.2 Business Opportunity
   - 1.3 Business Objectives and Success Criteria
   - 1.4 Customer or Market Needs
   - 1.5 Business Risks
2. **Vision of the Solution**
   - 2.1 Vision Statement
   - 2.2 Major Features
   - 2.3 Assumptions and Dependencies
3. **Scope and Limitations**
   - 3.1 Scope of Initial Release
   - 3.2 Scope of Subsequent Releases
   - 3.3 Limitations and Exclusions
4. **Business Context**
   - 4.1 Stakeholder Profiles
   - 4.2 Project Priorities
   - 4.3 Operating Environment

**Σχήμα 22.** Πρότυπο έγγραφο οράματος και πεδίου εφαρμογής του έργου

- **Business Requirements**

Όλα τα έργα ξεκινούν με την πεποίθηση ότι θα βελτιώσουν κατά κάποιο τρόπο μία συγκεκριμένη διαδικασία. Οι επιχειρηματικές απαιτήσεις περιγράφουν τα οφέλη που θα παρέχει το νέο σύστημα στους χορηγούς, τους αγοραστές και τους χρήστες.

- ✓ **Background.** Μία συνοχή του σκεπτικού της δημιουργίας του νέου προϊόντος. Μία γενική περιγραφή της κατάστασης που οδήγησε στην απόφαση της δημιουργίας του συστήματος.
- ✓ **Business Opportunity.**Για ένα εμπορικό προϊόν περιγράφεται η δυνατότητα που υπάρχει στην αγορά και η αγορά στην οποία το προϊόν θα ανταγωνιστεί. Για ένα εταιρικό πληροφοριακό σύστημα περιγράφονται τα επιχειρηματικά προβλήματα που λύνονται, οι επιχειρηματικές διεργασίες που βελτιώνονται και το περιβάλλον στο οποίο θα χρησιμοποιείται. Παρέχεται μία συγκριτική αξιολόγηση του υφιστάμενου προϊόντος με το νέο, αλλά και τα πλεονεκτήματα που αυτό θα επιφέρει στην εταιρεία. Περιγράφονται τα προβλήματα που ήδη υπάρχον δεν μπορεί να λύσει. Φαίνεται το πώς το νέο σύστημα ευθυγραμμίζεται με τις ανάγκες της αγοράς, την εξέλιξη της τεχνολογίας και την στρατηγική κατεύθυνση της εταιρείας. Συμπεριλαμβάνει μία σύντομη περιγραφή κάποιων άλλων τεχνολογιών, διαδιακασίων ή πόρων που προσέφεραν ακόμη πιο ολοκληρωμένη λύση στον πελάτη.
- ✓ **Business Objectives and Success Criteria.**Συνοψίζει τα επιχειρηματικά οφέλη που προσφέρει το νέο προϊόν με ποσοτικό και μετρήσιμο τρόπο. Ο Πίνακας 2 απεικονίζει κάποια παραδείγματα τόσο οικονομικών, όσο και μη οικονομικών στόχων. Αναφέρονται οι παράγοντες που έχουν μεγάλο αντίκτυπο στην επιτυχία του έργου και καθορίζονται μετρήσιμα κριτήρια για τον αν έχουν επιτευχθεί οι επιχειρηματικοί στόχοι.

| **Οικονομικές** | **Μη Οικονομικές** |
|---|---|
| • Καταγραφή μεριδίου αγοράς<br>• Αύξηση μεριδίου αγοράς<br>• Απόδοση επένδυσης<br>• Επίτευξη θετικών ταμειακών ροών<br>• Μείωση κόστους<br>• Αύξηση περιθωρίου κέρδους | • Ικανοποίηση των πελατών<br>• Ανάπτυξη συμπληρωματικών προϊόντων<br>• Επίτευξη θετικών κριτικών του προϊόντος<br>• Συμμόρφωση με επιχειρηματικές και κρατικές ρυθμίσεις<br>• Μείωση χρόνων παράδοσης προϊόντων σε πελάτες |

**Πίνακας 2.** Παραδείγματα οικονομικών και μη οικονομικών επιχειρηματικών στόχων

- ✓ **Customeror Market Needs.**Περιγράφονται οι ανάγκες των πελατών που τα ήδη υπάρχοντα πληροφοριακά συστήματα δεν μπορούν να ικανοποιήσουν. Παρουσιάζεται πώς το νέο σύστημα θα ικανοποιήσει αυτές τις ανάγκες και παραδείγματα για το πώς θα χειρίζονται οι πελάτες το σύστημα. Περιλαμβάνει απαιτήσεις απόδοσης υψηλού επιπέδου, αλλά δεν αναφέρονται λεπτομερείς σχεδιασμού και υλοποίησης.
- ✓ **Business Risks.** Συνοψίζονται οι επιχειρηματικοί κίνδυνοι που προκύπτουν από την δημιουργία ή μη του συστήματος. Τέτοιοι κίνδυνοι είναι ο ανταγωνισμός της αγοράς, η αποδοχή των χρηστών, η σωστή χρονική στιγμή παράδοσης του έργου, οι δυσκολίες υλοποίησης και κάθε άλλη πιθανή αρνητική επίπτωση που θα επιφέρει το έργο στην επιχείρηση.

- **Vision of the Solution**

Αυτό το κομμάτι του εγγράφου περιλαμβάνει το στρατηγικό όραμα του συστήματος που θα επιτύχει τους επιχειρηματικούς στόχους. Το όραμα αυτό παρέχει το πλαίσιο σύμφωνα με το οποίο θα λαμβάνονται οι αποφάσεις κατά την διάρκεια του έργου. δεν θα πρέπει να περιλαμβάνει λεπτομερείς λειτουργικές απαιτήσεις και πληροφορίες σχεδιασμού του έργου.

- ✓ Vision Statement**.** Είναι μία περιεκτική δήλωση του οράματος που συνοψίζει τον μακροπρόθεσμο σκοπό και την πρόθεση του νέου προϊόντος. Είναι μία ισορροπημένη άποψη που αντανακλά τις ανάγκες των διάφορων ενδιαφερόμενων. Πρέπει να ανταποκρίνεται στις ανάγκες τις αγοράς, στον υφιστάμενο ανταγωνισμό, στην αρχιτεκτονική στρατηγική κατεύθυνση της επιχείρησης και στους περιορισμούς που αφορούν τους πόρους. Παρακάτω βλέπουμε κάποιες λέξεις "κλειδιά" για τον ορισμό του οράματος του προϊόντος:

**For**. Οι πελάτες που απευθύνεται

**Who**. Η ευκαιρία ή ανάγκη της δημιουργίας του

**The**. Το όνομα του προϊόντος

**Is.** Η κατηγορία του προϊόντος

**That**. Το βασικό όφελος. Γιατί θα το χρησιμοποιήσουν οι πελάτες

**Unlike.** Η εναλλακτική λύση ή το υπάρχον σύστημα ή υπάρχουσα επιχειρηματική διαδικασία

**Ourproduct**. Η διαφοροποίηση και τα πλεονεκτήματα του νέου προϊόντος

- ✓ MajorFeatures. Απαρίθμηση των κύριων χαρακτηριστικών του νέου προϊόντος. Επισημάνει τις δυνατότητες που παρέχει στους χρήστες με τρόπο που να διαφοροποιεί το προϊόν από τα ήδη υπάρχοντα. Δίνοντας μία ξεχωριστή ετικέτα

σε κάθε χαρακτηριστικό, επιτρέπεται ο εντοπισμός των απαιτήσεων των χρηστών, των λειτουργικών απαιτήσεων και κάθε άλλων στοιχείων του συστήματος.

- ✓ Assumptions and Dependencies. Καταγραφή όλων των υποθέσεων που έκαναν οι ενδιαφερόμενοι κατά την σύλληψη του έργου. Οι υποθέσεις αυτές βοηθούν στην ανάπτυξη του έργου στα πλαίσια που έχουν καθοριστεί και αποτρέπουν τυχόν παρεξηγήσεις μεταξύ των αναλυτών και των ενδιαφερόμενων στο μέλλον. Επίσης καταγράφονται οι σημαντικές εξαρτήσεις που έχει το έργο από εξωτερικούς παράγοντες, όπως βιομηχανικά πρότυπα, κυβερνητικούς κανονισμούς, προμηθευτές ή τρίτους συνεργάτες ανάπτυξης.

- **Scope and Limitations**

Το πεδίο εφαρμογής ορίζει την έννοια και το εύρος της προτεινόμενης λύσης. Οι περιορισμοί ταξινομούν εκείνες τις δυνατότητες που το έργο δεν θα περιέχει. Το πεδίο εφαρμογής σε συνδυασμό με τους περιορισμούς συμβάλουν στην δημιουργία πιο ρεαλιστικών προσδοκιών από τους ενδιαφερόμενους. Μερικές φορές οι πελάτες ζητούν κάποια χαρακτηριστικά που είναι αρκετά ακριβά ή εκτός του πεδίου εφαρμογής του έργου. Οι απαιτήσεις αυτές θα πρέπει είτε να απορριφθούν, είτε να συμπεριληφθούν αφού πρώτα διευρυνθεί το πεδίο εφαρμογής του έργου. Θα πρέπει να κρατείται ένα αρχείο των απορριφθέντων απαιτήσεων και του λόγου που οδήγησε στην απόρριψη τους, γιατί μπορεί στην συνέχεια του έργου να αντιπροταθούν.

- ✓ Scope of Initial Release. Συνοψίζει τα κύρια χαρακτηριστικά που θα συμπεριληφθούν στην αρχική έκδοση του έργου. Γίνεται περιγραφή των ποιοτικών χαρακτηριστικών που θα προσφέρει το έργο στις κύριες κατηγορίες χρηστών. Εστιάζει στα χαρακτηριστικά που θα προσφέρουν την μεγαλύτερη αξία στο πιο αποδεκτό κόστος και στο συντομότερο χρονικό διάστημα.
- ✓ Scope of Subsequent Releases. Αν υπάρχει η προσδοκία για μελλοντική εξέλιξη του προϊόντος, θα πρέπει να υποδειχθούν ποια χαρακτηριστικά θα διαφοροποιηθούν στις μεταγενέστερες εκδόσεις. Όσο πιο μακριά κοιτάμε τόσο και πιο ασαφής θα είναι οι δηλώσεις που θα γίνουν και μπορεί εύκολα μην εκτιμήσουμε σωστά την προσδοκώμενη λειτουργικότητα, η οποία μπορεί είτε να μειωθεί είτε να αυξηθεί.
- ✓ Limitations and Exclusions .Ο καθορισμός του ορίου του τι θα συμπεριλαμβάνει και τι όχι το νέο σύστημα βοηθά στο να περιοριστεί το Scope Creep και να οριστούν σαφέστερα οι προσδοκίες των χρηστών. Γίνεται απαρίθμηση των χαρακτηριστικών που επιθυμούσαν οι ενδιαφερόμενοι αλλά δεν συμπεριλήφθηκαν σε κάποια συγκεκριμένη έκδοση.

- **Business Context**

Το τμήμα αυτό συνοψίζει τις απόψεις των κύριων κατηγοριών των ενδιαφερομένων και των προτεραιοτήτων που έχει θέσει η διοίκηση του έργου.

- ✓ Stakeholder Profiles. Οι ενδιαφερόμενοι είναι τα άτομα που εμπλέκονται ενεργά σε ένα έργο, επηρεάζονται από τα αποτελέσματά του και είναι σε θέση να επηρεάσουν την έκβασή του. Το Stakeholder Profile (Προφίλ των Ενδιαφερομένων) περιγράφει τις διαφορετικές κατηγορίες πελατών και άλλων σημαντικών παραγόντων στο έργο. Κάθε προφίλ των ενδιαφερομένων θα πρέπει να περιλαμβάνει τις ακόλουθες πληροφορίες:

  - Την αξία που θα πάρει ο κάθε ενδιαφερόμενος από το προϊόν και πώς προϊόν θα τον ικανοποιήσει. Η αξία που παίρνουν οι ενδιαφερόμενοι είναι:
    - ✓ Η αύξηση της παραγωγικότητας.
    - ✓ Η μείωση των επαναλήψεων κατά τη διάρκεια της εργασίας.
    - ✓ Η εξοικονόμηση κόστους.
    - ✓ Η αυτοματοποίηση των μέχρι πρότινος χειρονακτικών εργασιών.
    - ✓ Η ικανότητα εκτέλεσης νέων καθηκόντων.
    - ✓ Η συμμόρφωση με τις σχετικές προδιαγραφές και κανονισμούς.
    - ✓ Η βελτίωση της χρηστικότητας σε σχέση με το υπάρχον προϊόν.
  - Την πιθανή στάση τους απέναντι στο προϊόν.
  - Το ενδιαφέρον τους για τα νέα χαρακτηριστικά και στοιχεία του συστήματος.
  - Τους περιορισμούς που πρέπει να συμπεριληφθούν.

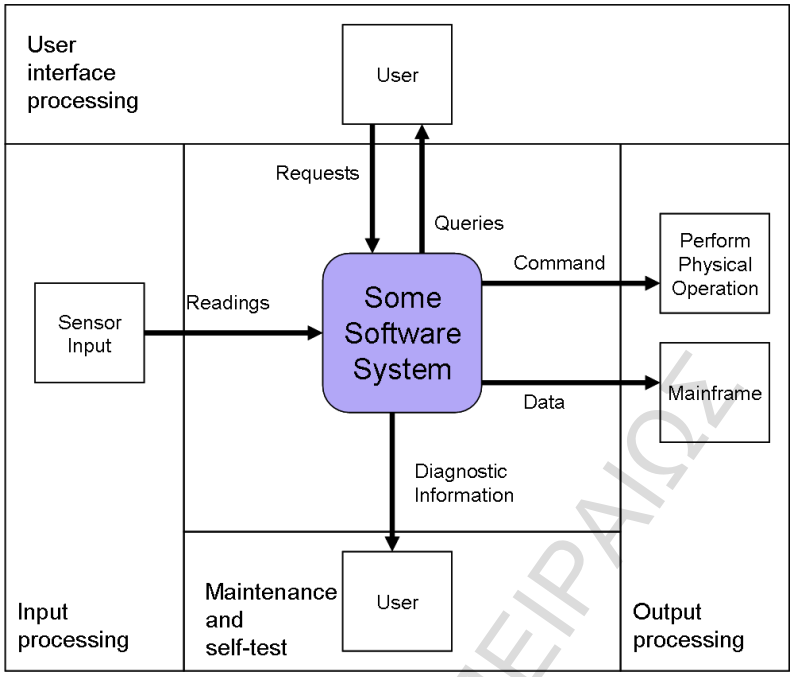
- Project Priorities. Για να καταστεί δυνατή η αποτελεσματική λήψη αποφάσεων, οι ενδιαφερόμενοι πρέπει να συμφωνήσουν σχετικά με τις προτεραιότητες του έργου. Ένας τρόπος να προσεγγίσουμε αυτό το ζήτημα είναι να εξετάσουμε τις πέντε κύριες διαστάσεις ενός έργου λογισμικού που δεν είναι άλλες από:
  - ✓ Το πεδίο εφαρμογής (τα χαρακτηριστικά).
  - ✓ Την ποιότητα.
  - ✓ Το χρονοδιάγραμμα.
  - ✓ Το κόστος.
  - ✓ Το προσωπικό.

Στόχος του Project Manager είναι να προσαρμόσει και να προσαρμοστεί στους παραπάνω παράγοντες έτσι ώστε να εκπληρώσει με επιτυχία το έργο μέσα στις περιοριστικές γραμμές που του έχουν δοθεί.

- Operating Environment. Περιγράφεται το περιβάλλον στο οποίο θα χρησιμοποιηθεί το σύστημα και ορίζονται η απόδοσή του , η αξιοπιστία του και η ακεραιότητα των απαιτήσεων. Οι πληροφορίες αυτές θα βοηθήσουν στο να οριστεί η αρχιτεκτονική του συστήματος, που είναι και το πρώτο βήμα της σχεδίασης. Ένα σύστημα το οποίο θα είναι προσβάσιμο από τους χρήστες όλο το εικοσιτετράωρο διαφέρει πολύ στην αρχιτεκτονική με ένα που θα χρησιμοποιείται σε ένα κανονικό ωράριο εργασίας. Πρέπει οι ενδιαφερόμενοι να ρωτηθούν για τα ακόλουθα:
  - ✓ Είναι οι χρήστες χωρισμένοι σε διάφορες γεωγραφικές περιοχές ή είναι συγκεντρωμένοι στον ίδιο χώρο;
  - ✓ Πότε οι χρήστες θα χρειάζονται να έχουν πρόσβαση στο σύστημα;
  - ✓ Χρειάζεται να συνδυαστούν δεδομένα από διάφορες γεωγραφικές περιοχές;
  - ✓ Θέλουμε τα δεδομένα να μπορούν να αποθηκευτούν και από απόσταση;
  - ✓ Μπορούν οι χρήστες να ανεχτούν κάποια διακοπή της υπηρεσίας ή είναι κρίσιμη η συνεχής πρόσβαση στο σύστημα;
  - ✓ Χρειάζονται διάφορα επίπεδα ελέγχου πρόσβασης στο σύστημα;

***The Context Diagram***

Η περιγραφή του πεδίου εφαρμογής ορίζει τις διαχωριστικές γραμμές και τις διασυνδέσεις μεταξύ του συστήματος που αναπτύσσεται και του εξωτερικού περιβάλλοντος. Παρακάτω αυτές οι διασυνδέσεις και τα όρια φαίνονται και διαγραμματικά. Προσδιορίζει τα τερματικά εκτός του συστήματος, που αλληλεπιδρούν όμως με αυτό, καθώς και τα δεδομένα που ανταλλάσσονται μεταξύ τους. Το Context Diagram είναι η ανώτερη βαθμίδα ενός Διαγράμματος Ροής Δεδομένων (ΔΡΔ). Μπορούμε να το συμπεριλάβουμε στο έγγραφο του οράματος και του πεδίου εφαρμογής ή ως ένα παράρτημα του Έγγραφο Προσδιορισμού των Απαιτήσεων.




**Σχήμα 23.**Το Context Diagram κάποιου υποτιθέμενου έργου λογισμικού

Ο σκοπός αυτών των διαγραμμάτων είναι να γίνει σαφής και ακριβής η επικοινωνία μεταξύ των ενδιαφερομένων. Πρέπει κατά στην κατασκευή αυτών των διαγραμμάτων να υπακούμε σε κάποιους κοινούς κανόνες έτσι ώστε να μπορούν όλοι να το διαβάσουν και να το καταλάβουν.

Οι επιχειρηματικές απαιτήσεις και το πώς οι πελάτες θα χρησιμοποιούν το προϊόν είναι πολύτιμα εργαλεία για να αντιμετωπιστεί το Scope Creep. Η αλλαγή του πεδίου εφαρμογής δεν είναι κακός παράγοντας αλλά είναι αναγκαίος στο να ικανοποιηθούν οι εξελισσόμενες ανάγκες των πελατών. Το έγγραφο οράματος και πεδίου εφαρμογής μας επιτρέπει να εκτιμήσουμε κατά πόσο οι προτεινόμενες απαιτήσεις θα πρέπει να συμπεριληφθούν στο σύστημα.

Αν η απαίτηση είναι εντός του πεδίου εφαρμογής τότε θα πρέπει να συμπεριληφθεί. Εάν είναι εκτός τότε ή θα πρέπει να απορριφθεί ή θα πρέπει να αλλάξει το πεδίο εφαρμογής και κατά συνέπεια να συμπεριληφθεί. Απόρροια αυτής της διεύρυνσης είναι και η επανεξέταση των χρηματικών πόρων, του χρονοδιαγράμματος και των υπαλλήλων που είχαν αρχικά οριστεί.

Ένα αποτέλεσμα της δημιουργίας αλλαγών είναι το ότι θα πρέπει να επαναπροσδιοριστεί η δουλειά που έχει ήδη γίνει έτσι ώστε αυτές να ενσωματωθούν στο σύστημα. Όταν το πεδίο εφαρμογής διευρύνεται αλλά οι πόροι παραμένουν οι ίδιοι, αυτό το οποίο συμβαίνει το έργο να μην ικανοποιεί την αρχικά συμφωνημένη ποιότητα.

Εάν θέλουμε το έργο να επιτευχθεί σωστά θα πρέπει οι πελάτες να παίξουν έναν καθοριστικό ρόλο. Η επιτυχία του έργου εξαρτάται από την άποψη που έχουν οι πελάτες για αυτό. Θα πρέπει

να συμμετέχουν από την αρχή της δημιουργίας του και θα πρέπει οι αναλυτές να ακούσουν "την φωνή των πελατών".

Για να γίνει αυτό θα πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Θα πρέπει να οριστούν οι κατηγορίες των πελατών του προγράμματος.
- Θα πρέπει να οριστούν οι πηγές των απαιτήσεων.
- Θα πρέπει να επιλεγούν εκπρόσωποι από κάθε κατηγορία ενδιαφερομένων.
- Πρέπει να αποφασιστεί ποιοί είναι εκείνοι που θα παίρνουν τις αποφάσεις για το έργο.

Η συμμετοχή των πελατών είναι ο μόνος τρόπος να αποφευχθεί μία απόκλιση μεταξύ των προσδοκιών που εκείνοι έχουν από το έργο και αυτού που τελικά τους παραδίδεται. Δεν αρκεί να συμμετέχουν μόνο στην αρχή του έργου αλλά σε όλη την διάρκειά του γιατί οι ανάγκες και οι απαιτήσεις τους συνεχώς αλλάζουν με την πάροδο την χρόνου.

Πολλές φορές τα χαρακτηριστικά που ζητούν να συμπεριληφθούν δεν ταυτίζονται με την λειτουργικότητα που χρειάζονται ώστε το να εκτελούν τα καθήκοντά τους στο νέο σύστημα. Ο αναλυτής θα πρέπει να συλλέγει τις απαιτήσεις που εισάγουν οι χρήστες, να τις αναλύει τις αποσαφηνίζει και να δημιουργεί αυτές που θα επιτρέπουν στον χρήστη να εκτελεί τα καθήκοντά του. Ο αναλυτής έχει την κύρια ευθύνη της καταγραφής των ικανοτήτων και ιδιοτήτων του νέου συστήματος και την ευθύνη να ενημερώνει τους υπόλοιπους ενδιαφερόμενους. Είναι μία επαναλαμβανόμενη διαδικασία που απαιτεί την συνεννόηση από όλες τις πλευρές αν θέλουμε το αποτέλεσμα να ανταποκρίνεται στις ανάγκες των χρηστών[11,24,21].

### 3.1.3 Πηγές Απαιτήσεων

Η προέλευση των απαιτήσεων του λογισμικού εξαρτάται από την φύση του προϊόντος και από το περιβάλλον ανάπτυξης. Είναι καλό οι απαιτήσεις να συγκεντρώνονται από πολλές πηγές και διαφορετικές οπτικές γωνίες. Διάφορες πηγές συγκέντρωσης απαιτήσεων είναι οι εξής:

**Συνεντεύξεις και συζητήσεις με πιθανούς χρήστες**. Ο πιο προφανής τρόπος να μάθουμε τι ζητούν οι χρήστες από ένα σύστημα είναι να τους ρωτήσουμε.

**Έγγραφα που περιγράφουν τα τρέχοντα ή ανταγωνιστικά προϊόντα**. Τα έγγραφα αυτά περιγράφουν τα βιομηχανικά πρότυπα που πρέπει να ακολουθηθούν και τους κανονισμούς που πρέπει να πληρούνται. Ιδιαίτερα σημαντικές είναι και οι αναφορές των σημερινών και των μελλοντικών επιχειρηματικών διαδικασιών. Οι αναφορές, τέλος, των ανταγωνιστικών προϊόντων βοηθούν ώστε να δώσουμε στο έργο μας ένα ανταγωνιστικό πλεονέκτημα.

**Προδιαγραφές των απαιτήσεων του συστήματος.** Ένα σύστημα που αποτελείται τόσο από λογισμικό όσο και από Hardware έχει υψηλού επιπέδου προδιαγραφές απαιτήσεων. Ο αναλυτής μπορεί να αντλήσει λειτουργικές απαιτήσεις από τα υποσυστήματα που θα συνεργάζονται με το σύστημα που πρόκειται να δημιουργηθεί.

**Εκθέσεις προβλημάτων και αιτήσεις ενίσχυσης του τρέχοντος συστήματος**. Το help-desc και το προσωπικό υποστήριξης είναι πολύτιμες πηγές πληροφόρησης. Ακούν τα προβλήματα που αντιμετωπίζουν οι χρήστες στο τρέχον σύστημα και είναι δέκτες ιδεών για το πώς θα μπορούσε αυτό να βελτιωθεί.

**Έρευνες του τμήματος Marketing και ερωτηματολόγια προς τους χρήστες.**Είναι ένας τρόπος με τον οποίο μπορεί να συγκεντρωθεί ένας μεγάλος όγκος πληροφοριών από δυνητικούς χρήστες του προϊόντος.

**Παρατήρηση των χρηστών στον χώρο εργασίας τους.** Παρατηρώντας την ροή εργασίας του χρήστη στο περιβάλλον εργασίας του, ο αναλυτής μπορεί να επικυρώσει τις πληροφορίες που έχει ήδη συλλέξει, να δει από κοντά τι δεν πάει καλά με το τρέχον σύστημα και να εξακριβώσει μόνος του τι ακριβώς θα πρέπει να παρέχει το νέο σύστημα στον χρήστη ώστε να διευκολύνει την εργασία του.

**Ανάλυση σεναρίου των εργασιών του χρήστη.** Αν ο αναλυτής προσδιορίσει τις διεργασίες που πρέπει να κάνει ο χρήστης, θα καταλάβει και την λειτουργικότητα που θα πρέπει να προσδώσει στο σύστημα για να επιτευχθούν.

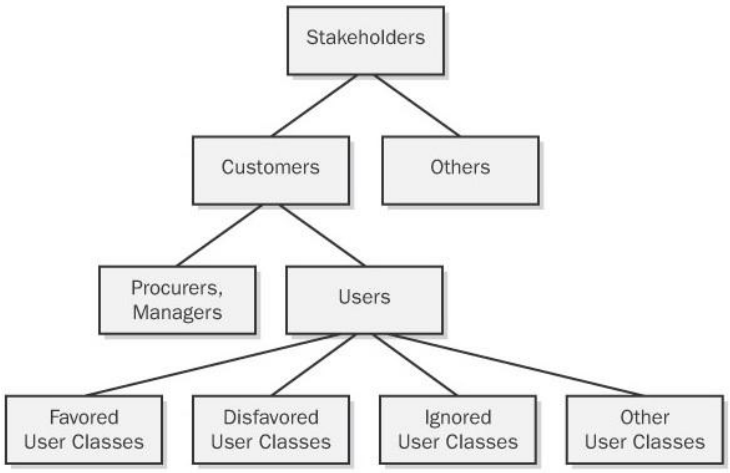
**Γεγονότα και Αντίδραση Συστήματος.** Απαρίθμηση των εξωτερικών γεγονότων (Events) στα οποία θα πρέπει να αποκριθεί το σύστημα.

**Κατηγορίες Χρηστών**

Οι κατηγορίες των χρηστών του προϊόντος διαφέρουν σύμφωνα με τα επόμενα στοιχεία [9]:

- Στην συχνότητα με την οποία χρησιμοποιούν το προϊόν.
- Στην εμπειρία και την τεχνογνωσία που έχουν στα πληροφοριακά συστήματα.
- Στα χαρακτηριστικά τα οποία χρησιμοποιούν.
- Στις διεργασίες που χρησιμοποιούν για την διενέργεια των επιχειρηματικών διαδικασιών τους.
- Στο επίπεδο ασφάλειας και πρόσβασης που έχουν.

Οι χρήστες θα πρέπει να διαφοροποιηθούν με βάσει τις παραπάνω κατηγορίες. Ένας χρήστης μπορεί να ανήκει σε διαφορετικές κατηγορίες. Οι εξωτερικοί χρήστες, αυτοί που είναι εκτός του Context Diagram είναι και αυτοί υποψήφιοι χρήστες του συστήματος. Μία κατηγορία χρηστών είναι ένα υποσύνολο των συνολικών χρηστών, που είναι υποσύνολο των πελατών, που είναι υποσύνολο των ενδιαφερομένων όπως αυτό φαίνεται και στο παρακάτω σχήμα.




**Σχήμα 24.** Διαγραμματική ιεράρχηση των ενδιαφερομένων, των πελατών και των χρηστών

Ορισμένες κατηγορίες χρηστών είναι πιο σημαντικές από κάποιες άλλες. Οι ευνοημένες κατηγορίες χρηστών λαμβάνουν ιδιαίτερη αντιμετώπιση όσον αφορά στην λήψη των αποφάσεων και στην επίλυση αντικρουόμενων απαιτήσεων με τις υπόλοιπες ομάδες. Οι ευνοημένες ομάδες είναι εκείνες που θα αποδεχθούν ή όχι πως το νέο σύστημα συμβαδίζει με τις επιχειρηματικές ανάγκες τους. Αυτό δεν σημαίνει ότι στην ομάδα αυτή συμπεριλαμβάνονται απαραίτητα εκείνοι οι ενδιαφερόμενοι που πληρώνουν ή εκείνοι που έχουν πολιτική επιρροή. Οι μη ευνοούμενες ομάδες χρηστών είναι εκείνες που δεν θα χρησιμοποιήσουν το προϊόν για νομικούς λόγους ή λόγους ασφάλειας. Αυτές οι ομάδες δεν σημαίνει ότι δεν συνυπολογιστούν κατά την δημιουργία του έργου, αλλά ότι το έργο δεν φτιάχνεται αποκλειστικά για να ικανοποιήσει τις δικές τους ανάγκες.

Κάθε κατηγορία χρηστών έχει τις δικές της απαιτήσεις για τις διεργασίες που πρέπει τα μέλη της να εκτελέσουν. Μπορεί επίσης να έχουν και διαφορετικές λειτουργικές απαιτήσεις, όπως απαιτήσεις χρηστικότητας, από τους υπόλοιπους. Συνήθως οι περιστασιακοί χρήστες, που ασχολούνται σε μικρό ποσοστό με το σύστημα, ζητούν ένα πιο φιλικό περιβάλλον εργασίας, συντομέυσεις και γραφικά που θα τους βοηθούν στην εργασία τους.

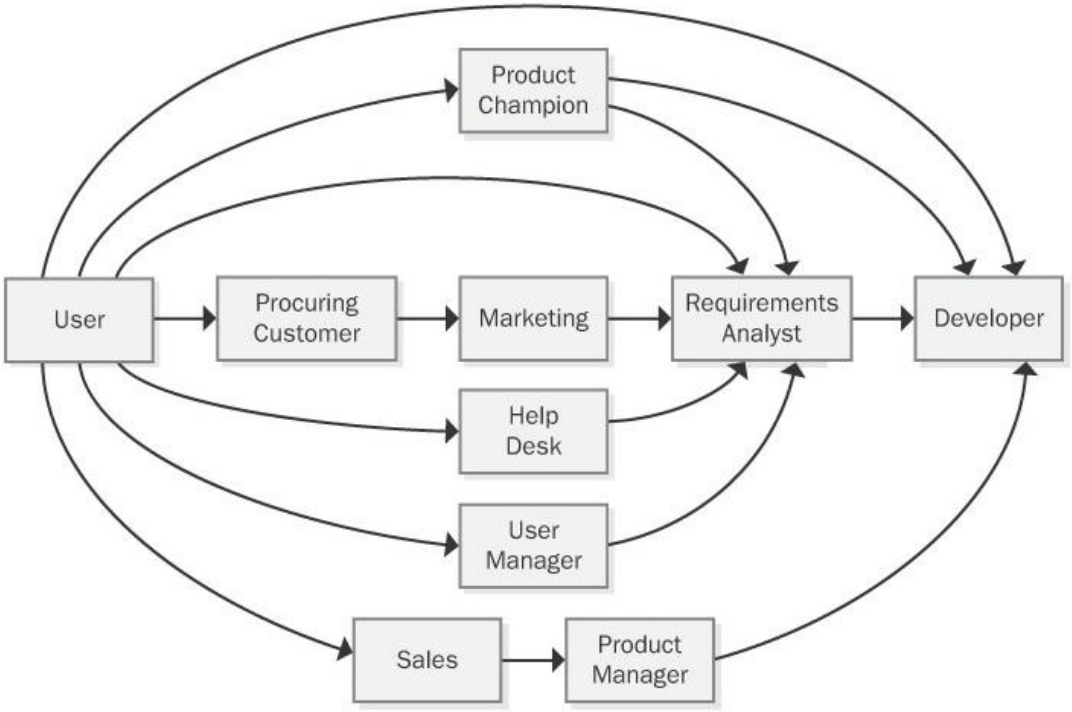
Ορισμένες κατηγορίες χρηστών μπορεί να μην είναι άνθρωποι. Μπορεί να είναι στοιχεία λογισμικού ή hardware με τα οποία το σύστημά μας αλληλεπιδρά. Τα στοιχεία αυτά όμως δεν μπορούν να "ζητήσουν" τις ανάλογες απαιτήσεις και έτσι ο αναλυτής θα πρέπει να τα συνυπολογίσει και να βρει τις κατάλληλες για να συνεχίσουν να αλληλεπιδρούν.

Πρέπει οι διαφορετικές κατηγορίες χρηστών να οριστούν από την αρχή του έργου για να μπορούν να αποσπαστούν οι απαιτήσεις από τους εκπροσώπους τους. Μία χρήσιμη τεχνική για αυτό ονομάζεται "Expand Then Contract". Πρέπει να γίνει ανταλλαγή απόψεων με όλες τις κατηγορίες χρηστών και είναι ιδιαίτερα σημαντικό να μην παραβλεφθεί καμία. Αν οι ανάγκες των ομάδων συμπίπτουν ,θα πρέπει να αντιμετωπιστούν σαν ένα ενιαίο σύνολο.

Θα πρέπει λοιπόν να καταγραφούν σε έναν κατάλογο όλες οι ομάδες των χρηστών και οι απαιτήσεις που έχει η κάθε ομάδα ξεχωριστά. Με τον τρόπο αυτό θα ξέρουμε τι ακριβώς πρέπει να γίνει στο έργο έτσι ώστε να μείνουν όλοι ικανοποιημένοι. Στην κατηγοριοποίηση θα πρέπει να συνυπολογίσουμε και την γεωγραφική θέση των χρηστών αφού αποτελεί μία από τις συνιστώσες της διαφοροποίησης.

Κάθε είδος έργου, συμπεριλαμβανομένων των πληροφοριακών συστημάτων, των εμπορικών εφαρμογών και των εφαρμογών διαδικτύου χρειάζεται τους κατάλληλους εκπροσώπους των χρηστών που θα αναδείξουν τις ανάγκες τους. Θα πρέπει να συμβάλουν σε όλη τη διάρκεια του έργου και όχι μόνο στο αρχικό στάδιο του εντοπισμού των απαιτήσεων. Στην δημιουργία των ομάδων των εκπροσώπων θα πρέπει να περιλαμβάνονται τόσο άτομα με μεγάλη εμπειρία όσο και άτομα με μικρότερη.

Στο Σχήμα 25 φαίνονται μερικοί τρόποι επικοινωνία των συμμετεχόντων για να έχουμε όσο δυνατόν καλύτερο αποτέλεσμα στο τέλος. Έχει παρατηρηθεί ότι όσο περισσότερα κανάλια επικοινωνίας υπάρχουν τόσο πιο επιτυχημένο είναι ένα έργο. Το αποτέλεσμα είναι ακόμα καλύτερο αν η ομάδα ανάπτυξης εκτελέσει ένα μέρος της δουλειάς του αναλυτή και μιλήσουν οι ίδιοι με τους χρήστες.




**Σχήμα 25.** Πιθανά μονοπάτια επικοινωνίας μεταξύ των χρηστών και της ομάδας ανάπτυξης

Με τον τρόπο αυτό προστίθεται αξία στο τελικό προϊόν. Είναι πολύ σημαντικό ένας έμπειρος αναλυτής να συνεργάζεται με τους άλλους ενδιαφερόμενους στην συλλογή, την αξιολόγηση, τον ορισμό και την οργάνωση των απαιτήσεων. Είναι πολύ σημαντικό να ακούγεται η φωνή των

χρηστών του συστήματος αφού εκείνοι είναι που τελικά θα το δουλεύουν. Είναι μεν μία χρονοβόρα και δύσκολη διαδικασία αλλά καθόλα σημαντική για την επιτυχία του εγχειρήματος [5,20,21].

***Ο Product Champion***

Ο Product Champion είναι ο σύνδεσμος μεταξύ των μελών της κάθε ομάδας χρηστών με τον αναλυτή απαιτήσεων του έργου. Ιδανικά θα πρέπει να είναι και οι ίδιοι χρήστες του συστήματος και όχι υποκατάστατα όπως οικονομικοί χορηγοί, μέλη της ομάδας marketing ή προγραμματιστές λογισμικού που προσπαθούν να καταλάβουν τις ανάγκες του χρήστη. Δουλειά τους είναι να συλλέγουν τις απαιτήσεις τις ομάδας τους και να τις διαπραγματεύονται με τους αναλυτές. Αυτό που πρέπει να κατανοηθεί πως η διαδικασία ανάλυσης των απαιτήσεων είναι μία συλλογική δουλειά από όλα τα μέλη του οργανισμού και των ενδιαφερομένων.

Πρέπει να έχουν ένα σαφή όραμα του προγράμματος και να επικοινωνούν συνεχώς με τους αναλυτές αλλά και με την ομάδα τους. Εξουσιοδοτείται, από την ομάδα που αντιπροσωπεύει, να πάρει σημαντικές αποφάσεις για την πορεία του έργου. Οι αποφάσεις που παίρνουν θα πρέπει να γίνονται σεβαστές τόσο από την ίδια την ομάδα τους, όσο και από την διοίκηση του οργανισμού. Όταν δεν δεν υπάρχει σωστή επικοινωνία και μεταφορά απόψεων, είναι πιθανό το έργο να αντιμετωπίσει προβλήματα με την πάροδο του χρόνου.

**Εξωτερικός Product Champions (External Product Champion).** Στα έργα που αφορούν ανάπτυξη εμπορικού λογισμικού είναι δύσκολο να βρεθούν Product Champions εντός της εταιρείας, αφού οι χρήστες δεν θα είναι εργαζόμενοι στον οργανισμό. Στην περίπτωση αυτή τον ρόλο του Product Champion θα μπορούσε να πάρουν κάποιοι κύριοι συνεργαζόμενοι πελάτες. Ακόμα όμως και στην περίπτωση αυτή οι External product managers δεν θα μιλούσαν εκ μέρους όλων των εν δυνάμει πελατών. Θα πρέπει λοιπόν να οριστούν οι κύριες απαιτήσεις που είναι κοινές σε όλη την γκάμα των πελατών και να στην συνέχεια να οριστούν συμπληρωματικές απαιτήσεις σε συγκεκριμένους πελάτες, τμήματα της αγοράς ή άλλες κατηγορίες χρηστών. Η επιλογή του External Product Manager θα πρέπει να γίνεται με συνέπεια. Θα πρέπει να έχει το κατάλληλο υπόβαθρο και τις αντίστοιχες γνώσεις για να μπορεί να ανταπεξέλθει σε μία τόσο κρίσιμη για το έργο διαδικασία.

***Δραστηριότητες του Product Champion***

Για να θεωρηθεί επιτυχημένος ο ρόλος του κάθε Product Champion θα πρέπει να έχει κάνει τα ακόλουθα:

Όσον αφορά τον σχεδιασμό:

- Να ορίσει την εμβέλεια και τους περιορισμούς του έργου.
- Να καθορίσει τις διεπαφές με τα υπόλοιπα συστήματα.
- Να αξιολογήσει την επίπτωση του συστήματος στις επιχειρηματικές διεργασίες.

- Να ορίσει ένα μονοπάτι μετάβασης από τις τρέχουσες εφαρμογές.

Όσον αφορά τις απαιτήσεις:

- Να συλλέξει τις απαιτήσεις από την ομάδα του.
- Να αναπτύξει σενάρια και περιπτώσεις χρήσης.
- Να επιλύσει τις συγκρούσεις μεταξύ των προτεινόμενων απαιτήσεων.
- Να ορίσει προτεραιότητες εφαρμογής των απαιτήσεων.
- Να ορίσει της απαιτήσεις ποιότητας και απόδοσης του συστήματος.
- Να αξιολογήσει τα Prototypes των διεπαφών των χρηστών (User Interface).

Όσον αφορά την επαλήθευση και την επικύρωση:

- Να ελέγξει τα έγγραφα των απαιτήσεων.
- Να καθορίσει τα κριτήρια αποδοχής των χρηστών.
- Να δημιουργήσει περιπτώσεις δοκιμών από τα σενάρια χρήσης.
- Να παρέχει τα δεδομένα των δοκιμών.
- Να ελέγχει τις εκδόσεις του έργου.

Όσον αφορά την ομάδα του:

- Να γράφει εγχειρίδιαχρήσης και κείμενα βοηθείας.
- Να ετοιμάζει εκπαιδευτικό υλικό για τα σεμινάρια.
- Να αξιολογεί τις αιτήσεις αλλαγών και την επίπτωση που θα έχουν δουλειά των χρηστών που εκπροσωπεί.

Όσον αφορά την διοίκηση:

- Να αξιολογεί τις επιδιορθώσεις.
- Να αξιολογεί και να ιεραρχεί τις αιτήσεις αλλαγών και την επίπτωση που θα έχουν στην επιχειρηματική διαδικασία.
- Να συμμετέχει στην λήψη αποφάσεων αλλαγών.

Ο ρόλος του Product Champion είναι πολύ σημαντικός για την επιτυχία του έργου,αφού αυτός θα διαπραγματευτεί με την κάθε ομάδα χρηστών και με την διοίκηση. Εκείνος θα πάρει την απόφαση να επιλυθούν οι αντικρουόμενες απαιτήσεις των χρηστών και μετά θα τους εξηγήσει τον λόγο που τον οδήγησαν σε αυτή την απόφαση. Δεν υπάρχει κάτι που να ορίζει τι "προαπαιτούμενα" έχει ρόλος του ProductChampion αλλά το σίγουρο είναι πως πρέπει να οριστεί από την αρχή του έργου για να μην χαθεί χρόνος ή δημιουργηθούν προβλήματα στην πορεία [5,14,21].

### 3.1.4 Διαδικασίες Εκμαίευσης Απαιτήσεων

Η καρδιά της Μηχανικής των Απαιτήσεων είναι η εκμαίευση (elicitation). Είναι η διαδικασία προσδιορισμού των αναγκών και των περιορισμών των διάφορων ενδιαφερομένων μερών ενός συστήματος λογισμικού. Η εκμαίευση επικεντρώνεται στην ανακάλυψη των απαιτήσεων των χρηστών. Οι απαιτήσεις αυτές είναι ένα μόνο επίπεδο των απαιτήσεων του συνολικού έργου. Τα άλλα δύο επίπεδα είναι οι λειτουργικές και οι επιχειρηματικές απαιτήσεις. Οι απαιτήσεις των χρηστών περιλαμβάνουν τις διεργασίες εκείνες που οι χρήστες προσδοκούν να μπορούν να πραγματοποιήσουν μέσω του συστήματος που είναι προς δημιουργία. Αφορούν την απόδοση, την χρηστικότητα και άλλα ποιοτικά χαρακτηριστικά του συστήματος [9].

Η ανάπτυξη των απαιτήσεων είναι μία κοινή κατανόηση των αναγκών μεταξύ των ενδιαφερομένων του συστήματος. Όταν οι αναλυτές καταλάβουν αυτές τις ανάγκες μπορούν να διερευνήσουν εναλλακτικές λύσεις για να τις ικανοποιήσουν. Οι συμμετέχοντες στην εκμαίευση πρέπει πρώτα να κατανοήσουν πλήρως το πρόβλημα και τις ανάγκες και μετά να προχωρήσουν στον σχεδιασμό. Σε διαφορετική περίπτωση θα πρέπει να γίνει επανασχεδιασμός του συστήματος, όσο οι απαιτήσεις ορίζονται με μεγαλύτερη ακρίβεια. Για να παραμείνει η ομάδα προσανατολισμένη είναι επιτακτική ανάγκη να δίνεται μεγαλύτερη σημασία στις απαιτήσεις των χρηστών και στην ρίζα του προβλήματος, παρά στις διεπαφές των χρηστών (User Interfaces).

Πρέπει να γίνει προσδιορισμός της δραστηριότητας εκμαίευσης απαιτήσεων.Ακόμα και ένα απλό σχέδιο δράσης αυξάνει την επιτυχία και θέτει ρεαλιστικές προσδοκίες στους ενδιαφερόμενους. Πρώτα από όλα θα πρέπει να γίνει μία ρητή δέσμευση των διαθέσιμων πόρων, του χρονοδιαγράμματος και των παραδοτέων. Το πλάνο θα πρέπει να συγκεντρώνεται στα εξής:

**Εκμαίευση των στόχων**. Επικύρωση δηλαδή των στοιχείων της αγοράς, διερεύνηση των περιπτώσεων χρήσης και ανάπτυξη ενός λεπτομερούς συνόλου λειτουργικών απαιτήσεων του συστήματος.

**Εκμαίευση των στρατηγικών και των διαδικασιών**. Θα πρέπει να γίνουν έρευνες, συνεντεύξεις με χρήστες, επισκέψεις σε εν δυνάμει πελάτες και άλλες τακτικές προσέγγισης σε διάφορες ομάδες ενδιαφερομένων.

**Εργαλεία προσπάθειας εκμαίευσης**. Λίστες δηλαδή από περιπτώσεις χρήσης, μία λεπτομερής SRS, ανάλυση των αποτελεσμάτων της έρευνας και ποιοτικά χαρακτηριστικά επίδοσης και ποιότητας.

**Πρόγραμμα και πηγές εκτίμησης**. Πρέπει να εκτιμηθούν και οριστούν τόσο οι συμμετέχοντες της δραστηριότητας εκμαίευσης, όσο και το χρονικό διάστημα που θα διαρκέσει.

**Κίνδυνοι εκμαίευσης.** Πρέπει να γίνει προσδιορισμός των παραγόντων μπορεί να παρεμποδίσουν την ολοκλήρωση της διαδικασίας, να εκτιμηθεί η σοβαρότητα του κάθε κινδύνου και να βρεθεί μία λύση για το πώς μπορούν αυτοί οι κίνδυνοι να μετριαστούν και να ελεγχθούν.

### 3.1.5 Τεχνικές Εκμαίευσης Απαιτήσεων

Κάθε προσπάθεια ανάπτυξης λογισμικού, είτε πρόκειται για μικρές εφαρμογές είτε για μεγάλα έργα λογισμικού, ξεκινά με σκοπό την κάλυψη κάποιων αναγκών ή την παροχή επιπλέον υπηρεσιών. Για την ανάπτυξη οποιουδήποτε πληροφοριακού συστήματος, είναι αναγκαίο να σημειωθούν επακριβώς και να αναλυθούν όλες οι απαιτήσεις που σχετίζονται με αυτό.

Η εκμαίευση απαιτήσεων είναι η διαδικασία κατά την οποία οι απαιτήσεις αυτές διαπιστώνονται, τόσο από τους αναλυτές όσο και από τους ίδιους τους χρήστες, και καταγράφονται. Όσο απλό και να φαίνεται, διατυπωμένο με αυτό τον τρόπο, οι εμπειρία έχει δείξει ότι είναι συχνές οι αποτυχίες κατά τη διάρκεια αυτής της πρώτης φάσης ανάπτυξης ενός συστήματος.

Συγκεκριμένα, σύμφωνα με έρευνες που έλαβαν χώρα το, έχει διαπιστωθεί ότι το 31% των έργων λογισμικού ακυρώνονταν πριν να ολοκληρωθούν. Διαπιστώθηκε ότι οι κύριοι λόγοι ήταν οι εξής:

- Μη πλήρεις απαιτήσεις (13.1%)
- Έλλειψη ενασχόλησης χρηστών (12.4%)
- Έλλειψη πόρων (10.6%)
- Μη ρεαλιστικές προσδοκίες (9.9%)
- Έλλειψη διευθυντικής υποστήριξης (9.3%)
- Αλλαγή απαιτήσεων και προδιαγραφών (8.7%)
- Έλλειψη σχεδιασμού (8.1%)
- Το σύστημα δε χρειαζόταν πλέον πουθενά (7.5%)

Τα λάθη στη συλλογή των απαιτήσεων πληρώνονται πολύ ακριβά. Αν για παράδειγμα το κόστος για την ανεύρεση και επίλυση ενός προβλήματος που αφορά τις απαιτήσεις είναι ένα ευρώ κατά τη διάρκεια της εκμαίευσης και ανάλυσης των απαιτήσεων, είναι πέντε ευρώ κατά τη διάρκεια της σχεδίασης, δέκα ευρώ κατά τη διάρκεια της συγγραφής του κώδικα, είκοσι ευρώ κατά τη διάρκεια ελέγχου και διακόσια ευρώ μετά την παράδοση του συστήματος. Εύλογο είναι το συμπέρασμα ότι πρέπει να δίνεται ιδιαίτερη σημασία στη διαδικασία εκμαίευσης απαιτήσεων, και να γίνεται προσεκτική επιλογή της μεθόδου ή του συνδυασμού μεθόδων που θα χρησιμοποιηθεί, ώστε να προκύψουν όσο το δυνατόν πληρέστερες και αντικειμενικές απαιτήσεις [10,26,27,28,29].

### 3.1.6 Μέθοδοι εκμαίευσης απαιτήσεων

***Ενδοσκόπηση (Introspection).*** Η μέθοδος την ενδοσκόπησης, όπως υπονοείται από το όνομά της, είναι η προσπάθεια του αναλυτή απαιτήσεων να σκεφτεί πώς θα ήθελε να λειτουργεί το σύστημα αν ήταν στη θέση του χρήστη. Πολλές φορές είναι η μόνη επιλογή που έχει ο αναλυτής, καθώς οι πελάτες δεν είναι σε θέση να καταλάβουν τη σημασία της συμβολής τους στη διαδικασία της ανάλυσης απαιτήσεων και δεν είναι πρόθυμοι να αφιερώσουν χρόνο για να βοηθήσουν τον αναλυτή [9].

Παρόλο που η ενδοσκόπηση μπορεί να βοηθήσει τον αναλυτή να φτάσει σε χρήσιμα συμπεράσματα, έχει πολλούς περιορισμούς και αδυναμίες. Ο αναλυτής είναι ειδικός στην Μηχανική Απαιτήσεων, αλλά όχι και στο γνωστικό αντικείμενο του τελικού χρήστη. Ως εκ τούτου, είναι αναμενόμενο να παραβλέψει σημεία, τα οποία ο χρήστης θεωρεί σημαντικά. Επίσης, είναι πιθανό, να υπερτιμήσει την ικανότητα του χρήστη να κατανοήσει το σωστό τρόπο χειρισμού του συστήματος, κρίνοντας από τη δική του εμπειρία στο χειρισμό παρόμοιων συστημάτων. Το πρόβλημα μεγεθύνεται από την αδιαφορία των χρηστών να διορθώσουν τυχόν προβλήματα που παρουσιάζονται, κατατάσσοντάς τα σε άλλη μία ενοχλητική ιδιοτροπία του υπολογιστή.

Παρόλο που η Γνωσιακή Ψυχολογία απορρίπτει την ενδοσκόπηση ως μη αξιόπιστη μέθοδο έρευνας, στην περίπτωση της Μηχανικής Απαιτήσεων, μπορεί να φανεί χρήσιμη στο βαθμό που δίνει στον αναλυτή ιδέες και κατεύθυνση για περαιτέρω αναζήτηση. Όμως, είναι σαφώς πιο ασφαλές, να αφιερώνεται χρόνος από τους χρήστες, ώστε να αποφεύγονται οι δυσάρεστες εκπλήξεις κατά την παράδοση του συστήματος. Για το σκοπό αυτό έχει διαπιστωθεί ότι είναι επωφελές να αφιερώνεται χρόνος για την επιμόρφωση των χρηστών πάνω σε θέματα Μηχανικής Απαιτήσεων. Οι εμπλεκόμενοι δείχνουν περισσότερη προθυμία να συμμετάσχουν αφού καταλάβουν τη σημασία των απαιτήσεων και τις δυσκολίες που πρέπει να αντιμετωπισθούν.

***Συνεντεύξεις (Interviews).*** Οι συνεντεύξεις είναι πιθανότατα η πιο διαδεδομένη τεχνική εκμαίευσης απαιτήσεων. Θεωρείται ότι ακόμα και ένας άπειρος αναλυτής μπορεί να αποκτήσει χρήσιμες πληροφορίες, μετά από λίγες μόνο ώρες συνεντεύξεων. Όμως, είναι σημαντική η δυσκολία οργάνωσης, δόμησης, επεξεργασίας της πληροφορίας. Οι αναλυτές ίσως δυσκολευτούν να εξηγήσουν το σκεπτικό τους στους χρήστες, οι οποίοι δεν καταλαβαίνουν πως προέκυψαν οι απαιτήσεις που τους παρουσιάζονται.

Η μέθοδος αυτή παρουσιάζει το πλεονέκτημα της άμεσης επικοινωνίας με τους χρήστες του συστήματος, δεδομένου, φυσικά, ότι έχει γίνει σωστή επιλογή ερωτήσεων και συμμετέχουν στη διαδικασία όλες οι ομάδες χρηστών. Όμως, δεν αποκλείεται το γεγονός η επικοινωνία αυτή να είναι προβληματική, εάν ο αναλυτής δεν έχει φροντίσει η γλώσσα και το λεξιλόγιο που χρησιμοποιείται να είναι κατανοητό, τόσο από τον ίδιο όσο και από τους χρήστες. Παρακάτω θα παρουσιαστούν τρεις τύποι συνεντεύξεων:

- Οι συνεντεύξεις με ερωτηματολόγια.
- Οι συνεντεύξεις ανοιχτού τύπου.
- Οι συνεντεύξεις σε ομάδες εστίασης και ανάπτυξης εφαρμογής.

**Συνεντεύξεις με ερωτηματολόγια (Questionnaires).** Περιλαμβάνει ερωτηματολόγια με ερωτήσεις κλειστού τύπου. Είναι ο λιγότερο ευέλικτος τύπος συνεντεύξεων, καθώς οι ερωτήσεις είναι κλειστού τύπου και αυστηρά προκαθορισμένες. Λόγω των χαρακτηριστικών αυτών, τα ερωτηματολόγια μπορούν να χρησιμοποιηθούν για στατιστική ανάλυση. Δε θα πρέπει όμως να παραβλέπεται το γεγονός ότι η στατιστική ανάλυση είναι χρήσιμη μόνο όταν το πλήθος των

χρηστών του συστήματος είναι σημαντικό. Από την άλλη πλευρά, όταν το σύστημα προορίζεται για χρήση από μεγάλο αριθμό χρηστών, είναι σαφώς ο πιο απλός και γρήγορος τρόπος για να καταγραφούν όλες οι διαφορετικές απόψεις.

Τα ερωτηματολόγια έχουν το μειονέκτημα ότι ακόμα και οι πιο προσεκτικά διατυπωμένες ερωτήσεις αφήνουν περιθώρια παρερμηνείας, ενώ δεν δίνεται δυνατότητα επεξηγήσεων. Επιπροσθέτως, δεν είναι δυνατό να προσαρμοστούν οι ερωτήσεις ανάλογα με την πορεία των απαντήσεων. Το αποτέλεσμα είναι να παραβλέπονται, τυχόν ενδιαφέροντες παράμετροι με μια απλή σημείωση στην κατηγορία «Άλλο» ή «Δεν ξέρω/Δεν απαντώ». Φαίνεται λοιπόν ότι η μέθοδος είναι ανεπαρκής για την εκμαίευση υπονοούμενης γνώσης λόγω των υπερβολικά στενών ορίων αλληλεπίδρασης μεταξύ των χρηστών και του αναλυτή.

Ένα ακόμα μειονέκτημα είναι η δυσκολία σύνταξης πλήρων και αμερόληπτων ερωτηματολογίων. Εκτός από τη διατύπωση των ερωτήσεων, ο αναλυτής πρέπει να λάβει υπόψη τον αριθμό των ερωτήσεων, το εύρος και τον τύπο των κατηγοριών των απαντήσεων, τις οδηγίες συμπλήρωσης, ακόμα και τη σειρά των ερωτήσεων.

**Συνεντεύξεις ανοιχτού τύπου (Open Ended Interviews).** Στις συνεντεύξεις ανοιχτού τύπου, ο ερωτών κάνει ερωτήσεις τις οποίες ο ερωτούμενος μπορεί να απαντήσει όπως επιθυμεί. Η επικοινωνία είναι πιο ελεύθερη, και ευέλικτη από ότι στις συνεντεύξεις κλειστού τύπου και έτσι, μπορεί να εξαχθεί περισσότερη λεπτομέρεια. Επίσης ο αναλυτής μπορεί να προσαρμόσει επιτόπου την προσέγγισή του ανάλογα με την προσωπικότητα του χρήστη. Ως εκ τούτου, οι συνεντεύξεις ανοιχτού τύπου ξεπερνούν τα μειονεκτήματα των ερωτηματολογίων, όμως είναι ακόμα πιο δύσκολο να γίνει επεξεργασία της πληροφορίας που εξάγεται. Ένα σημαντικό πλεονέκτημα της μεθόδου, είναι ότι μπορεί να οδηγήσει σε απροσδόκητα συμπεράσματα και σε εκμαίευση υπονοούμενων πληροφοριών.

Ένα άλλο πρόβλημα είναι η τάση των ανθρώπων να παραλείπουν κάποιες, ενδεχομένως σημαντικές, πτυχές που θεωρούν προφανείς, καθώς και το γεγονός ότι ένας ειδικός κάποιου τομέα μπορεί να θεωρήσει ανιαρό να επαναλαμβάνει πληροφορίες που θεωρεί τετριμμένες.

Ομάδες εστίασης (FocusGroups). Πρόκειται για ένα είδος ομαδικής συνέντευξης. Παραλλαγές της μεθόδου είναι οι λεγόμενες «ομάδες από κοινού ανάπτυξης» (Joint Application Development) και «ομάδες γρήγορης ανάπτυξης» (Rapid Application Development). Ο ερευνητής σχηματίζει μία ομάδα από ενδιαφερόμενους και κατευθύνει τις συζητήσεις με ερωτήσεις, συνήθως ανοιχτού τύπου και οι συμμετέχοντες είναι ελεύθεροι να συζητούν μεταξύ τους. Η τεχνική αυτή χρησιμοποιείται ευρέως σε έρευνες αγοράς, αλλά η χρησιμότητά της στην εκμαίευση απαιτήσεων είναι αμφισβητούμενη.

Οι υποστηρικτές της μεθόδου ισχυρίζονται ότι η μέθοδος συμβάλλει στο να ξεπερνώνται οι επικοινωνιακές δυσκολίες, λόγω της φυσικότητας της αλληλεπίδρασης σε σχέση με τις προηγούμενες μεθόδους συνεντεύξεων . Δίνεται η ευκαιρία στους εμπλεκόμενους να ανταλλάσσουν γνώμες και ιδέες καθώς και να βρίσκουν κοινό τόπο μεταξύ των διαφορετικών

απόψεων. Τέλος, οι ομάδες εστίασης θεωρούνται ένας εναλλακτικός, γρήγορος και φθηνός τρόπος εξαγωγής απαιτήσεων.

Οι πολέμιοι της μεθόδου από την άλλοι υποστηρίζουν ότι η επικοινωνία δεν είναι τόσο αυθόρμητη, λόγω του ότι οι ομάδες σχηματίζονται κατά την κρίση του αναλυτή και δεν αποτελούν πραγματικές κοινωνικές ομάδες. Ελλοχεύει ο κίνδυνος της «ομαδικής σκέψης» (group think), γεγονός που οδηγεί σε μεροληψία, αυτό-λογοκρισία, πίεση και εκλογίκευση. Η τεχνική των ομάδων εστίασης δεν θεωρείται αρκετή για την εκμαίευση χρήσιμης και πάνω από όλα αξιόπιστης γνώσης.

***Ανάλυση Πρωτοκόλλου (Protocol Analysis).*** Η διαδικασία της ανάλυσης πρωτοκόλλου είναι η ανάθεση μιας συγκεκριμένης εργασίας στο χρήστη, την οποία θα πρέπει να προσπαθήσει να φέρει εις πέρας, περιγράφοντας ταυτόχρονα τις σκέψεις του. Κατόπιν ο αναλυτής θα μελετήσει τη συμπεριφορά του ώστε να καταλήξει σε ένα δομημένο μοντέλο της γνώσης του χρήστη με σκοπό την εξαγωγή χρήσιμων συμπεράσματα σχετικά με τα αντικείμενα που βλέπει ο χρήστης, τα χαρακτηριστικά τους, τις συσχετίσεις μεταξύ τους και τα συμπεράσματα που εξάγονται από αυτές.

Ένα από τα κύρια πλεονεκτήματα της μεθόδου, είναι το γεγονός ότι ένας χρήστης μπορεί ευκολότερα να μιλάει για τη διαδικασία που ακολουθεί όταν έχει ένα συγκείμενο πρόβλημα να λύσει παρά όταν μιλάει αφηρημένα για αυτό, και φυσικά δεν χάνεται πληροφορία επειδή δεν υπάρχει καθυστέρηση από τη στιγμή που γίνεται η εργασία μέχρι την περιγραφή της διαδικασίας και έτσι δεν εξασθενεί η μνήμη του χρήστη. Επίσης, το βασικότερο επιχείρημα υπέρ της μεθόδου της ανάλυσης πρωτοκόλλου, είναι ότι πρόκειται για τη μόνη μέθοδο που περιγράφει με ακρίβεια η γνωστική διαδικασία του χρήστη.

Το επιχείρημα αυτό όμως αμφισβητείται έντονα, καθώς θεωρείται ότι η ανάλυση πρωτοκόλλου δεν είναι αξιόπιστη μέθοδος για να κατανοήσουμε τη λειτουργία του ανθρώπινου μυαλού και ότι είναι εύκολο να παρερμηνευτεί λόγω των υπεραπλουστευμένων και τεχνητά δημιουργημένων προβλημάτων που χρησιμοποιούνται, πολλές φορές με σκοπό να επιβεβαιωθούν τα προειλημμένα συμπεράσματα των ερευνητών. Επίσης υπάρχουν γλωσσικοί περιορισμοί όπως και στην περίπτωση των ανοιχτού τύπου συνεντεύξεων καθώς και δυσκολίες που προκύπτουν από τις αντιδράσεις των χρηστών στην έρευνα [29,30,31].

### 3.1.7 Αδυναμίες μεθόδων εκμαίευσης απαιτήσεων

Οι αδυναμίες των μεθόδων που παρουσιάστηκαν μέχρι τώρα οφείλονται στην αδυναμία τους να λάβουν υπόψη τις κοινωνικές και ψυχολογικές συνθήκες του «κόσμου» των χρηστών. Οι άνθρωποι δεν είναι δυνατόν να είναι απόλυτα αντικειμενικοί και ανεπηρέαστοι από το περιβάλλον τους. Ειδικά στην περίπτωση της εκμαίευσης απαιτήσεων, όπου πολλοί διαφορετικοί άνθρωποι, με διαφορετικό υπόβαθρο, κίνητρα και σκοπούς, πρέπει να συνεργαστούν και να βρουν έναν τρόπο συνεννόησης ώστε το αποτέλεσμα να τους διευκολύνει

όλους. Κάποιοι παράγοντες που επηρεάζουν τις κοινωνικές συνθήκες είναι η τοποθεσία, οι κατηγορίες χρηστών και οι μέθοδοι χρηστών.

Όπως διαπιστώθηκε στην περίπτωση της ανάλυσης πρωτοκόλλου, οι άνθρωποι επηρεάζονται από το περιβάλλον στο οποίο γίνεται η έρευνα. Σε κάθε περίπτωση προσπαθούν να ερμηνεύσουν τις συνθήκες που αντιμετωπίζουν και να προσαρμοστούν ανάλογα, οπότε οι εργαστηριακές συνθήκες δεν εγγυώνται αμερόληπτη έρευνα.

Οι χρήστες ενός συστήματος δεν εμπίπτουν όλοι στην ίδια κατηγορία. Ενδεχομένως να διαχωρίζονται σε έμπειρους και άπειρους χρήστες, ειδικευμένους και ανειδίκευτους, προϊσταμένους και υπαλλήλους, άντρες και γυναίκες. Ο αναλυτής οφείλει να διαπιστώσει τις κατηγορίες στις οποίες κατατάσσουν οι ίδιοι οι χρήστες τους εαυτούς τους, αντί να κάνει δικές του υποθέσεις, και να αποφασίσει ποιες από αυτές τις κατηγορίες είναι σημαντικές για την ανάλυσή του και για ποιο λόγο.

Όπως είδαμε κάθε μέθοδος παρουσιάζει κάποιους περιορισμούς. Η τεχνική της ενδοσκόπησης, αν και χρήσιμη, σε καμία περίπτωση δεν επαρκεί από μόνη της καθώς παρουσιάζει μόνο την οπτική του αναλυτή. Η μέθοδος των συνεντεύξεων με ερωτηματολόγια επιτρέπει ελάχιστη αλληλεπίδραση μεταξύ αναλυτή και χρήστη και είναι ιδιαίτερα ευάλωτη σε παρερμηνείες και μεροληψίες. Από την άλλη πλευρά, οι συνεντεύξεις ανοιχτού τύπου, αν και πιο ευέλικτες από τα ερωτηματολόγια, επίσης περιορίζονται από γλωσσικούς παράγοντες και είναι επιρρεπείς σε παρερμηνείες. Η μέθοδος των ομάδων εστίασης, δε λαμβάνει υπόψη τις κοινωνικές συνθήκες, επομένως είναι πιθανό να παραλειφθούν στοιχεία και να εξαχθούν λανθασμένα συμπεράσματα.

Τέλος, η ανάλυση πρωτοκόλλου είναι επίσης εξαιρετικά περιορισμένη και αγνοεί παντελώς οποιαδήποτε κοινωνική αλληλεπίδραση. Η ανάλυση συνδιάλεξης, φαίνεται να ξεπερνά πολλούς από τους περιορισμούς των παραπάνω μεθόδων, επειδή δίνεται σημασία στις κοινωνικές συνθήκες των χρηστών, οι οποίες ερμηνεύονται από τη δική τους οπτική και όχι από την οπτική του αναλυτή. Επιτρέπεται περισσότερο ευέλικτη επικοινωνία και αυτό επίσης συμβάλλει στην εκμαίευση μεγαλύτερης λεπτομέρειας και υπονοούμενης γνώσης. Παρόλα αυτά η μέθοδος ωφελεί ιδιαίτερα μόνο σε συστήματα στα οποία είναι σημαντική η κοινωνική αλληλεπίδραση.

Ένας ακόμα σημαντικός περιορισμός είναι ο χρόνος ο οποίος πρέπει να αφιερωθεί στην ανάλυση, γεγονός το οποίο την καθιστά ακατάλληλη για συστήματα που πρέπει να αναπτυχθούν σε μικρό χρονικό διάστημα. Προτείνεται συνδυασμός των μεθόδων των συνεντεύξεων και της ανάλυσης συνδιάλεξης, ώστε να διαπιστώνονται πρώτα οι βασικές απαιτήσεις και έπειτα να αναλύονται περαιτέρω τα σημεία που κρίνονται σημαντικότερα.

Η χρησιμοποίηση εργαλείων εκμαίευσης μπορεί να διευκολύνει την δουλειά του αναλυτή των απαιτήσεων. Θα μπορούσε κάποιος να κάνει τον ρόλο του διαμεσολαβητή ο οποίος θα είναι υπεύθυνος για την επιλογή των συμμετεχόντων και την επιτυχή έκβαση της διαδικασίας, αφήνοντας έτσι τον αναλυτή να αφιερώσει την πλήρη προσοχή του μόνο στα θέματα προς

συζήτηση. Παρακάτω υπάρχουν μερικές διεργασίες που θα βοηθήσουν στην επιτυχή διαδικασία της εκμαίευσης απαιτήσεων:

**Δημιουργία κανόνων.** Οι συμμετέχοντες θα πρέπει να συμφωνήσουν σε μερικούς κανόνες λειτουργίας:

- Οι συνεδριάσεις θα ξεκινούν και θα λήγουν στην ώρα τους.
- Θα επιστρέφουν αμέσως μετά τα διαλείμματα.
- Ο καθένας θα πρέπει να συμβάλει στην διαδικασία.

**Παραμονή στο πεδίο εφαρμογής**. Θα πρέπει να συμβουλευόμαστε το έγγραφο οράματος και πεδίου εφαρμογής για να είμαστε σίγουροι ότι η προτεινόμενη απαίτηση εμπίπτει μέσα σε αυτό. Η ομάδα θα πρέπει να μένει συγκεντρωμένη στους στόχους της κάθε ημέρας και να μην αναλώνεται σε λεπτομερείς και θέματα εκτός προγράμματος.

**Καταγραφή των στοιχείων.** Σε μία τέτοια διαδικασία θα αναφερθούν μία σειρά από πληροφορίες που αφορούν χαρακτηριστικά ποιότητας, επιχειρηματικούς κανόνες, περιορισμούς, ιδέες για userinterfaces και μία σειρά από άλλα δεδομένα. Θα πρέπει αυτές οι πληροφορίες όχι μόνο να καταγράφονται, για κάποια μελλοντική επανεξέταση, αλλά θα πρέπει να καταγράφονται και εκείνοι που τα ανέφεραν.

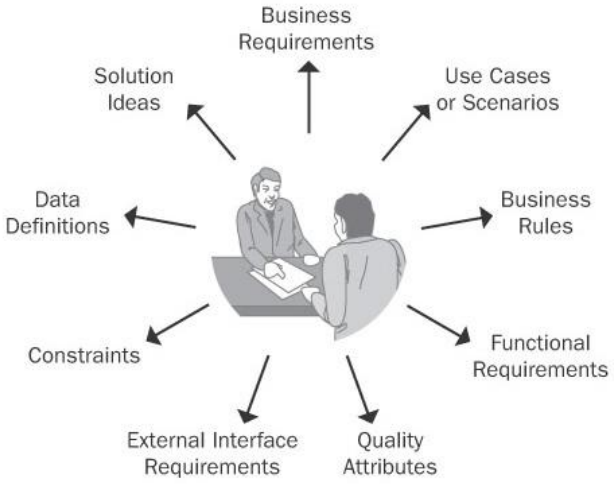
**Συζητήσεις καθορισμένου χρόνου (Time box Discussions).** Στο κάθε θέμα θα πρέπει να δίνεται ένα χρονικό περιθώριο , ας υποθέσουμε 30 λεπτά, για να συζητηθεί. Αυτό μας βοηθάει στο να δώσουμε δυνατότητα συζήτησης για όλα τα θέματα της ατζέντας και να μην επικεντρωθούμε μόνο στα πρώτο ή πρώτα από αυτά.

**Δημιουργία μικρών και αποδοτικών ομάδων**. Μία μικρή ομάδα είναι πολύ πιο ευέλικτη και γρήγορη από μία μεγάλη. Μία ομάδα με πάνω από πέντε ή έξι άτομα μπορεί να οδηγήσει την κουβέντα σε παράλληλες συζητήσεις και αντιπαραθέσεις. Για να κερδηθεί χρόνος θα μπορούσαν να γίνουν παράλληλες συζητήσεις με διαφορετικές ομάδες χρηστών. Οι συμμετέχοντες θα πρέπει να περιλαμβάνουν τον Product Champion, άλλους εκπροσώπους χρηστών, κάποιον εμπειρογνώμονα, έναν αναλυτή απαιτήσεων και ένα μέλος της ομάδας ανάπτυξης. Οι κριτήρια επιλογής των συμμετεχόντων θα πρέπει να είναι η γνώση, η εμπειρία και η εξουσιοδότηση να λαμβάνει αποφάσεις.

**Θα πρέπει όλοι να ασχολούνται.** Πολλές φορές κάποιος συνομιλητής δεν συμμετέχει στην κουβέντα επειδή νομίζει ότι δεν έχει να πει κάτι σημαντικό και δεν θέλει να διακόψει την διαδικασία. Καμιά φορά δεν συμμετέχει γιατί δεν επιθυμεί την δημιουργία ενός νέου συστήματος. Είναι όμως επιτακτική ανάγκη η συμμετοχή όλων στην συζήτηση εκμαίευσης απαιτήσεων για να αποφευχθούν μελλοντικές παρεξηγήσεις.

Δεν πρέπει να περιμένουμε από τους πελάτες να παρουσιάσουν μία συνοπτική, πλήρης και καλά οργανωμένη λίστα των αναγκών τους. Η ταξινόμηση των απαιτήσεων από τις διάφορες ομάδες

των χρηστών θα πρέπει να γίνει από τους αναλυτές. Στο παρακάτω Σχήμα 26 φαίνονται οι διάφορες κατηγορίες απαιτήσεων πουφτάνουν στον αναλυτή από την διαδικασία της εκμαίευσης:




**Σχήμα 26.** Κατηγοριοποίηση των αναγκών των πελατών

Οι πληροφορίες που δεν σχετίζονται με το παραπάνω σχήμα έχουν να κάνουν με:

- Απαιτήσεις που δεν σχετίζονται με ανάπτυξη λογισμικού, όπως αυτές που αφορούν την εκπαίδευση των χρηστών στο νέο σύστημα.
- Περιορισμοί του έργου, όπως περιορισμοί κόστους ή χρονοδιαγράμματος.
- Υποθέσεις
- Πρόσθετες πληροφορίες που αφορούν το ιστορικό κάποιου χαρακτηριστικού.

Η ένταξη των απαιτήσεων στις παρακάτω κατηγορίες θα μας βοηθήσει κατά την διαδικασία της συλλογής τους:

**Επιχειρηματικές απαιτήσεις**. Οτιδήποτε περιγράφει οικονομικά στοιχεία, την αγορά ή επιχειρηματικά οφέλη που οι πελάτες ή η ομάδα ανάπτυξης επιθυμούν να κερδίσουν από το προϊόν, καταγράφονται σαν επιχειρηματικές απαιτήσεις.

**Περιπτώσεις χρήσης ή σενάρια.** Οι δηλώσεις που έχουν να κάνουν με τους στόχους των χρηστών ή επιχειρηματικές διεργασίες που επιθυμούν να εκτελούν μέσω του συστήματος, κατατάσσονται σαν περιπτώσεις χρήσης ή σενάρια. Τέτοια σενάρια προέρχονται από την περιγραφή της ροής εργασίας των χρηστών. Ένας άλλος τρόπος συλλογής περιπτώσεων χρήσης είναι να ρωτηθεί ο χρήστης για τον στόχο που έχει στο μυαλό του όταν πρόκειται να κάνει μία διεργασία.

**Επιχειρηματικοί κανόνες.** Όταν ένας πελάτης λέει ότι υπό ορισμένες προϋποθέσεις μπορεί να εκτελέσει κάποιες συγκεκριμένες διεργασίες, τότε περιγράφει κάποιον επιχειρηματικό κανόνα.

**Λειτουργικές απαιτήσεις**. Λειτουργικές απαιτήσεις είναι η συμπεριφορά που θα παρουσιάσει το σύστημα κάτω από ορισμένες συνθήκες και οι διεργασίες που το σύστημα επιτρέπει στους χρήστες να εκτελέσουν. Προκύπτουν από τις απαιτήσεις του συστήματος, τις απαιτήσεις των χρηστών, τους επιχειρηματικούς κανόνες και από το Έγγραφο Προσδιορισμού των Απαιτήσεων.

**Χαρακτηριστικά ποιότητας.** Οι δηλώσεις που υποδηλώνουν πόσο καλά συμπεριφέρεται το σύστημα σε κάποιες διεργασίες και κατά πόσο επιτρέπει στους χρήστες να κάνουν την δουλειά τους, είναι τα χαρακτηριστικά ποιότητας του συστήματος. Συνήθως έχουν να κάνουν δηλώσεις όπως : γρήγορο, αξιόπιστο, φιλικό, ασφαλές και αποδοτικό. Πρέπει οι στόχοι ποιότητας να είναι σαφείς και επαληθεύσιμοι.

**Εξωτερικές απαιτήσεις διεπαφής.** Εδώ περιγράφονται οι συνδέσεις που θα έχει το σύστημα με το εξωτερικό περιβάλλον. Θα πρέπει να περιλαμβάνονται συνδέσεις με χρήστες, λογισμικό και hardware.

**Περιορισμοί.** Οι περιορισμοί θα περιορίσουν τις διαθέσιμες επιλογές για την υλοποίηση του έργου. θα πρέπει νε έχουμε περιορισμούς που αφορούν την συμβατότητα, το μέγεθος και τις επιτρεπόμενες συνδέσεις. Θα πρέπει να καταγράφονται τόσο οι περιορισμοί, όσο και το σκεπτικό πίσω από αυτές και ποιοί τις καταθέτουν. Οι περιττοί περιορισμοί εμποδίζουν την δημιουργία της βέλτιστης λύσης σχεδίασης του συστήματος. Για τον λόγο αυτό θα πρέπει είναι συγκεκριμένοι και εμπεριστατωμένοι. Οι σωστοί περιορισμοί βοηθούν στην επίτευξη των ποιοτικών στόχων.

**Ορισμοί δεδομένων.** Κάθε φορά που οι πελάτες περιγράφουν μία μορφή ή έναν τύπο δεδομένων, τις επιτρεπόμενες τιμές εισόδου και εξόδου του συστήματος ή μία σύνθετη δομή επιχειρηματικών δεδομένων, στην ουσία ορίζουν τα δεδομένα. Θα πρέπει να δημιουργηθεί ένα λεξικό δεδομένων στο οποίο βασίζεται η ομάδα κατά την ανάπτυξη του έργου. Ο ορισμός των δεδομένων οδηγεί σε λειτουργικές απαιτήσεις που ο χρήστης δεν έχει ζητήσει άμεσα.

**Ιδέες επίλυσης.**Πολλά από αυτά που παρουσιάζουν οι χρήστες σαν απαιτήσεις, εμπίπτουν σε αυτή την κατηγορία. Όποιος περιγράφει ένα υποτιθέμενο τρόπο που θα αλληλεπιδρά με το σύστημα, στην ουσία προτείνει μία ιδέα αλληλεπίδρασής του με το σύστημα. Η δουλειά του αναλυτή είναι να εξετάσει αυτή την ιδέα και να φτάσει στην σωστή απαίτηση.

Εάν δεν χρησιμοποιηθεί ένα δομημένο σύστημα οργάνωσης, όπως τα usecases, θα είναι πολύ δύσκολο να συγχωνευτούν οι απαιτήσεις όλων των χρηστών. Είναι πολύ σημαντικό να ακουστούν όλες οι απόψεις αν θέλουμε να έχουμε ένα επιτυχημένο έργο. Είναι πολύ σημαντικό λοιπόν να οριστούν σωστά οι Product Champions που θα εκπροσωπήσουν όλους τους χρήστες του συστήματος.

Κατά την διάρκεια της εκμαίευσης των απαιτήσεων μπορεί να διαπιστωθεί πως το πεδίο εφαρμογής του έργου είναι είτε μεγάλο, είτε μικρό. Αν είναι μεγάλο συνεπάγεται την συλλογή περισσότερων απαιτήσεων που θα χρειαστούν για να εξασφαλίσουν την επάρκεια των

επιχειρηματικών διεργασιών, πράγμα το οποίο σημαίνει ότι θα αφιερωθεί πολύς χρόνος στην συγκεκριμένη διαδικασία. Αν από την άλλη το πεδίο εφαρμογής είναι μικρό, τότε δεν θα αποδοθεί ένα προϊόν που θα ικανοποιήσει τις ανάγκες των χρηστών. Άρα λοιπόν η διαδικασία της εκμαίευσης μπορεί να οδηγήσει σε επαναπροσδιορισμό του πεδίου εφαρμογής προς όφελος του έργου.

Οι απαιτήσεις σχετίζονται με το τί θα μπορεί να κάνει το σύστημα, ενώ το πώς θα υλοποιηθεί η λύση έχει να κάνει τον σχεδιασμό. Η εκμαίευση όντως επικεντρώνεται στο τί πρέπει να κάνει το σύστημα, αλλά η γραμμή που διαχωρίζει την ανάλυση με τον σχεδιασμό είναι λεπτή και δύσκολα διακριτή. Πρέπει λοιπόν να αποσαφηνιστεί στους χρήστες πώς τα διαγράμματα που τους παρέχονται κατά την εκμαίευση είναι για να τους βοηθήσει να κατανοήσουν πως πληρούνται οι ανάγκες τους και δεν είναι απαραίτητα το τελικό σχέδιο λύσης.

Όταν προτείνεται μία ιδέα θα πρέπει να εκτιμηθεί πριν ληφθεί κάποια απόφαση ενσωμάτωσής της στο έργο. Αυτές οι ιδέες θα πρέπει να αντιμετωπιστούν σαν ξεχωριστές διεργασίες με τους δικούς τους στόχους και τις δικές τους ξεχωριστές απαιτήσεις. Το prototyping είναι ένας τρόπος να αντιμετωπίσουμε τέτοια ζητήματα. Αν πάλι η ιδέα αυτή απαιτεί εκτενή έρευνα θα πρέπει να προσεγγίσουμε μία σταδιακή ανάπτυξη για διερευνηθούν οι απαιτήσεις που χρειάζονται [21,25].

### 3.1.8 Εκμαίευση Αόρατων Απαιτήσεων

Ένα σημαντικό εμπόδιο κατά τη διάρκεια της μηχανικής των απαιτήσεων είναι οι Missing Requirements. Είναι δύσκολο να εντοπιστούν γιατί είναι αόρατες. Παρακάτω υπάρχουν ορισμένες τεχνικές που θα μας βοηθήσουν αν τις ανακαλύψουμε.

- Αποσύνθεση των απαιτήσεων υψηλού επιπέδου σε λεπτομέρειες για να αποκαλυφθεί τι ακριβώς ζητείται. Αυτές οι απαιτήσεις προκαλούν ένα χάσμα μεταξύ του τι ήθελε αυτός που την ζήτησε και του τι τελικά έκανε η ομάδα ανάπτυξης. Οι ανακριβείς και συγκεχυμένοι όροι οδηγούν σε λανθασμένη υλοποίηση, υποστήριξη και διαχείριση.
- Να βεβαιώσουμε ότι όλες οι ομάδες χρηστών έχουν προσφέρει στην διαδικασία και ότι η κάθε περίπτωση χρήσης έχει τουλάχιστον μία ομάδα που την έχει προτείνει.
- Πρέπει οι λειτουργικές απαιτήσεις να ανιχνεύονται σε όλες τις απαιτήσεις του συστήματος, τις περιπτώσεις χρήσης και τους επιχειρηματικοί κανόνες, έτσι ώστε να βεβαιωθεί ο αναλυτής ότι έχει προσδιοριστεί όλη απαραίτητη λειτουργικότητα.
- Θα πρέπει να παρουσιάζονται οι πληροφορίες των απαιτήσεων με πολλούς τρόπους. Πολλές φορές είναι ευκολότερο να καταλάβουμε μία εικόνα, ένα διάγραμμα, ένα μοντέλο και ένα δέντρα απόφασης από ένα σύνθετο έγγραφο. Με την καλύτερη κατανόηση των απαιτήσεων μπορούμε να ευκολότερα να βρούμε τυχόν λάθη και παραλείψεις.

Ένας ακόμη τρόπος να βρούμε τις αόρατες απαιτήσεις (Missing Requirements) είναι η δημιουργία της CRUD Matrix. Το αρκτικόλεξο της CRUD σημαίνει Create, Read, Update και Delete (Δημιούργησε, Διάβασε, Ενημέρωσε και Διέγραψε). Σχετίζει την δράση του

συστήματος με οντότητες δεδομένων για να ξέρουμε ακριβώς πού και πώς το κάθε δεδομένο έχει δημιουργηθεί, διαβαστεί, ενημερωθεί και διαγραφεί. Μερικοί την CRUD Matrix την ονομάζουν CRUDL (List - Κατηγοριοποίηση). Μερικά είδη συσχετίσεων είναι τα ακόλουθα:

- Οντότητες δεδομένων με συμβάντα του συστήματος.
- Οντότητες δεδομένων με διεργασίες χρηστών ή περιπτώσεις χρήσης.
- Κλάσεις αντικειμένων με συμβάντα το συστήματος .
- Κλάσεις αντικειμένων με περιπτώσεις χρήσης.

Παρακάτω, στον Πίνακα 3 φαίνεται μία CRUDL Matrix. Κάθε κελί δείχνει πώς κάθε περίπτωση χρήσης, στην αριστερή στήλη, χρησιμοποιεί κάθε οντότητα δεδομένων, των υπολοίπων στηλών. Η κάθε περίπτωση χρήσης μπορεί να C (δημιουργήσει), R (διαβάσει), U (ενημερώσει), D (διαγράψει) ή L (κατηγοριοποιήσει) μία οντότητα δεδομένων. Μετά την δημιουργία της λίστας θα ελέγξουμε τα γράμματα που έχουν μπει σε κάθε κελί και αν αυτά βγάζουν νόημα. Για παράδειγμα αν μία απαίτηση έχει γίνει U, έχει δηλαδή ενημερωθεί, θα πρέπει στην ίδια απαίτηση να υπάρχει και το γράμμα C, πως έχει δηλαδή δημιουργηθεί. Αν δεν βγαίνει νόημα με τις συσχετίσεις των γραμμάτων στα κελιά τότε κάπου έχει γίνει λάθος και χρειάζεται να επανεξεταστούν οι συγκεκριμένες απαιτήσεις.

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 |
|---|---|---|---|---|---|---|---|---|---|
| Task1 | C | C | | | | | | | |
| Task2 | | R | | | | C | | | |
| Task3 | | | | | | R | C | | |
| Task4 | R | | C | | | | | | |
| Task5 | R | R | | C | | | | | |
| Task6 | | R | R | R | C | | | | |
| Task7 | R | U | | | R | | R | | |
| Task8 | | | | | | | R | C | |
| Task9 | R | R | | | | | | R | C |
| (C=Create, R=Read, U=Update, D=Delete) | | | | | | | | | |

**Πίνακας 3.** Παράδειγμα CRUDL Matrix

Δεν υπάρχει κάποιο ξεκάθαρο μήνυμα ότι έχει τελειώσει η διαδικασία της εκμαίευσης απαιτήσεων. Συνεχώς θα παράγονται νέες ιδέες για επιπρόσθετες απαιτήσεις. Υπάρχουν όμως κάποια σημάδια που δηλώνουν ότι η διαδικασία φτάνει στο τέλος της.

- Εάν οι χρήστες δεν μπορούν να σκεφτούν άλλες περιπτώσεις χρήσης.
- Εάν οι περιπτώσεις χρήσης που ειπώνονται προέρχονται από λειτουργικές απαιτήσεις άλλων περιπτώσεων χρήσης.
- Εάν οι χρήστες επαναλαμβάνουν ζητήματα που έχουν καλυφθεί σε προηγούμενες συζητήσεις.
- Εάν οι προτεινόμενες απαιτήσεις χρηστών ή οι προτεινόμενες λειτουργικές απαιτήσεις είναι εκτός του πεδίου εφαρμογής του έργου.

- Εάν οι προτεινόμενες απαιτήσεις είναι χαμηλής προτεραιότητας.
- Εάν οι προτεινόμενες δυνατότητες δεν αφορούν την συγκεκριμένη έκδοση του προγράμματος αλλά μία μεταγενέστερη.

Ένας άλλος τρόπος είναι να συγκεντρωθούν σε ένα έγγραφο όλες οι απαιτήσεις που αφορούν διάφορες λειτουργικές περιοχές του έργου. Θα πρέπει αυτή η λίστα να συγκρίνεται με τις λειτουργίες που έχουν ήδη οριστεί. Εφόσον από αυτή την σύγκριση δεν προκύπτουν κενά, τότε η διαδικασία εκμαίευσης οδεύει προς το τέλος της.

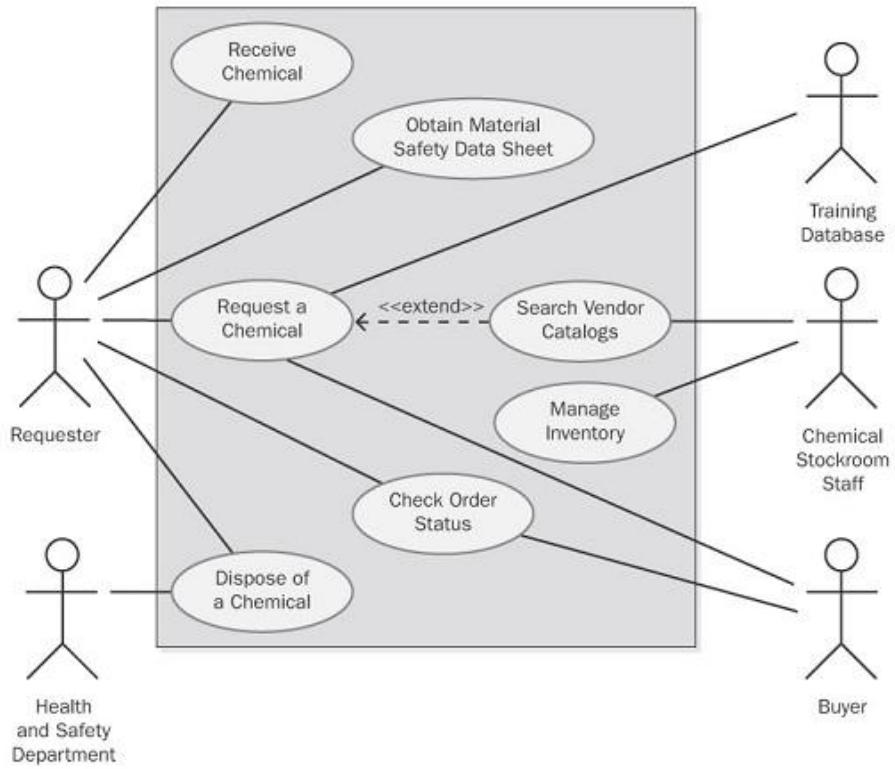
Παρά την επιθυμία μας να ανακαλύψουμε όλες τις απαιτήσεις, κάτι τέτοιο δεν είναι εύκολο να γίνει. Όσο θα προχωράει η κατασκευή του έργου θα γίνονται συνεχώς αλλαγές. Στόχος είναι οι απαιτήσεις να είναι αρκετά καλές ώστε να επιτρέπουν την κατασκευή του έργου με ένα επιτρεπτό επίπεδο κινδύνου [10,21].

### 3.1.9 Δημιουργία Περιπτώσεων Χρήσης και Σεναρίων για την Εκμαίευση των Λειτουργικών Απαιτήσεων

Μία περίπτωση χρήσης περιγράφει μία σειρά από αλληλεπιδράσεις μεταξύ του συστήματος και μιας εξωτερικής πηγής. Αυτή η εξωτερική πηγή μπορεί να είναι ένα άτομο, ένα σύστημα λογισμικού ή μία συσκευή hardware που αλληλεπιδρά με το σύστημα για να επιτευχθεί ένας στόχος. Μία άλλη ονομασία της εξωτερικής πηγής είναι ο εξωτερικός χρήστης. Οι περιπτώσεις χρήσης προέκυψαν από την αντικειμενοστραφή ανάπτυξη, ωστόσο μπορούν να χρησιμοποιηθούν από οποιαδήποτε προσέγγιση ανάπτυξης και αν χρησιμοποιείται. Είναι στο κέντρο της ευρύτερης κατηγορίας Unified Software Development Process.

Η usecase προσέγγιση ζητάει από τους χρήστες να δηλώσουν τι επιθυμούν οι ίδιοι να κάνουν δια μέσω του συστήματος και όχι τι επιθυμούν το σύστημα για αυτούς. Στόχος του είναι να περιγραφούν όλες οι διεργασίες που οι χρήστες επιθυμούν να διεξάγουν. Τα ενδιαφερόμενα μέρη πριν αποδεχτούν το κάθε usecaseπρέπει να εξασφαλίσουν ότι εμπίπτει στο πεδίο εφαρμογής του έργου. Θεωρητικά τα usecase θα πρέπει να περιλαμβάνουν όλες τις επιθυμητές διεργασίες, αλλά στη πράξη κάτι τέτοιο είναι πολύ δύσκολο. Θα περιλαμβάνει όμως το μεγαλύτερο σύνολο των τελικών διεργασιών.

Τα διαγράμματα των περιπτώσεων χρήσης παρέχουν μία υψηλού επίπεδου οπτική παρουσίαση των απαιτήσεων των χρηστών. Το Σχήμα 27 μας δείχνει ένα τέτοιο διάγραμμα, χρησιμοποιώντας UML σημειογραφία.Το κουτί αντιπροσωπεύει το όριο του συστήματος, οι γραμμές μας δείχνουν πως αλληλεπιδρά ο κάθε χρήστης (οι εικόνες εκτός πλαισίου) με τις περιπτώσεις χρήσης (τα οβάλ εντός πλαισίου).



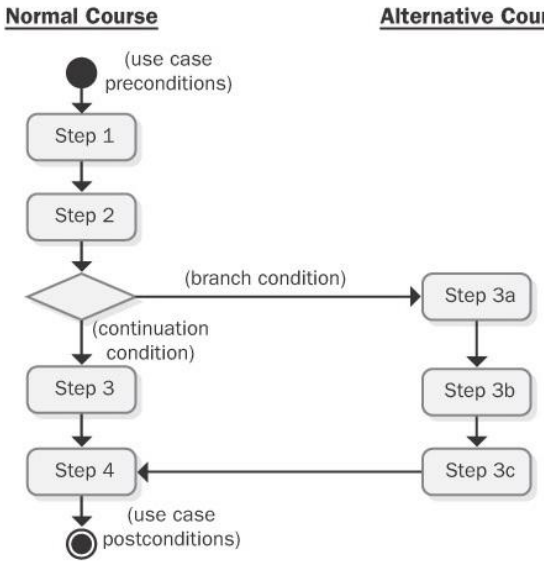

**Σχήμα 27**. Παράδειγμα ενός διαγράμματος περίπτωσης χρήσης

***Περιπτώσεις Χρήσης και Σενάρια***

Μία περίπτωση χρήσης είναι είτε μία διακριτή και αυτόνομη διεργασία που μπορεί να εκτελέσει ένας χρήστης, είτε να περιλαμβάνει μία σειρά από παρόμοιες εργασίες που έχουν έναν κοινό στόχο. Η περίπτωση χρήσης δηλαδή είναι μία συλλογή από σενάρια και ένα σενάριο είναι ένα συγκεκριμένο παράδειγμα μίας περίπτωσης χρήσης. Όταν διερευνούμε τις απαιτήσεις των χρηστών μπορούμε είτε να ξεκινήσουμε με αφηρημένες περιπτώσεις χρήσης και να οδηγηθούμε σε συγκεκριμένα σενάρια είτε και το αντίστροφο, να ξεκινήσουμε δηλαδή από ένα συγκεκριμένο σενάριο και να οδηγηθούμε σε μία γενικευμένη περίπτωση χρήσης. Τα βασικά στοιχεία περιγραφής μίας περίπτωσης χρήσης είναι τα ακόλουθα:

- Ένα μοναδικό χαρακτηριστικό.
- Ένα όνομα που να δηλώνει συνοπτικά την εργασία των χρηστών. Συνήθως είναι της μορφής " Ρήμα + Αντικείμενο".
- Μία σύντομη περιγραφή κειμένου σε φυσική γλώσσα.
- Ένας κατάλογος προϋποθέσεων που πρέπει να πληρούνται πριν ξεκινήσει η περίπτωση χρήσης.
- Ένα κείμενο που να περιγράφει την κατάσταση του συστήματος μετά την επιτυχή ολοκλήρωση της περίπτωσης χρήσης.
- Μία αριθμημένη λίστα που να δείχνει την ακολουθία των βημάτων που έγιναν μεταξύ του χρήστη και του συστήματος.

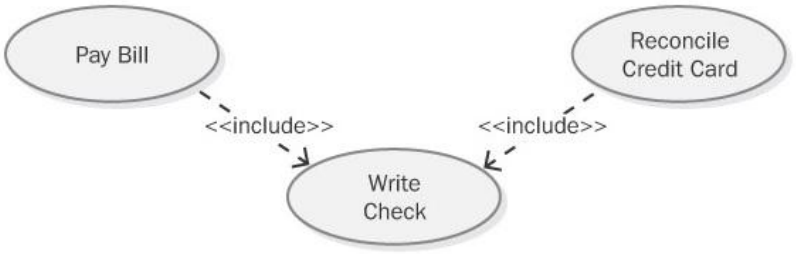
Ένας τύπος σεναρίου που ταυτίζεται με την πορεία μιάς περίπτωσης χρήσης ονομάζεται κύριο σενάριο ή βασικό σενάριο επιτυχίας. Υπάρχουν όμως και άλλα σενάρια που ονάζονται εναλλακτικά ή δευτερεύοντα. Οδηγούν και αυτά σε επιτυχή ολοκλήρωση των διεργασιών, ωστόσο αντιπροσωπεύουν διάφορες ιδιαιτερότητες της εργασίας που γίνονται για να εκτελεστεί το έργο. Μία διεργασία μπορεί να διακλαδωθεί σε κάποια αναλλακτική πορεία και στην συνέχεια να επιστρέψει στην φυσιολογική της. Οι περισσότερες περιπτώσεις χρήσης μπορούν να περιγραφούν σε ένα απλό κείμενο, αλλά μία σύνθετη περίπτωση θα πρέπει να απεικονιστεί με ένα διάγραμμα ροής δεδομένων ή ένα διάγραμμα δραστηριοτήτων (UML). Όπως φείνεται και από το Σχήμα 28 τέτοια διαγράμματα μπορούμε να δείξουν τις εναλλακτικές πορείες που μπορεί να χρησιμοποιηθούν.




**Σχήμα 28**. UML διάγραμμα δραστηριοτήτων που δείχνει εναλλακτικούς τρόπους υλοποίησης μίας περίπτωσης χρήσης

Μερικά από τα βήματα σε μία εναλλακτική πορεία θα είναι ίδια με την κύρια, αλλά ορισμένες δραστηριότητες θα παρεκλίνουν σε εναλλακτικά μονοπάτια. Αν μία εναλλακτική περίπτωση χρήσης είναι αυτόνομη, τότε μπορούμε να συμπεριληφθεί σαν μία ξεχωριστή περίπτωση.

Πολλές φορές οι περιπτώσεις χρήσης μοιράζονται ένα σύνολο κοινών βημάτων. Για να αποφευχθεί η επικάλυψη των βημάτων σε κάθε περίπτωση θα πρέπει να οριστεί ένα ξεχωριστό σενάριο χρήσης που θα περιλαμβάνει την κοινή λειτουργικότητα και να δηλωθεί πως όλες οι περιπτώσεις χρήσης περιέχουν αυτήν την "υποπερίπτωση". Παρακάτω, στο Σχήμα 29, φαίνεται πώς δύο υποπεριπτώσεις περιλαμβάνονται στην κύρια περίπτωση χρήσης.



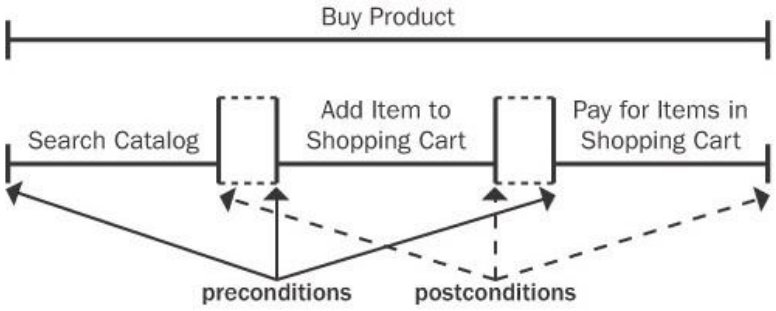

**Σχήμα 29.** Διαγραμματική απεικόνιση μίας κύριας περίπτωσης χρήσης που πριλαμβάνει δύο υποπεριπτώσεις

Υπάρχουν κάποιες συνθήκες, οι εξαιρέσεις, που εμποδίζουν μία διεργασία να ολοκληρωθεί επιτυχώς. Αν αυτές οι εξαιρέσεις δεν καθοριστούν κατά την διάρκεια της εκμαίευσης τότε υπάρχουν δύο πιθανές επιπτώσεις:

- Η ομάδα ανάπτυξης θα κάνει τις δικές της εικασίες για το πώς θα αντιμετωπίσουν τις εξαιρέσεις.
- Το σύστημα θα αποτύχει αν ο χρήστης επιλέξει την λάθος κατάσταση.

Δεν είναι απαραίτητο να εφαρμοστούν όλα τα εναλλακτικά σενάρια, αλλά θα πρέπει να εφαρμοστούν οι εξαιρέσεις, αφού είναι κρίσιμος παράγοντας για την επιτυχία του έργου. οι εξαιρέσεις είναι το μεγαλύτερο μέρος της προσπάθειας κωδικοποίησης. Αν θέλουμε να δημιουργήσουμε ένα ισχυρό προϊόν λογισμικού θα πρέπει όλες οι εξαιρέσεις να καθοριστούν.

Σε μερικά συστήματα ο χρήστης μπορεί να ενώσει μικρότερες πειρπτώσεις χρήσης σε μία μεγλύτερη "Macro Use Case" που περιγράφει μία μεγαλύτερη διεργασία του έργου. Αν αυτές οι μικρότερες περιπώσεις χρήσης εκτελούν μία ανεξάρτητη δραστηριότητα μπορούν να θεωρηθούν σαν ανεξάρτητες περιπτώσεις. Απαραίτητη προϋπόθεση όμως για να γίνει αυτό θα πρέπει η κάθε μεμονωμενη περίπτωση χρήσης να αφήνει το σύστημα σε μία κατάσταση που να επιτρέπει τον χρήστη να αρχίσει αυτόματα την επόμενη περίπτωση χρήσης. Θα πρέπει δηλαδή οι περιπτώσεις χρήσης να ευθυγραμμίζονται και η κάθε μία να δημιουργεί τις προϋποθέσεις να εφαρμοστεί η επόμενη.




**Σχήμα 30.** Προϋποθέσεις εφαρμογής των επόμενων περιπτώσεων χρήσης

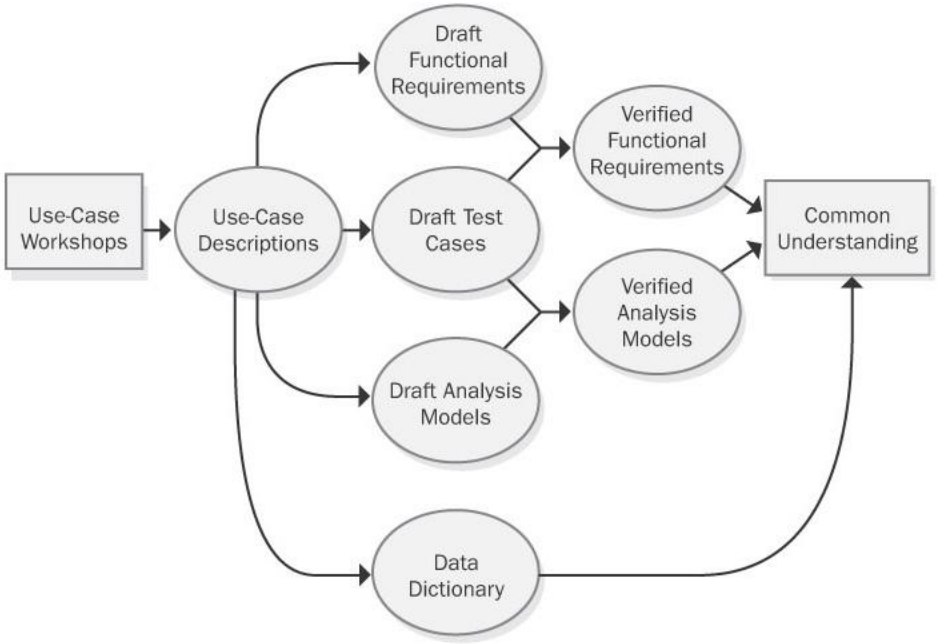
**Προσδιορισμός περιπτώσεων χρήσης**
Ο προσδιορισμός των περιπτώσεων χρήσης μπορεί να γίνει με διάφορους τρόπους:

- Προσδιορισμός των χρηστών και των επιχειρηματικών διεργασιών που θα συμμετέχουν.
- Προσδιορισμός των εξωτερικών συμβάντων που το σύστημα θα πρέπει να αποκριθεί.
- Συνδιαμός των παραπάνω συμβάντων με συγεκριμένους χρήστες και ειδικές πειρπτώσεις χρήσης.
- Έκφραση των επιχειρηματικών διεργασιών με συγκεκριμένα σενάρια και γενίκευση των σεναρίων σε περιπτώσεις χρήσης.
- Εντοπισμός των χρηστών που εμπλέκονται σε κάθε περίπτωση χρήσης.
- Άντληση πιθανών περιπτώσεων χρήσης από τις υφιστάμενες λειτουργικές απαιτήσεις. Αν μερικές απαιτήσεις δεν εκφράζονται σε καμμία περίπτωση χρήσης, τότε θα πρέπει να εξεταστεί το ενδεχόμενο να μην χρειάζονται πραγματικά.

Οι χρήστες συνήθως προτείνουν τις πιο σημαντικές περιπτώσεις χρήσης, οπότε η σειρά που έχουν καταγραφεί μας δείχνουν και ενδείξεις προτεραιότητας. Μία άλλη προσέγγιση ιεράρχησης είναι να γραφτεί μία σύντομη περιγραφή των περιπτώσεων χρήσης. Στην συνέχεια θα πρέπει να γίνει κατανομή των περιπτώσεων σε διαφορετικές εκδόσεις του προϊόντος. Θα πρέπει να δίνεται προτεραιότητα σε αυτές που εμφανίζονται περισσότερες φορές. Αυτές είναι και εκείνες που η εφαρμογή τους θα πρέπει να αρχίσει το συντομότερο δυνατόν [11,20,21].

**Δημιουργία Αναφοράς Περιπτώσεων Χρήσης**
Σε αυτό το στάδιο θα πρέπει να δίνεται βάση στις κύριες περιπτώσεις χρήσης. Θα πρέπει να δίνεται έμφαση στον στόχο που θέλει ο χρήστης να εκπληρώσει και στις ευθύνες του συστήματος για την επίτευξή του. Στη συνέχεια θα πρέπει να διαχωρίζονται τα όρια των περιπτώσεων χρήσης και να ορίζονται οι προϋποθέσεις που καθιστούν δυαντή την μετάβαση σε επόμενες περιπτώσεις. Θα πρέπει να αριθμηθούν όλες οι περιπτώσεις χρήσης, ανάλογα με την σειρά που χρησιμοποιούνται, για να προκύψει η σωστή ακολουθία των διεργασιών.

Το Σχήμα 31 δείνει την ακολουθία των γεγονότων για την ανάπτυξη των περιπτώσεων χρήσης. Μετά ο αναλυτής θα πρέπει να γράψει μία λεπτομερή αναφορά της κάθε περίπτωσης, όπως παρουσιάζεται στο δεύτερο σχήμα. Υπάρχουν δύο τρόποι να αντιπροσωπευθούν οι σχέσεις μεταξύ του συστήματος και των χρηστών. Στην συνέχεια, το Σχήμα 32 δείχνει μία λίστα με τα αριθμημένα βήματα, οπως αυτά εκτελούνται. Ο ίδιος συμβολισμός χρησιμοποιείται για την αναφορά των εναλλακτικών δρόμων και των εξαιρέσων που πρέπει να ληφθούν. Μία άλλη τεχνική είναι να παρουσιαστούν σε έναν πίνακα με δύο στήλες, όπως φαίνεται στον Πίνακα 4. Οι δράσεις των χρηστών εμφανίζονται στην αριστερή και αυτές του συστήματος στην δεξία στήλη, ενώ οι αριθμοί δείχνουν την ακολουθία των βημάτων.




**Σχήμα 31.** Προσέγγιση εκμαίευσης περιπτώσεων χρήσης

| | | | |
|---|---|---|---|
| Use Case ID | **UC-1** | Use Case Name | **Request a Chemical** |
| Created By | Tim | Last Updated By | Janice |
| Date Created | 12/4/02 | Date Last Updated | 12/27/02 |
| Actors | Requester | | |
| Description | The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system satisfies the request either by offering the Requester a new or used container of the chemical from the chemical stockroom or by letting the Requester create a request to order from an outside vendor. | | |
| Preconditons | 1. User's identity has been authenticated.<br>2. User is authorized to request chemicals.<br>3. Chemical inventory database is online. | | |
| Postconditions | 1. Request is stored in the Chemical Tracking System.<br>2. Request was e-mailed to the chemical stockroom or to a Buyer. | | |
| Normal Course | **1.0 Request a Chemical from the Chemical Stockroom**<br>1. Requester specifies the desired chemical.<br>2. System verifies that the chemical is valid.<br>3. System lists containers of the desired chemical that are in the chemical stockroom.<br>4. Requester has the option to View Container History for any container.<br>5. Requester selects a specific container or asks to place a vendor order (alternative course 1.1).<br>6. Requester enters other information to complete the request.<br>7. System stores request and e-mails it to chemical stockroom. | | |
| Alternative Courses | **1.1 Request a Chemical from a Vendor (branch after step 5)**<br>1. Requester searches vendor catalogs for a chemical.<br>2. System displays a list of vendors with available container sizes, grades, and prices.<br>3. Requester selects a vendor, container size, grade, and number of containers.<br>4. Requester enters other information to complete the request.<br>5. System stores request and e-mails it to Buyer. | | |

**Σχήμα 32.** Μερική περιγραφή περίπτωσης χρήσης

| Actor Actions | System Responses |
|---|---|
| 1. Specify the desired chemical. | 2. Verify that the chemical requested is valid.<br>3. Display a list of containers of the desired chemical that are in the chemical stockroom's current inventory. |
| 4. If desired, ask to view the history of any container.<br>5. Select a specific container (done) or ask to place a vendor order (alternative course 1.1). | |

**Πίνακας 4.** Περιγραφή περίπτωσης χρήσης σε δύο στήλες

Υπάρχουν κάποιες περιπτώσεις χρήσης που περιλαμβλανουν κάποιες πρόσθετες πληροφορίες που δεν ταιριάζουν σε καμμία από τις πρότυπες ενότητες. Θα πρέπει να καταγραφούν σαν "Special Requirements" ("Ειδικές Απαιτήσεις"). Αυτές είναι συνήθως απαιτήσεις που έχουν να

κάνουν με την ποιότητα και την επίδοση του συστήματος. Δεν χρεάζεται πάντα οι περιπτώσεις χρήσης να είναι τόσο καλα τεκμηριωμένες. Αλλά η σωστή τεκμηρίωση θα μας βοηθήσει όταν:

- Οι εκπρόσωποι των χρηστών δεν συνεργάζονται στενά με την ομάδα ανάπτυξης καθόλη την διάρκεια του έργου.
- Όταν η εφαρμογή είναι σύνθετη και ο κίνδυνος αποτυχίας είναι μεγάλος.
- Όταν πρόκειται να αναπτυχθεί μία ολοκληρωμένη δοκιμή βασισμένη στις απαιτήσεις των χρηστών.

***Περιπτώσεις Χρήσης Και Λειτουργικές Απαιτήσεις***

Η ομάδα ανάπτυξης δεν υλοποιεί επιχειρηματικές απαιτήσεις ή περιπτώσεις χρήσης. Υλοποιεί τις λειτουργικές απαιτήσεις, συγκεκριμένα κομμάτια δηλαδή του συστήματος που θα επιτρέπουν τους χρήστες να επιτύχουν τους στόχους τους. Η κάθε περίπτωση χρήσης περιγράφει την συμπεριφορά του συστήματος από την οπτική γωνία ενός χρήστη. Θα χρειαστούν λοιπόν αρκετές περιπτώσεις χρήσης για να γίνει ένας σωστός σχεδιασμός.

Δεν πρέπει να θεωρούμε τις περιπτώσεις χρήσης σαν λειτουργικές απαιτήσεις. Οι περιπτώσεις χρήσης περιγράφουν την σκοπιά του χρήστη και την συμπεριφορά του συστήματος. Δεν περιέχουν όλες τις πληροφορίες που χρειάζεται η ομάδα ανάπτυξης για να γράψει το απαραίτητο λογισμικό. Για να γίνει αυτό θα πρέπει ο αναλυτής να ορίσει με κάθε λεπτομέρεια τις λειτουργικές απαιτήσεις που προκύπτουν από κάθε περίπτωση χρήσης.

Ένας τρόπος να γίνει αυτό είναι να συμπεριληφθούν οι λειτουργικές απαιτήσεις σε κάθε περίπτωση χρήσης. Θα πρέπει όμως να καταγραφούν και μη λειτουργικές απαιτήσεις που σχετίζονται με κάθε περίπτωση.

Ένας άλλος τρόπος είναι να γίνει μία καλή περιγραφή της κάθε περίπτωσης χρήσης και να τεκμηριωθούν οι λειτουργικές απαιτήσεις σε ένα Έγγραφο Προδιαγραφής Απαιτήσεων. Σε αυτή την περίπτωση θα πρέπει να δημιουργηθούν διασυνδέσεις ιχνηλασιμότητας μεταξύ των περιπτώσεων χρήσης και των λειτουργικών απαιτήσεων. Ο καλύτερος τρόπος για να γίνει αυτό είναι να αποθηκευτούν τόσο οι περιπτώσεις χρήσης, όσο και οι λειτουργικές απαιτήσεις σε ένα εργαλείο διαχείρησης απαιτήσεων.

Μία τρίτη προσέγγιση είναι να οργανωθεί σωστά η SRS και να αποθηκευτούν εκεί όλες οι περιπτώσεις χρήσης και όλες οι λειτουργικές απαιτήσεις που προκύπτουν. Η περίπτωση αυτή δεν απαιτεί την δημιουργία ενός ξεχωριστού κειμένου για τις περιπτώσεις χρήσης. Θα πρέπει να οριστούν όλες οι διπλές λειτουργικές απαιτήσεις και κάθε φορά που θα εμφανίζονται θα γίνεται αναφορά στην συγκεκριμένη περίπτωσης χρήσης [21,32,33,35].

**Πλεονεκτήματα Περιπτώσεων Χρήσης**

Η προσέγγιση με περίπτωση χρήσης επικεντρώνεται στις διεργασίες και στους χρήστες και αυτό είναι και το βασικό της πλεονέκτημα. Οι χρήστες έχουν μία σαφέστερη προσδοκία για το νέο σύστημα από από ότι θα είχαν αν η προσέγγιση ήταν βασισμένη στις λειτουργίες. Βοηθάει τους

αναλυτές και την ομάδα ανάπτυξης να κατανοήσουν καλύτερα τις επιχειρηματικές διεργασίες και το πεδίο εφαρμογής του έργου. Μία προσεκτική παρακολούθηση των αλληλεπιδράσεων μπορεί να αποκαλύψει νωρίτερα τις τυχόν ασάφειες και αοριστίες στο έργο.

Ένα μεγάλο πρόβλημα στα έργα λογισμικού είναι η δημιουργία λειτουργιών που στην ουσία δεν χρειάζονται. Αυτό σπαταλάει πολύτιμους πόρους από το έργο χωρίς να υπάρχει λόγος. Με την προσέγγιση αυτή δεν υπάρχουν τέτοια ζητήματα, αφού ορίζονται με σαφήνεια οι λειτουργίες που πρέπει να γίνουν και που θα επιτρέψουν στους χρήστες να ολοκληρώσουν τις διεργασίες τους.

Επίσης βοηθά στην ιεράρχηση των απαιτήσεων. Οι περιπτώσεις χρήσης με υψηλή προτεραιότητα έχουν και λειτουργικές απαιτήσεις αντίστοιχη προτεραιότητα. Οι λόγοι που κάνουν μία περίπτωση χρήσης να έχει υψηλή προτεραιότητα είναι οι εξής:

- Περιγράφουν βασικές επιχειρηματικές διεργασίες που πρέπει να κάνει το σύστημα.
- Θα χρησιμοποιείται από πολλούς χρήστες.
- Παρέχει μία δυνατότητα που είναι αναγκαία για την συμμόρφωση του συστήματος με τους κανονισμούς.
- Από την παρουσία της εξαρτώνται και άλλες λειτουγίες του συστήματος.

Τέλος υπάρχουν και τεχνικά πλεονεκτήματα. Οι προγραμματιστές που χρησιμοποιούν αντικειμενοστρεφή μέθοδο σχεδιασμού μπορούν να μετατρέψουν τις περιπτώσεις χρήσης σε μοντέλα αντικειμένων όπως διαγράμματα κλάσεων και ακολουθίας. Επίσης θα είναι ευκολότερη η ανίχνευση λειτουργικών απαιτήσεων που θα πρέπει να αλλάξουν σε μία υποτιθέμενη αλλαγή των επιχειρηματικών διεργασιών, που μπορεί να συμβεί με την πάροδο του χρόνου.

Όπως και σε κάθε τεχνική τεχνολογίας λογσιμικού, έτσι και με την προσέγγιση αυτή μπορούμε εύκολα να παραστρατήσουμε αν δεν προσέξουμε τα ακόλουθα:

**Να υπάρχουν πάρα πολλές περιπτώσεις χρήσης.** Δεν πρέπει να δημιουργείται μία ξεχωριστή περίπτωση χρήσης για κάθε πιθανό σενάριο. Συνήθως οι περιπτώσεις χρήσης θα είναι περσσότερες από τις επιχειρηματικές διεργασίες, αλλά οι λειτουργικές απαιτήσεις θα είναι πολύ περισσότερες από τις πρειπτώσεις χρήσης.

**Δεν θα πρέπει να υπάρχουν ιδιαίτερα πολύπλοκες περιπτώσεις χρήσης.** Δεν μπορούμε να ελέγξουμε την πολυπλοκότητα των διεργασιών, αλλά μπορούμε να ελέγξουμε την πολυπλοκότητα των περιπτώσεων χρήσης. Ναι μεν να υπάρχουν πολλοί δυαντοί εναλλακτικοί τρόποι, αλλά ο καθένας από αυτούς θα πρέπει να είναι σύντομος και εύκολος να κατανοηθεί. Οι περιπτώσεις χρήσης θα πρέπει να περιέχουν ουσιώδεις πληροφορίες για τις διεργασίες του χρήστη και την συμπεριφορά του συστήματος και να μην περιέχουν την κάθε αλληλεπίδραση με κάθε λεπτομέρεια.

**Να συμπεριλαμβάνουν σχεδιασμό της διεπαφής του χρήστη.** Οι περιπτώσεις χρήσης θα πρέπει να επικεντρωθούν σε αυτό που θέλει να επιτύχει ο χρήστης και όχι στην οθόνη που θα δει. Θα πρέπει περιλαμβάνουν τις εννοιολογικές αλληλεπιδράσεις των στοιχείων, αλλά όχι και στοιχεία σχεδιασμού.

**Να μην περιλαμβάνουν ορισμούς δεδομένων.** Με την προσέγγιση αυτή είναι πολύ δύσκολο να βρουν οι συμμετέχοντες τον ορισμό που χρειάζονται γιατί δεν γίνεται εύκολα αντιληπτό ποιά περίπτωση χρήσης περιέχει τον κάθε ορσιμό. Οι ορισμοί θα πρέπει να συλλέγονται σε ένα ξεχωριστό λεξικό.

**Να μην χρησιμοποιούμε περιπτώσεις χρήσης που δεν κατλαβαίνουν οι χρήστες**. Αν οι χρήστες δεν κατανοούν την περιγραφή της περίπτωσης χρήσης ως προς τις επιχειρηματικές τους διεργασίες, τότε υπάρχει πρόβλημα με την συγκεκριμένη περίπτωση. Οι περιπτώσεις χρήσης θα πρέπει να γράφονται από την σκοπιά του χρήστη και όχι από εκείνη του αναλυτή.

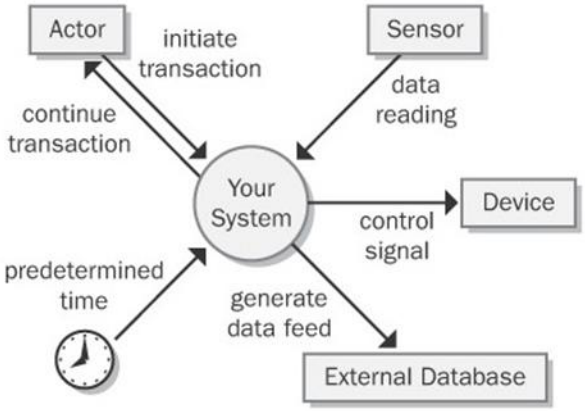
**Να μην δημιουργούμε νέες επιχειρηματικές διεργασίες**. Οι χρήστες δεν θα κατανοήσουν μία περίπτωση χρήσης που αφορά ένα κομμάτι λογισμικού που ακόμα δεν έχει υλοποιηθεί.

**Να μην υπάρχει εκτεταμένη χρήση διευρυμένων σχέσεων.** Η πρώτη επαφή με τις περιπτώσεις χρήσης αλλάζει τον τρόπο που οι αναλυτές, η ομάδα ανάπτυξης και οι χρήστες σκέφτονατι τις απαιτήσεις. Οι περιπτώσεις χρήσης που περιλαμβάνουν διευρημένες σχέσεις αλληλεπιδράσεων μπορεί να προκαλέσουν σύχγυση. Είναι καλύτερο να αποφεύγονται, τουλάχστον στα πρώτα στάδια ενασχόλησης με τις περιπτώσεις χρήσης.

***Event-Response Tables***

Ένας άλλος τρόπος να οργανωθούν και τεκμηριωθούν οι απαιτήσεις των χρηστών είναι να οριστούν τα εξωτερικά γεγονότα - συμβάντα στα οποία θα πρέπει να ανταποκριθεί το σύστημα. Σαν γεγονός (event) ορίζονται οι δραστηριότητεςπου γίνονται στο περιβάλλον του χρήστη και προκαλούν κάποια αντίδραση στο σύστημα λογισμικού. Ένας πίνακας Event – Response (Δράσης – Αντίδρασης) καταγράφει αυτά τα γεγονότα αλλά και την επιθυμητή απόκριση που πρέπει να έχει το σύστημα σε αυτά. Υπάρχουν διάφοροι τύποι απόκρισης του συστήματος, όπως φαίνεται και παρακάτω:

- Η ενέργεια ενός ανθρώπινου χρήστη που "ανοίγει έναν διάλογο" με το λογισμικό, όπως όταν ένας χρήστης ξεκινά μία περίπτωση χρήσης, ονομάζεται Business Event. Η ακολουθία Event–Response αντιστοιχεί στα βήματα μίας περίπτωσης χρήσης. Σε αντίθεση όμως με τις περιπτώσεις χρήσης, ο Event – Response πίνακας δεν περιγράφει τον στόχο του χρήστη που χρησιμοποιεί το σύστημα και ούτε εξηγεί πώς αυτή η διαδικασία δίνει αξία στον χρήστη.
- Ένα σήμα ελέγχου, ανάγνωση δεδομένων ή μία διακοπή λήψης από μία εξωτερική συσκευή όπως όταν κλείνει κάποιος διακόπτης, αλλάζει η τάση ή μετακίνηση του σπόνσορα ποντικιού από τον χρήστη.
- Μία ενεργοποίηση ενός συμβάντος όπως όταν ρολόι φθάσει μία προκαθορισμένη θέση.




**Σχήμα 33.**Παραδείγματα εξωτερικών γεγονότων και απόκρισης

Οι πίνακες Event – Response είναι κατάλληλοι για συστήματα ελέγχου πραγματικού χρόνου. Η απόκριση του συστήματος δεν εξαρτάται μόνο από το σήμα που θα λάβει αλλά και από την χρονική στιγμή που θα το λάβει.

Αυτός ο πίνακας καταγράφει πληροφορίες στο επίπεδο των απαιτήσεων των χρηστών. Εάν ο πίνακας ορίζει και χαρακτηρίζει κάθε πιθανό συνδιασμό γεγονότος, κατάστασης και απόκρισης τότε θα μπορεί να λειτουργήσει σαν μέρος των λειτουργικών απαιτήσεων για το συγκεκριμένο κομμάτι του συστήματος. Ωστόσο ο αναλυτής θα πρέπει να συμπληρώσει την SRS με επιπρόσθετες λειτουργικές και μη λειτουργικές απαιτήσεις. Στον πίνακα Event – Response καλό θα είναι να γράφονται και οι περιορισμοί που υπάρχουν για να διευκολυνθεί η σχεδίαση.

***Επιχειρηματικοί Κανόνες και Περιορισμοί***

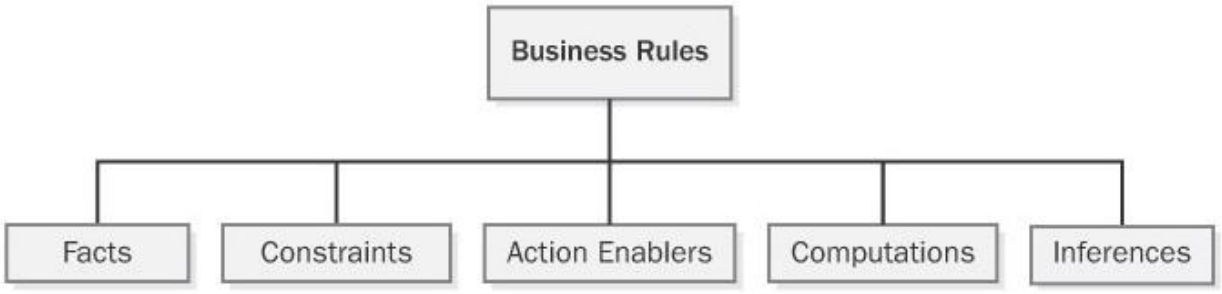
Κάθε επιχειρηματικός οργανισμός λειτουργεί σύμφωνα με ένα σύνολο εταιρικών πολιτικών, κανόνων και βιομηχανικών standards. Αυτές οι αρχές ονομάζονται επχειρηματικοί κανόνες. Αποτελούν μία μεγάλη πηγή των λειτουργικών απαιτήσεων ενός λογισμικού αφού υπαγορεύουν στο σύστημα τι πρέπει να διεθέτει για να συμμορφώνεται με τους κανόνες. Μπορούν ακόμα να κατευθύνουν και τις υψηλού επιπέδου επιχειρηματικές απαιτήσεις. Εάν είμαστε σε θέση να ξέρουμε πού και πώς εφαρμόζεται η κάθε απαιτήση σύμφωνα με τους επιχειρηματικούς κανόνες, τότε είμαστε σε θέση να αλλάξουμε εύκολα τις απαιτήσεις σε μία υποτιθέμενη αλλαγή των κανόνων.

Η συλλογή των επιχειρηματικών κανόνων βοηθά τον αναλυτή να προσδιορίσει τις πιθανές πηγές των απαιτήσεων που θα πρέπει να εφαρμοστούν στο σύστημα και τις οποίες οι χρήστες δεν είναι πάντα σε θέση να γνωρίζουν. Έχοντας μία σαφή εικόνα των επιχειρηματικών κανόνων μπορούμε να εφαρμόσουμε τα επηρεαζόμενα από αυτές χαρακτηριστικά με πιο συνεπή τρόπο.

Σύμφωνα με το Business Rules Group: “ Ένας επιχειρηματικός κανόνας είναι μία δήλωση που ορίζει ή περιορίζει κάποια στοιχεία της επιχείρησης. Έχει σκοπό να επιβεβαιώσει την δομή της επιχείρησης και να ελέγξει ή να επηρεάσει την συμπεριφορά της”. Έχουν αναπτυχθεί ολόκληρες μεθοδολογίες για την ανακάλυψη και την τεκμηρίωση και εφαρμογή των επιχειρηματικών κανόνων. Αν δεν ασχολούμαστε με την δημιουργία ενός αμιγώς επιχειρηματικού προϊόντος, τότε

μπορούμε απλά να τεκμηριώσουμε τους κανόνες που αφορούν το σύστημά μας και να τους συνδέσουμε με συγκεκριμένες λειτουργικές απαιτήσεις.

Έχουν προταθεί διάφορες ταξινομήσεις των επιχειρηματικών κανόνων. Στο Σχήμα 34 φαίνεται ένα σχεδιάγραμμα με πέντε είδη επιχειρηματικών κανόνων. Θα είναι επίσης καλό να δημιουργηθεί και ένα λεξικό για να περιληφθούν οι σημαντικοί όροι για την επιχείρηση και οι επιχειρηματικοί κανόνες με συνεπή τρόπο έτσι ώστε να προσθέσουν αξία στο λογισμικό που πρόκειται να υλοποιηθεί. Είναι ιδιαίτερα σημαντικό να ταξινομηθούν σωστά οι επιχειρηματικοί κανόνες έτσι ώστε να γίνει σωστή εκμαίευση των αντίστοιχων λειτουργικών απαιτήσεων.




**Σχήμα 34.** Μία απλή ταξινόμηση των επιχειρηματικών κανόνων

**Γεγονότα (Facts).** Τα γεγονότα είναι δηλώσεις σχετικές με την επιχείρηση. Περιγράφουν τις σχέσεις μεταξύ των σημαντικών επιχειρηματικών όρων. Παραδείγματα γεγονότων είναι:

- "*Κάθε παραγγελία πρέπει να έχει επιβάρυνση με έξοδα αποστολής"*.
- *"Ο φόρος θα πρέπει να υπολογίζεται στα έξοσα αποστολής"*.
- *"Κάθε αποστολή θα πρέπει να έχει ένα μοναδικό αναγνωριστικό χαρακτηριστικό*".

**Περιορισμοί (Constraints).** Οι περιορισμοί περιορίζουν τις διεργασίες που το σύστημα και οι χρήστες μπορούν αν εκτελέσουν. Μερικές λέξεις ή φράσεις που υποδηλώνουν περιορισμό είναι: *πρέπει, πρέπει να, δεν πρέπει, μόνο*. Τα έργα λογισμικού έχουν πολλά είδη περιορισμών. Οι Project Managers θα πρέπει να συμβιβαστούν με το χρονοδιάγραμμα, το ανθρώπινο προσωπικό και τους οικονομικούς πόρους που έχουν οριστεί. Αυτού του επιπέδου οι περιορισμοί ανήκουν στο πλάνο διαχείρησης του έργου. Οι περιορισμοί σε πίπεδο σχεδιασμού και εφαρμογής που στην που μειώνουν τις διαθέσιμες επιλογές ανήκουν στο Έγγραφο Προσδιορισμού Απαιτήσεων ή στις προδιαγραφές σχεδιασμού. Οι επιχειρηματικοί κανόνες που επιβάλουν περιορισμούς σχετικά με τον τρόπο λειτουργίας της επιχείρησης αντανακλόνται στις λειτουργικές απαιτήσεις του λογισμικού.

**Action Enablers.** Ένας κανόνας που προκαλεί κάποια δραστηριότητα υπό ορισμένες συνθήκες ονομάζεται actionenabler. Είναι περιορισμοί του τύπου:

- *"Αν η παραγγελία υπάρχει στην αποθήκη τότε να δρομολογηθεί η αποστολή"*.

- *"Αν ένας πελάτης έχει κάνει μία συγκεκριμένη παραγγελία να του εμφανιστούν και άλλα προτεινόμενα προϊόντα πριν πεικυρωθεί η παραγγελεία".*

**Inferences.** Αυτός ο κανόνας βασίζεται σε προϋποθέσεις. Το αποτέλεσμα δηλασή δεν είναι προκαθορισμένο αλλά εξαρτάται. Είναι του τύπου: *" If , Then", "Αν συμβεί δηλαδή το Α τότε να συμβεί το Β, αλλιώς να μην γίνει τίποτα"*.

**Υπολογισμοί (Computations).** Είναι μία κατηγορία επιχειρηματικών κανόνων που καθορίζεται από μαθηματικούς τύπους και αλγόριθμους. Έχουν συνήθως να κάνουν με οικονομικό υπόβαθρο. Αν δηλαδή ένας πελάτης κάνει μία παραγγελεία μεγαλύτερη από ένα δεδομένο ποσό να έχει μία καθορισμένη έπτωση.

| ID | Αριθμός προϊόντων | % Έκπτωση |
|---|---|---|
| **DISC-1** | Από 1 έως 5 | 0 |
| **DISC-2** | Από 6 έως 10 | 10 |
| **DISC-3** | Από 11 έως 20 | 25 |
| **DISC-3** | Από 20 και πάνω | 40 |

**Πίνακας 5.** Πίνακας αναπαράστασης υπολογιστικών επιχειρηματικών κανόνων

**Καταγραφή Επιχειρηματικών Κανόνων**

Οι επιχειρηματικοί κανόνες μπορούν να επηρεάσουν πολλές εφαρμογές του λογισμικού. Θα είναι καλό λοιπόν να υπάρχει ένας κατάλογος που να συγκεντρώνσει αυτούς τους κανόνες για να εξασγαλιστεί ότι δεν παραλειφθεί κανένας. Ωστόσο οι μεγάλοι οργανισμοί των οποίων η δραστηριότητα επηρεάζεται πολύ από αυτούς τους κανόνες θα πρέπει να τους συγκεντρώνουν σε μία βάση δεδομένων.

Όσο αποκτούμε εμπειρία στην συλλογή και την τεκμηρίωση των επιχειρηματικών κανόνων θα μπορούμε να τους συγκεντρώνουμε σε καθορισμένα πρότυπα. Τα πρότυπα αυτά διευκολύνουν την αποθήκευση των κανόνων σε μία βάση δεδομένων ή σε ένα εργαλείο διαχείρησης.

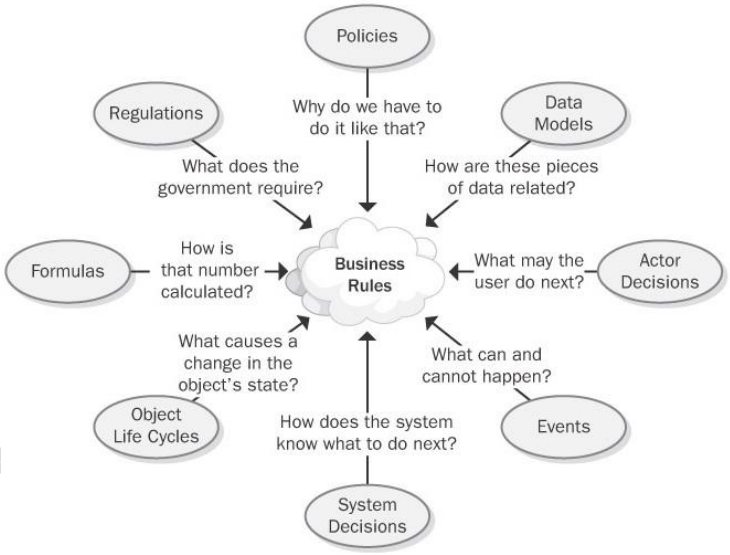
Όπως φαίνεται και στον Πίνακα 6, στον κάθε επιχειρηματικό κανόνα δίνεται ένα μοναδικό αναγνωριστικό χαρακτηριστικό που μας επιτρέπει να ανιχνεύσουμε τις λειτουργικές απαιτήσεις που κρύβονται πίσω από αυτούς. Η στήλη "Τύπος Κανόνα" προσδιορίζει τον κανόνα σαν ένα γεγονός, έναν περιορισμό, ένα Action enabler, ένα Inference ή έναν υπολογισμό. Η στήλη "Στατικός ή Δυναμικός" μας δείχνει την πιθανότητα να αλλάξει ένας κανόνας με την πάροδο του χρόνου. Οι πηγές των επιχειρηματικών κανόνων είναι η πολιτική της επιχείρησης, κυβερνητικοί κανονισμοί και περιορισμοί που πρέπει να εφαρμοσούν ώστε να εξασφαλιστεί η συμβατότητα με κάποιο ήδη υπάρχον λογισμικό.

| ID | Ορισμός κανόνα | Τύπος κανόνα | Στατικός ή Δυναμικός | Πηγή |
|---|---|---|---|---|
| **ORDER - 15** | Ο χρήστης μπορεί να παραγγείλει το προϊόν εάν έχει μία εξάμηνη εμπειρία στην χρησιμοποιήση του. | Περιορισμός | Στατικός | Επιχειρηματικός Κανόνας |
| **DISC - 5** | Γίνεται κάποια έκπτωση ανάλογα με το μέγεθος της παραγγελείας. | Υπολογισμός | Δυναμικός | Επιχειρηματικός Κανόνας |

**Πίνακας 6.** Δείγμα καταλόγου επιχειρηματικών κανόνων

**Οι Επιχειρηματικοί Κανόνες Στην Εκμαίευση Απαιτήσεων**

Οι ενδιαφερόμενοι του έργου συνήθως γνωρίζουν τους επιχειρηματικούς κανόνες που θα πρέπει να συμπεριληφθούν και κατά τις καταθέτουν κατά την συνέντευξη με τους αναλυτές. Το Σχήμα 35 δείνει την προέλευση αυτών των κανόνων. Ο αναλυτής στην συνέχεια θα πρέπει να ρωτήσει τα κατάλληλα άτομα για την ορθότητα αυτών των κανόνων και στην συνέχεια τους τεκμηριώσει.




**Σχήμα 35.** Εκμαίευση επιχειρηματικών κανόνων με ερωτήσεις σε διάφορους ενδιαφερόμενους

Μετά τον εντοπισμό και την τεκμηρίωση των επιχειρηματικών κανόνων θα πρέπει να οριστεί ποιοί από αυτούς θα πρέπει να συμπεριληφθούν στο λογισμικό. Στην συνέχεια θα πρέπει οριστούν οι διασυνδέσεις μεταξύ αυτών των κανονισμών με τις λειτουργικές απαιτήσεις που θα προκύψουν. Για την αποφυγή διπλοκαταχωρήσεων δεν θα πρέπει να μεταφέρονται οι επιχειρηματικοί κανόνες στοέγγραφο προδιαγραφών λογισμικού, αλλά θα πρέπει να υπάρχουν

παραπομπές σε συγκεκριμένους κανόνες και αλγόριθμους. Ο προγραμματιστής που θα διαβάζει το έγγραφο προδιαγραφών λογισμικού θα πρέπει να ακολουθεί την παραπομπή του συνδέσμου για να έχει στις λεπτομέρειες του κανόνα. [14,15,21].

## 3.2 Ανάλυση Απαιτήσεων

Εύκολα μπορούμε να καταλάβουμε πως μία μονόπλευρη απεικόνιση δεν παρέχει πλήρη κατανόηση. Για να έχουμε μία πλήρη εικόνα των απαιτήσεων ενός συστήματος χρειάζεται ένας συνδιασμός κειμένου και οπτικών αναπαραστάσεων που περιλαμβάνουν λίστες των λειτουργικών απαιτήσεων, γραφικά μοντέλα ανάλυσης, δέντρα αποφάσεων και περιπτώσεις ελέγχου. Στην καλύτερη περίπτωση θα έπρεπε οι αναπαράστασεις να γίνονται από διαφορετικούς ενδιαφερόμενους, έτσι ώστε να πάρουν όλοι μέρος στην ανάλυση των απαιτήσεων. Η σύγκριση των διαφορετικών αναπαραστασέων θα αποκλύψει αντιφάσεις, ασάφειες και παραλείψεις που είναι δύσκολο να εντοπιστούν από μία ενιαία άποψη.

Τα διαγράμματα και οι εικόνες βοηθούν να γεφυρωθεί το χάσμα μεταξύ του προϊόντος που τελικά θα παραδοθεί σε σχέση με τις προσδοκίες που έχουν οι χρήστες για αυτό. Οι αναλυτές θα πρέπει να εξηγήσουν στους υπόλοιπους ενδιαφερόμενους των σκοπό των μοντέλων που θα χρησιμοποιηθούν.

### 3.2.1 Μοντελοποίηση Των Απαιτήσεων

Ο σκοπός της διαδικασίας της μοντελοποίησης είναι να δημιουργηθεί μία σχηματική αναπαράσταση των λειτουργιών που θα εκτελούνται τελικά από το αναπτυσσόμενο σύστημα.Το κλειδί για την επιτυχία της διαδικασίας είναι να βρεθεί ένας τρόπος να περιγράψουν οι πελάτες όχι το τι κάνουν και πώς το κάνουν, αλλά το τι χρειάζονται και πώς θα μπορούσε αυτό καλύτερα να επιτευχθεί, εάν δεν ήταν αναγκασμένοι να συμβιβάζονται με τους περιορισμούς του παρόντος συστήματος που χρησιμοποιούν. Ένα από τα συνηθισμένα λάθη που γίνονται κατά τη διάρκεια αυτής της φάσης του κύκλου ζωής της ανάπτυξης είναι να αυτοματοποιηθεί μία ¨κακή¨ διαδικασία, μόνο και μόνο επειδή αυτός είναι ο τρόπος που πάντα λειτουργούσε η επιχειρήση. Θα πρέπει να διασφαλιστεί ότι το λογικό μοντέλο πληροφοριών δεν εξαρτάται από το πώς επεξεργάζονται τα δεδομένα από το σύστημα, αλλά το πώς οργανόνωνται με βάση τις ανάγκες του συστήματος. Το όφελος από αυτόν τρόπο κατηγοριοποίησης είναι πως το μοντέλο των δεδομένων θα παραμείνει ανέπαφο ακόμα και αν μελλοντικά αλλάξει η λειτουργικότητα του συστήματος.

Δεν υπάρχει κάποια τεχνική που να παρέχει μία ολιστική απεικόνιση των απαιτήσεων ενός συστήματος. Αυτό μπορεί να επιτευχθεί με έναν συνδιασμό μοντέλων οπτικής απεικίνισης των απαιτήσεων. Τέτοια μοντέλα είναι τα Διαγράμματα Ροής Δεδομένων (ΔΡΔ - Data Flow Diagrams DFD), τα Διαγράμματα Οντοτήτων Συσχετίσεων (Entity-Relationship Diagrams ERD), τα State-Transition Diagrams (STD), τα Διαγράμματα Περιπτώσεων Χρήσης (Use Case

Diagrams), τα Διαγράμματα Κλάσεων (Class Diagrams) και τα Διαγράμματα Δραστηριοτήτων (Activity Diagrams).

Τα μοντέλα αυτά είναι χρήσιμα τόσο για την εκπόνηση και την διερεύνηση των απαιτήσεων, όσο και για τον σχεδιασμό του λογισμικού. Η χρησιμοποίησή τους για τον έναν ή τον άλλον σκοπό εξαρτάται από την κατάσταση του έργου την συγκεκριμένη χρονική στιγμή. Όταν χρησιμοποιούνται κατά την ανάλυση μας επιτρέπουν να μοντελοποιήσουμε το πρόβλημα και να αναπαραστήσουμε εννοιολογικά το νέο σύστημα. Απεικονίζουν τις λογικές πτυχές των συστατικών στοιχείων, τους μετασχηματισμούς και τις αλλαγές της κατάστασης του συστήματος. Όταν πάλι χρησιμοποιούνται κατά τον σχεδιασμό, υποδεικνύουν τις κλάσεις των αντικειμένων και τις ενότητες του κώδικα που θα αναπτυχθούν. Επειδή χρησιμοποιούν τον ίδιο συμβολισμό και στις δύο περιπτώσεις, θα πρέπει κάθε φορά να τα χαρακτηρίζουμε ως μοντέλα ανάλυσης ή σαν μοντέλα σχεδιασμού για να αποφύγουμε τυχόν συγχύσεις.

Οι τεχνικές μοντελοποίησης της ανάλυσης υποστηρίζονται από μία ποικιλία εμπορικών εφαρμογών τεχνολογίας λογισμικού, τα CASE Tools (εργαλεία CASE). Τα εργαλεία αυτά καθιστούν έυκολη την βελτίωση των διαγραμμάτων, εντοπίζουν συντακτικά λάθη και ανακολουθίες μπορεί να μην φαίνονται από μία απλή επανεξέταση των διαγραμμάτων και βοηθούν να διατηρηθεί η συνεκτικότητα μεταξύ των μοντέλων και των λειτουργικών απαιτήσεων του Εγγράφου Προδιαγραφής Απαιτήσεων.

Συνήθως οι ομάδες ανάπτυξης λογισμικού δεν καλούνται να μοντελοποιήσουν ένα ολόκληρο σύστημα. Αντιθέτως, εστιάζουν την μοντελοποίηση στα πιο πολύπλοκα και επικίνδυνα τμήματα του συστήματος και σε εκείνα που χαρακτηρίζονται από ασάφεια και αβεβαιότητα και σε τμήματα που τα ελαττώματα θα δημιουργήσουν μεγάλες αρνητικές επιπτώσεις στο έργο.

Για να γίνει κατανοητό το τι ακριβώς ζητάνε οι πελάτες ως απαιτήσεις θα πρέπει να ορίσουμε κάποιες λέξεις κλειδιά που θα μας βοηθήσουν. Ο Πίνακας 7 δείχνει κάποια κύρια ρήματα και ουσιαστικά που χρησιμοποιούν οι χρήστες και πώς αυτά πρέπει να μεταφράζονται. Οι αναλυτές θα πρέπει να είναι σε θέση να συνδέσουν το κάθε ένα από αυτά με μία συγκεκριμένη απαίτηση χρήστη [1,5].



| **Τύπος Λέξης** | **Παράδειγμα** | **Ανάλυση Λέξης – Συσχέτιση με Μοντέλα Ανάλυσης** |
|---|---|---|
| **Ουσιαστικό** | • Άνθρωπος.<br>• Οργανισμός.<br>• Σύστημα λογισμικού.<br>• Δεδομένα.<br>• Αντικείμενα.<br>• Αντικείμενα που | • Τερματικά ή Αποθήκευση Δεδομένων (DFD).<br>• Χρήστες (UseCaseDiagrams).<br>• Οντότητες και οι |

| | | |
|---|---|---|
| | υπάρχουν. | ιδιότητές τους (ERD).<br>• Κλάσεις και οι ιδιότητές τους (ClassDiagrams). |
| **Ρήμα** | • Διεργασίες.<br>• Πράγματα που μπορεί να κάνει ο χρήστης.<br>• Γεγονότα που μπορούν να γίνουν. | • Διεργασίες (DFD).<br>• Use Cases (Use Case Diagrams).<br>• Συσχετίσεις (ERD).<br>• Μεταβάσεις (STD).<br>• Δραστηριότητες (Διαγράμματα Δραστηριοτήτων). |

**Πίνακας 7.**Συσχέτιση της φωνής των πελατών με Μοντέλα Ανάλυσης

***Διάγραμμα Ροής Δεδομένων (DataFlowDiagram)***

Τα διαγράμματα ροής δεδομένων είναι το βασικό εργαλείο της δομημένης ανάλυσης. Προσδιορίζει τις διαδιακασίες μετασχηματισμού, τα δεδομένα που χειρίζεται το σύστημα και τις ροές δεδομένων μεταξύ των διεργασιών του συστήματος και τωνεξωτερικών παραγόντων.Αποσυνθέτει τα σύνθετα προβλήματα σε επίπδα λεπτομέρειας για να γίνουν κατανοητά από όλους. Είναι ιδιαίτερα χρήσιμα σε εφαρμογές με πολλές απαιτήσεις και στην μοντελοποίηση συστημάτων πραγματικού χρόνου.

Ένα διάγραμμα ροής δεδομένων (Data-Flow Diagram ή DFD) είναι μια γραφική αναπαράσταση της "ροής" των δεδομένων διαμέσου ενός πληροφοριακού συστήματος . Τα διαγράμματα αυτά μπορούν να χρησιμοποιηθούν επίσης για οπτικοποίηση της επεξεργασίας των δεδομένων (Δομημένη Σχεδίαση - Structured Design).

Σε ένα ΔΡΔ, τα δεδομένα ρέουν από μια εξωτερική πηγή δεδομένων προς έναν εσωτερικό αποθηκευτικό χώρο δεδομένων ή έναν εξωτερικό προορισμό δεδομένων, μέσω μιας εσωτερικής διεργασίας.

Ένα ΔΡΔ δεν παρέχει πληροφορίες για το χρονισμό των διεργασιών ή το αν αυτές λειτουργούν ακολουθιακά ή παράλληλα. Είναι επομένως διαφορετικό από ένα διάγραμμα ροής (flowchart), το οποίο δείχνει τη ροή του ελέγχου μέσα σε έναν αλγόριθμο και επιτρέπει στον αναγνώστη να βρίσκει ποιες λειτουργίες θα εκτελεστούν, με ποια σειρά και κάτω από ποιες συνθήκες, αλλά όχι τι είδους δεδομένα θα εισαχθούν και θα εξαχθούν από το σύστημα, ούτε από πού έρχονται τα δεδομένα και προς τα πού κατευθύνονται, ούτε πού αυτά αποθηκεύονται (όλες αυτές οι πληροφορίες εμφανίζονται σε ένα ΔΡΔ).

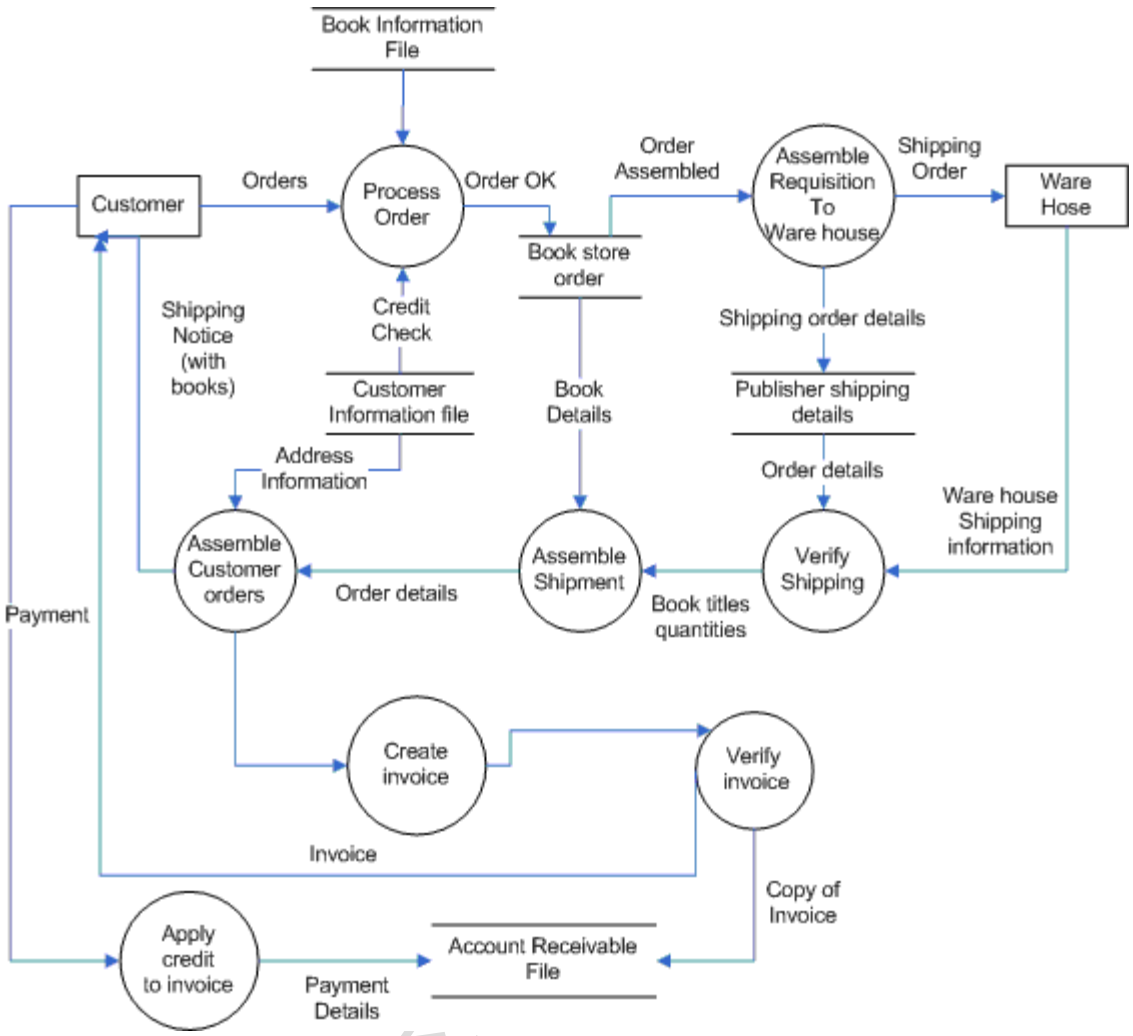
Αποτελεί κοινή πρακτική η δημιουργία αρχικά ενός διαγράμματος ροής δεδομένων contexts (System Context Diagram, Context-Level Data Flow Diagram), το οποίο να δείχνει την αλληλεπίδραση μεταξύ του συστήματος και των εξωτερικών agents, οι οποίοι λειτουργούν σαν

πηγές και προορισμοί δεδομένων. Σε αυτό το διάγραμμα, που είναι γνωστό και σαν "ΔΡΔ Επιπέδου 0", οι αλληλεπιδράσεις του συστήματος με τον εξωτερικό κόσμο μοντελοποιούνται αποκλειστικά με βάση τις ροές δεδομένων στο όριο του συστήματος. Το διάγραμμα δείχνει όλο το σύστημα σαν μια μοναδική διεργασία και δεν παρέχει πληροφορίες για την εσωτερική του οργάνωση.

Αυτό το ΔΡΔ σε επίπεδο Context ΔΡΔ στη συνέχεια "εκρήγνυται", για να παράγει ένα άλλο Επιπέδου 1, το οποίο δείχνει κάποιες από τις λεπτομέρειες του συστήματος που μοντελοποιείται. Το ΔΡΔ Επιπέδου 1 δείχνει πώς το σύστημα διαιρείται σε υποσυστήματα (διεργασίες), κάθε ένα από τα οποία χειρίζεται μια ή περισσότερες ροές δεδομένων από ή προς έναν εξωτερικό agent, και τα οποία μαζί παρέχουν τη συνολική λειτουργικότητα του συστήματος. Εμφανίζει επίσης τους εσωτερικούς αποθηκευτικούς χώρους δεδομένων, οι οποίοι πρέπει να υπάρχουν στο σύστημα για να λειτουργεί σωστά, και τη ροή των δεδομένων μεταξύ διάφορων τμημάτων του συστήματος.

Οι διεργασίες εμφανίζονται σαν φυσαλίδες και τα τερματικά του συστήματος εμφανίζονται με ορθογώνια σχήματα. Όλες οι ροές δεδομένων εμφανίζονται με βέλη. Οι αποθηκευτικοί χώροι είναι τα σχήματα που αποτελούνται από δύο παράλληλες γραμμές. Η ροή από μία φυσαλίδα σε έναν αποθηκευτικό χώρο υποδηλώνει πως γίνεται αποθήκευση κάποιας πληροφορίας. Η ροή από έναν αποθηκευτικό χώρο δείχνει πώς γίνεται ανάγνωση κάποιας πληροφορίας και τα αμφίδρομα βέλη δείχνουν πώς γίνεται ενημέρωση ή ανταλλαγή πληροφοριών.

Η κάθε διεργασία που εμφανίζεται σαν ξεχωριστή φυσαλίδα στο διάγραμμα επιπέδου 0 μπορεί να επεκταθεί περαιτέρω σε ένα ξεχωριστό Διάγραμμα Ροής Δεδομένων με περισσότερες λεπτομέρειες σχετικά με τις λειτουργίες του. Οι λειτουργικές απαιτήσεις του Εγγράφου Προσδιορισμού Απαιτήσεων καθορίζουν με ακρίβεια το τι συμβαίνει σε κάθε διεργασία και μας βοηθά να δημιουργήσουμε ένα ακριβές ΔΡΔ.




**Σχήμα 36.** Διάγραμμα Ροής Δεδομένων Μηδενικού επιπέδου

Δεν δημιουργούν όλοι τα ΔΡΔ με τον ίδιο τρόπο αλλά υπάρχουν ορισμένες συμβάσεις που καλό είναι να γίνονται για να ερμηνεύονται το ίδιο από όλους:

- Να τοποθετούμε αποθήκες δεδομένων μόνο στα ΔΡΔ μηδενικού επιπέδου και όχι στα Context Diagrams.
- Η επικοινωνία μεταξύ αποθηκευτικών χώρων να μην γίνεται απευθείας από τον έναν στον άλλον αλλά να περνούν μέσα από μία φυσαλίδα διεργασίας.
- Να ονομάζεται η καθε διεργασία για να διευκολύνεται η ερμηνεία του ΔΡΔ από τους χρήστες.
- Να αριθμούνται οι διεργασίες μοναδικά και ιεραρχικά. Στο μηδενικό επίπεδο η κάθε διεργασία να αριθμήται με ένα ακέραιο. Αν δημιουργήθει μετά μία ΔΡΔ για την διεργασία νούμερο 3, τότε η αρίθμηση των νέων διεργασιών να γίνεται ως 3.1, 3.2.
- Να προσέχουμε το κάθε διάγραμμα να μην έχει παραπάνω από 10 διεργασίες. Όσο πιο απλό είναι ένα ΔΡΔ τόσο πιο κατανοητό γίνεται.

- Όσες ροές των δεδομένων εισέρχονται σε μία διεργασία ή έναν αποθηκευτικό χώρο, τόσες θα πρέπει και να εξέρχονται.

Οι εκπρόσωποι των ενδιαφερομένων που θα εξετάσουν το ΔΡΔ θα πρέπει να βεβαιώσουν ότι δεν έχει παραλειφθεί καμμιά διεργασία και ότι δεν έχει προστεθεί κάποια που δεν έχει συμφωνηθεί. Συχνά τα Διαγράμματα Ροής Δεδομένων αποκαλύπτουν κατηγορίες χρηστών, επιχειρηματικές διεργασίες και συνδέσεις με άλλα συστήματα που πρηγουμένως δεν είχαν οριστεί.

Η χρησιμοποίηση των ΔΡΔ για την μοντελοποίηση των συστημάτων έχει πολλά πλεονεκτήματα. Ορισμένα από αυτά είναι:

- Η απλότητα που προσφέρουν αφού είναι ευανάγνωστα και εύκολα αντιληπτά.
- Εξασφαλίζουν μία δομημένη προσέγγιση στα αρχικά στάδια της εργασίας.
- Εξασφαλίζουν ένα πλαίσιο για την προδιαγραφή των απαιτήσεων ενός λογισμικού.
- Οδηγούν απευθείας στον προσδιορισμό των λειτουργιών του λογισμικού καθώς και στην περιγραφή όλων των εσωτερικών διεπαφών μεταξύ των λειτουργιών όπως επίσης και των εξωτερικών διεπαφών με άλλο λογισμικό.
- Εξασφαλίζουν την ιχνηλασιμότητα μεταξύ των απαιτήσεων και διευκολύνουν την μεθοδολογία σχεδίασης.

Υπάρχουν όμως και ορισμένα μειονεκτήματα:

- Δεν είναι κατάλληλα για την απεικόνιση της ροής ελέγχου.
- Δεν είναι κατάλληλα για πολύπλοκες ή παράλληλες διεργασίες.
- Δεν παρέχουν συμβολισμό για την μοντελοποίηση των δεδομένων.
- Λόγω της παλαιότητάς τους δεν είναι συμβατά με το αντικειμενοστρεφές υπόδειγμα ανάπτυξης, το οποίο είναι το πλέον κυρίαρχο.

***Entity-Relationship Diagram (Διάγραμμα Οντοτήτων Συσχετίσεων)***

Όπως το Διάγραμμα Ροής Δεδομένων δείχνει τις διεργασίες που συμβαίνουν σε ένα σύστημα, έτσι και το Διάγραμμα Οντοτήτων Συσχετίσεων δείχνει τις σχέσεις των δεδομένων του συστήματος. Μας βοηθά να κατανοήσουμε πώς οι διεργασίες επικοινωνούν μεταξύ τους.

Τα διαγράμματα οντοτήτων-σχέσεων, έχουν σαν βασικό στοιχείο τους την οντότητα. Η οντότητα, είναι μια αναπαράσταση κάποιας αυτόνομης ύπαρξης με υλική (στον πραγματικό κόσμο) ή θεωρητική υπόσταση (συμβατική ύπαρξη). Για παράδειγμα, οντότητα μπορεί να είναι ένας φοιτητής (ένας άνθρωπος με ονοματεπώνυμο, χαρακτηριστικά, κ.α) αλλά και ένα μάθημα σε μια σχολή (κάτι άυλο αλλά με συμβατική υπόσταση).

Τα δεδομένα τα οποία θα αποθηκευτούν σε κάποια οντότητα του μοντέλου οντοτήτων-σχέσεων, αντιστοιχούν στις εγγραφές στο φυσικό επίπεδο μίας Βάσης Δεδομένων. Έτσι η οντότητα, θα πρέπει να έχει κάποιο όνομα, και κάποια στοιχεία που να καθορίζουν τα χαρακτηριστικά της συγκεκριμένης οντότητας.

Οι οντότητες στα Διαγράμματα Οντοτήτων Συσχετίσεων εμφανίζονται με τετράγωνα. Κάθε οντότητα έχει διάφορα στοιχεία που την προσδιορίζουν. Ένα τέτοιο στοιχείο ονομάζεται ιδιότητα, χαρακτηριστικό ή πεδίο της οντότητας.Τα χαρακτηριστικά χωρίζονται σε:

- Μονότιμα (single valued), τα οποία έχουν μόνο μια τιμή.
- Πλειότιμα (multi-valued), τα οποία έχουν σύνολο από τιμές.

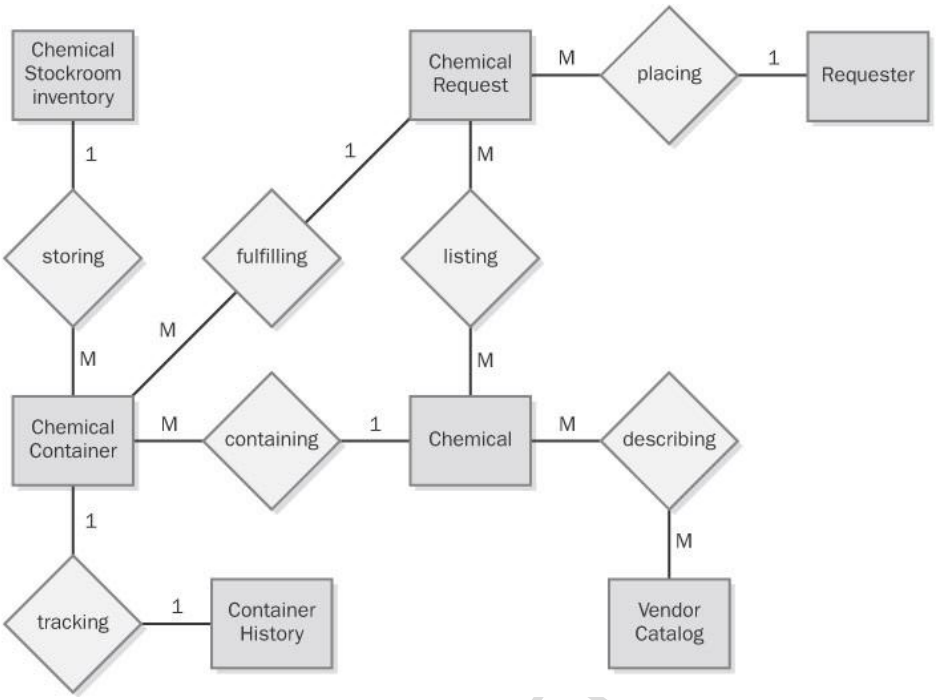
Στο Διάγραμμα Οντοτήτων Συσχετίσεων (Ο/Σ) οι ιδιότητες που έχει μια οντότητα παριστάνονται μέσα σε έλλειψη, με υπογραμμισμένο το πρωτεύον κλειδί. Τα πλειότιμα χαρακτηριστικά μιας οντότητας παριστάνονται μέσα σε έλλειψη με διπλό περίγραμμα.

Οι συσχετίσεις ανάμεσα στις οντότητες εμφανίζονται με ρόμβους στο εσωτερικό των οποίων αναγράφεται το όνομα της συσχέτισης με μικρά γράμματα. Έχουν διάφορα χαρακτηριστικά όπως ο βαθμός ή η πολυπλοκότητα και η πληθικότητα. Ο βαθμός μιας συσχέτισης είναι ο αριθμός των τύπων οντοτήτων που παίρνουν μέρος στη συσχέτιση. Οι πιο συνηθισμένες συσχετίσεις είναι:

- **Μοναδικές**, ο βαθμός τους τότε είναι 1.
- **Δυαδικές**, ο βαθμός τους τότε είναι 2.
- **Τριαδικές**, ο βαθμός τους τότε είναι 3.

Η πληθικότητα (Cardinality), περιγράφει τον αριθμό στιγμιοτύπων ενός τύπου οντοτήτων που μπορούν να αντιστοιχίζονται με μία οντότητα ενός άλλου τύπου σε μια συσχέτιση. Ο λόγος πληθικότητας ή πληθικός λόγος (CardinalityRatio), είναι ο λόγος των πληθικοτήτων μιας συσχέτισης. Μπορούμε να έχουμε συσχετίσεις με λόγο πληθικότητας:

- **1-1 (ένα-προς-ένα).** Αντιστοιχίζεται μια οντότητα ενός τύπου με το πολύ ή ακριβώς μια οντότητα ενός άλλου τύπου.
- **1-N (ένα-προς-πολλά).** Αντιστοιχίζεται μια οντότητα ενός τύπου με κανένα, ένα ή πολλά στιγμιότυπα ενός άλλου τύπου.
- **M-N (πολλά-προς-πολλά).** Αντιστοιχίζεται κάθε στιγμιότυπο του ενός τύπου με ένα, κανένα ή πολλά στιγμιότυπα του άλλου τύπου.




**Σχήμα 37.** Παράδειγμα Διαγράμματος Οντοτήτων Συσχετίσεων

Όταν οι πελάτες θα επανεξετάσουν το Διάγραμμα Οντοτήτων Συσχετίσεων θα πρέπει να ελέγξουν κατά πόσο οι σχέσεις που παρουσιάζονται είναι οι ενδεδειγμένες. Μπορούν επίσης να προτείνουν επιπρόσθετες σχέσεις που ανήκουν στο πεδίο εφαρμογής του έργου και δεν έχουν συμπεριληφθεί στο διάγραμμα.

***Λεξικό δεδομένων (ΛΔ)***

Είανι καλό να δημιουργούμε και ένα Λεξικό Δεδομένων που θα συνοδεύει τα διαγράμματά μας για να γίνεται κατανοητή και η χωρίς συγχύσεις η ανάγνωσή τους. Το Λεξικό περιέχει δεδομένα που περιγράφουν τα Διαγράμματα Ροής Δεδομένων και τα Διαγράμματα Οντοτήτων Συσχετίσεων. Συγκεκριμένα, το Λεξικό Δεδομένων περιέχει τις περιγραφές όλων των στοιχείων που εμφανίζονται στα ΔΡΔ και στα ΔΟΣ. Χρησιμοποιείται για την τεκμηρίωση, την αποφυγή πολλαπλών ορισμών, τον έλεγχο των διαγραμμάτων για ακρίβεια και πληρότητα. Είναι πολύ σημαντικό να χρησιμοποιείται η ίδια ονομασία στα διαφορετικά διαγράμματα που αναφέρονται στην ίδια έννοια.

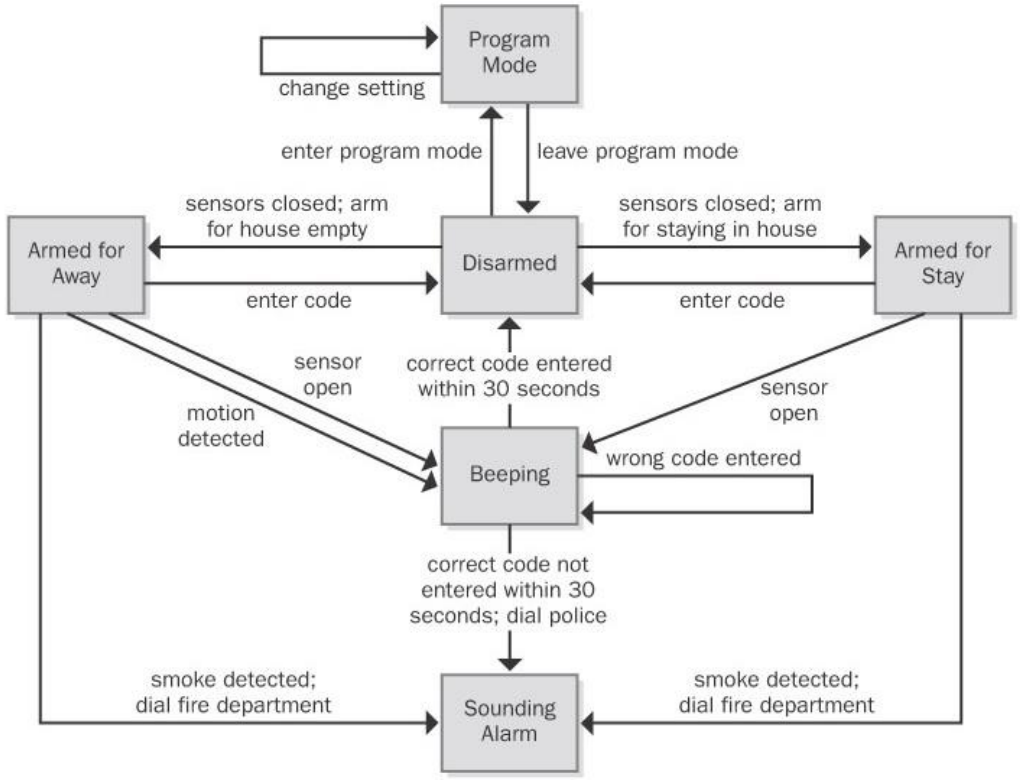
Για την ανάπτυξη ενός τέτοιου λεξικού μπορούν να χρησιμοποιηθούν ειδικά εργαλεία ή CASE Tools. Θα πρέπει να τονιστεί ότι δεν υπάρχει γενικότερη συμφωνία σχετικά με τα δεδομένα που χρησιμοποιούνται για να περιγράψουν τα στοιχεία των διαγραμμάτων.

***State-Transition Diagram***

Η περιγραφή ενός σύνθετου μοντέλου λογισμικού σε φυσική γλώσσα ελοχεύει τον κίνδυνο να παραβλευθεί μία επιτρεπόμενη κατάσταση και να επιτραπεί μία απαγορευμένη. Αυτό καθιστά δύσκολο να επιτευχθεί μια συνολική κατανόηση της συμπεριφοράς του συστήματος.

Το State – Transition Diagram (Διάγραμμα Μετάβασης Καταστάσεων ΔΜΚ) παρέχει μία συνοπτική, πλήρη και σαφή εικόνα αναπαράστασης ενός συστήματος. Περιέχει τρείς τύπους στοιχείων:

- Τις πιθανές καταστάσεις του συστήματος, που εμφανίζονται με ορθογώνια σχήματα.
- Τις επιτρεπόμενες αλλαγές των καταστάσεων ή μεταβάσεις, που εμφανίζονται ως βέλη που ενώνουν τα ορθογώνια.
- Γεγονότα ή συνθήκες που προκαλούν την κάθε μετάβαση, που εμφανίζονται σαν ετικέτες κειμένου σε κάθε βέλος μετάβασης. Το κείμενο αυτό προσδιορίζει τόσο το συμβάν , όσο και την αντίστοιχη απόκριση του συστήματος.




**Σχήμα 38.** Διάγραμμα Μετάβασης Καταστάσεων για ένα οικιακό σύστημα ασφάλειας

Το Διάγραμμα Μετάβασης Καταστάσεων δεν μας δείχνει τις λεπτομέρειες των διεργασιών που εκτελεί το σύστημα, αλλά μόνο τις πιθανές αλλαγές των καταστάσεων μετά τις διεργασίες. Βοηθά τους Dvelopers να κατανοήσουν την προβλεπόμενη συμπεριφορά του συστήματος και είναι ένας καλός τρόπος να ελέγξουμε αν οι μεταβάσεις περιγράφονται σωστά στις λειτουργικές απαιτήσεις. Τέλος από το διάγραμμα αυτό η ομάδα ελέγχου μπορεί να δημιουργήσει περιπτώσεις ελέγχου που θα καλύτουν όλες τις επιτρεπόμενες διαδρομές μετάβασης.

Στο Διάγραμμα Μετάβασης Καταστάσεων θα διακρίνουμε εφτά πιθανές καταστάσεις που μπορεί να πάρει ένα αίτημα. Αυτές είναι οι εξής:

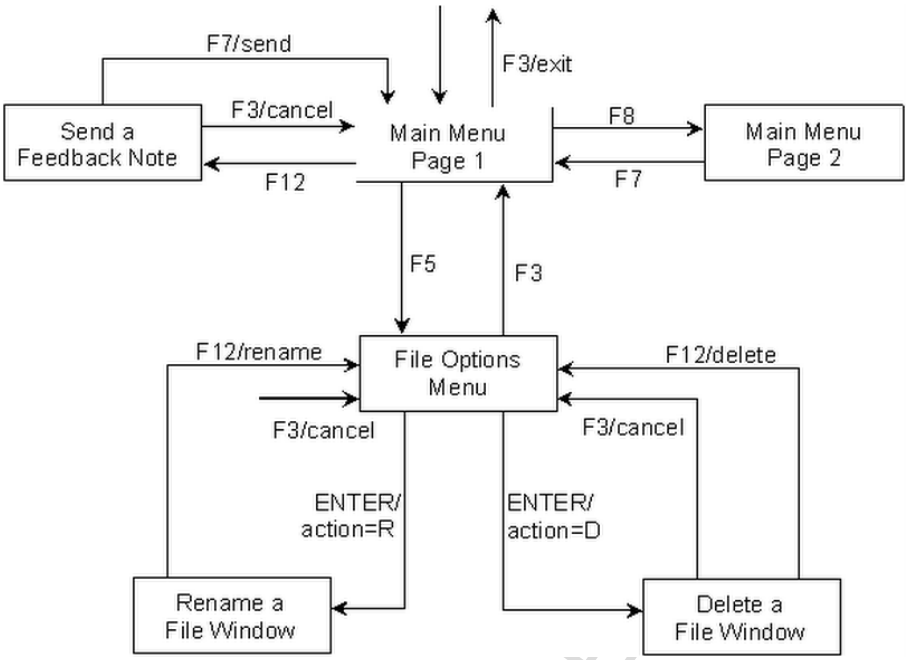
**In Preparation (Σε προετοιμασία).** Δημιουργήθηκε ένα νέο αίτημα.
**Postponed (Αναβλήθηκε).** Το αίτημα αναβλήθηκε για μελλοντική ολοκλήρωση.
**Accepted (Αποδεκτό).** Το αίτημα έγινε αποδεκτό από το σύστημα.
**Placed (Τοποθετήθηκε).** Το αίτημα τοποθετήκε για ολοκλήρωση που πρέπει να γίνει από κάποιον εξωτερικό παράγοντα.
**Fulfilled (Εκπληρώθηκε).** Το αίτημα έχει ικανοποιηθεί.
**Back-Ordered (Επιστροφή Αιτήματος).** Το αίτημα για κάποιο λόγο (όπως το ότι δεν υπάρχει απόθεμα) θα ικανοποιηθεί αργότερα.
**Canceled (Ακυρώθηκε).** Το αίτημα ακυρώθηκε πριν ολοκληρωθεί [21].

***Dialog Map***

Ο χρήστης μπορεί μέσα από τα User Interafaces (Διεπαφές Χρήστη) να πλοηγηθεί ανάμεσα στα πιθανά μονοπάτια διεργασιών του συστήματος. Όταν όμως τα πιθανά μονοπάτια είναι πολλά, τότε η το User Interface θα είναι πολύπλοκο και θα δυσκολεύει τον χρήστη στην πλοήγησή του. Το Dialog Map είναι στην ουσία ένα User Interface με ένα υψηλό επίπεδο αφαίρεσης. Δείχνει την αλληλεπίδραση και τους συνδέσμους των στοιχείων του συστήματος χωρίς να πολλές λεπτομέρειες.

Μέσω του Dialog Map οι χρήστες μπορούν να δουν τις λάθος ή τις περιττές μεταβάσεις και ως εκ τούτου τις ανακριβείς, ή περιττές απαιτήσεις. Λειτουργεί επίσης και σαν οδηγός ενός λεπτομερούς σχεδιασμού του User Interface. Το κάθε στοιχείο (Dialog Element) φαίνεται σαν ορθογώνιο και η κάθε δυνατότητα πλοήγης σαν βέλος. Η κατάσταση που προκαλεί μία πλοήγη εμφανίζεται ως κειμένο στο βέλος μετάβασης. Υπάρχουν διάφοροι τύποι καταστάσεων:

- Μία ενέργεια του χρήστη, όπως το πάτημα ενός πλήκτρου ή η κίνηση του ποντικιού.
- Μία τιμή των δεδομένων, όπως μία μη επιτρεπτή τιμή εισόδου που εμφανίζει ένα σφάλμα στην οθόνη.
- Η κατάσταση ενός συστήματος, όπως η ανίχνευση ότι λείπει χαρτί από τον εκτυπωτή.

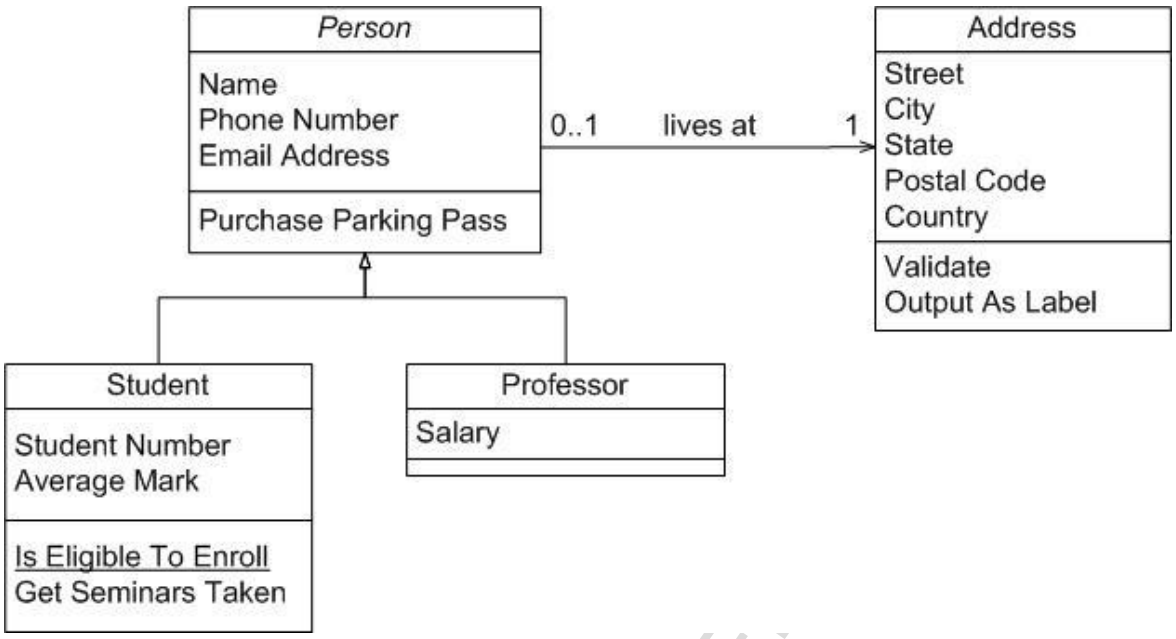



**Σχήμα 39.** Παράδειγμα Dialog Map

***Διαγράμματα Κλάσεων (Class Diagrams)***

Στην αντικειμενοστρεφή ανάπτυξη λογισμικού τα αντικείμενα αντιστοιχούν σε αντικείμενα του πραγματικού κόσμου που έχουν να κάνουν με ένα επιχειρηματικό πρόβλημα και οι κλάσεις περιλαμβάνουν τα χαρακτηριστικά και τις διεργασίες που μπορούν να εκτελέσουν τα αντικείμενα. Τα διαγράμματα κλάσεων αναπαριστούν τη στατική δομή του συστήματος σε επίπεδο κλάσεων, τις σχέσεις μεταξύ των κλάσεων και τα περιεχόμενά τους.

Για την δημιουργία των διαγραμμάτων κλάσης μπορούμε να χρησιμοποιήσουμε τον συμβολισμό της Unified Modeling Language (UML). Ο σχεδιαστής μπορεί να βασιστεί σε αυτά τα εννοιολογικά διαγράμματα κλάσεων για τον σχεδιασμό και την υλοποίηση του συστήματος. Οι αλληλεπιδράσεις μεταξύ των κλάσεων μπορούν περαιτέρω να υποδειχθούν σε Sequence Diagrams και Collaboration Diagrams.




**Σχήμα 40.** Παράδειγμα Διαγράμματος Κλάσεων

Το Σχήμα 40 περιλαμβάνει τέσσερις κλάσεις, την Person (Άτομο), Address (Διεύθυνση), Student (Μαθητής) και Professor (Καθηγητής). Τα χαρακτηριστικά και οι διεργασίες των αντικειμένων φαίνονται αντίστοιχα στο μεσαίο και τελευταίο κουτί της κάθε κλάσης. Οι γραμμές που συνδέουν τις κλάσεις δείχνουν και τις διασυνδέσεις μεταξύ τους. Όπως και Διάγραμμα Οντοτήτων Συσχετίσεων υπάρχουν και συσχετίσεις ένα προς ένα ή ένα προς πειρσσότερα.

Κατά την ανάπτυξη ενός συστήματος οι αλλαγές που προκύπτουν στις κλάσεις, στις διεπαφές, στα χαρακτηριστικά και στις λειτουργίες των κλάσεων πρέπει πρώτα να αναπαρίστανται στα μοντέλα των κλάσεων και ύστερα αυτές οι αλλαγές να πραγματοποιούνται στον κώδικα του συστήματος. Έτσι πετυχαίνουμε [21]:

- Σωστή και πλήρης τεκμηρίωση συστήματος.
- Σωστά και πλήρη παραδοτέα συστήματος.
- Πιο εύκολη συντήρηση συστήματος.
- Ένα κατάλληλο σημείο αναφοράς για όλους που εμπλέκονται στην ανάλυση, τον σχεδιασμό και την ανάπτυξη του συστήματος.

***Μη διαγραμματικά μοντέλα ανάλυσης (Decision Tables and Decision Trees)***

Ένα πρόβλημα στα σύνθετα συστήματα λογισμικού είναι ότι μπορούμε να παραβλέψουμε μία πιθανή κατάσταση. Αυτό οφείλεται στην έλλειψη κάποιων λειτουργικών απαιτήσεων που είναι δύσκολο να εντοπιστούν από μία ανάγνωση του κειμένου προδιαγραφών. Οι πίνακες και τα δέντρα αποφάσεων είναι δύο εναλλακτικές τεχνικές που μπορούμε να χρησιμοποιήσουμε για να

αναπαραστήσουμε το τι πρέπει να κάνει το σύστημα όταν έχουμε να διαπραγματευτούμε με σύνθετες αποφάσεις.

Ο πίνακας απόφασης παρέχει πηροφορίες για τις διάφορες τιμές των παραγόντων που επηρεάζουν την συμπεριφορά του συστήματος καθώς επίσης και για την συμπεριφορά που θα πρέπει να έχει το σύστημα μετά από ένα συνδιασμό παραγόντων. Αυτοί οι παράγοντες είτε είναι συνθήκες True – False, είτε ερωτήσεις με απαντήσεις ΝΑΙ ή ΟΧΙ. Υπάρχουν βέβαια πίνακες απόφασης με παράγοντες που παίρνουν περισσότερες από δύο τιμές.

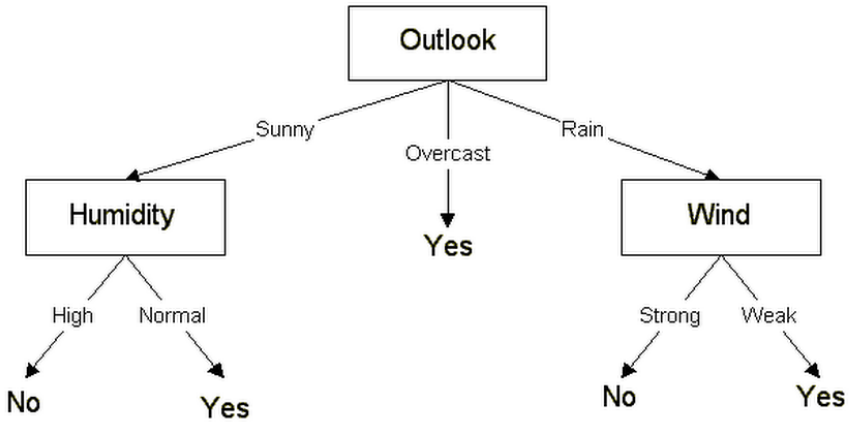
Στον Πίνακα 8 φαίνεται ένας πίνακας απόφασεων που έχει να κάνει με την κατάσταση παραγγελείας κάποιου πελάτη.

| **Παραγγελίες** | | **R1** | **R2** | **R3** | **R4** |
|---|---|---|---|---|---|
| **C1** | Η ποσότητα που παραγγέλθηκε είναι μικρότερη ή ίση του αποθέματος. | **Y** | **Y** | **Y** | **N** |
| **C2** | Ο πελάτης είναι αξιόπιστος. | **Y** | **Y** | **N** | **-** |
| **C3** | Η παραγγελέια είναι μεγαλύτερη ή ίση από το απόθεμα. | **Y** | **N** | **-** | **-** |
| **A1** | Η παραγγελία παραδόθηκε. | **X** | **-** | **-** | **-** |
| **A2** | Η παραγγελία απορρίφθηκε. | **-** | **-** | **X** | **X** |
| **A3** | Η παραγγελεία άλλαξε. | **-** | **X** | **-** | **-** |

**Πίνακας 8.** Παράδειγμα Δέντρου Αποφάσεων

Ο κάθε ένας από τους τρεις παραπάνω παράγοντες έχουν δύο πιθανές καταστάσεις Yes ή No. Αναλόγως από τις τιμές των παραγόντων C1, C2, C3 θα ληφθεί και η απόφαση A1, A2 και A3.

Στο Σχήμα 41 φαίνεται και ένα δέντρο απόφασης βασισμένο στην ίδια λογική πάλι με Yes ή No απαντήσεις για να ληφθεί κάποια απόφαση.




**Σχήμα 41.** Παράδειγμα Δέντρου Αποφάσεων

Τόσο οι πίνακες όσο και τα αποφάσεων είναι χρήσιμα για την αποφυγή τυχόν παραλείψεων συνδιασμών και καταστάσεων. Μία πολύπλοκη διαδιακασία απόφασης μπορεί με ένα ένα τέτοιο εργαλείο να διαβαστεί και κατανοηθεί καλύτερα από ότι σε ένα απλό έγγραφο.


**Σχήμα 42**. Τεχνικές Μοντελοποίησης

Κανένα μοντέλο από μόνο του δεν περιγράφει πλήρως το λογισμικο. Η καταληλότερη λύση είναι να χρησιμοποιηθεί ένας συνδιασμός τους αφού το κάθε ένα συμπληρώνει το άλλο. Η κάθε τεχνική όμως τεχνικές που αναπτύχθηκε δεν παύει να έχει τα δικά της πλεονεκτήματα και τους δικούς της περιορισμούς. Μερικές φορές θα δούμε πως το ένα επικαλύπτει το άλλο και δεν χρειάζεται να τα δημιουργήσουμε όλα κατά την διαδικασία της ανάλυσης.Τα δημιουργούμε επειδή μας παρέχουν ένα υψηλότερο επίπεδο κατανόησης των απαιτήσεων από εκείνο που μας προσφέρει το SRS. Θα πρέπει όμως κάθε φορά να διαλέξουμε την καλύτερη τεχνική ή έναν συνδιασμό τους για να απεικονίσουμε τις απαιτήσεις, τις διεργασίες και τις αλληλεπιδράσείς τους στο σύστημα που δημιουργούμε [2,11,21].

### 3.2.2 Ανάλυση Μη Λειτουργικών Απαιτήσεων

Οι χρήστες πέρα από τις λειτουργικές απαιτήσεις που επθυμούν να κάνει το σύστημα έχουν και προσδοκίες για το πώς αυτό θα πρέπει να λειτουργεί. Τα χαρακτηριστικά που εμπίπτουν σε αυτή την κατηγορία είναι το πόσο εύχρηστο είναι ένα σύστημα, πόσο γρήγορο, πόσο συχνά αποτυγχάνει και πώς θα μπορεί να διαχειριστεί τις απρόβλεπτες συνθήκες. Τα παραπάνω είναι χαρακτηριστικά ή παράγοντας ποιότητας του λογσιμικού και αποτελούν μέρος των μη λειτουργικών απαιτήσεων του συστήματος.

Τα χαρακτηριστικά αυτά είναι δύσκολο να προσδιοριστούν αλλά για να γίνει αντιληπτή η επιτυχία ή αποτυχία των συστημάτων παίζει σημαντικό ρόλο η τήρηση των μη λειτουργικών

απαιτήσεων. Θα πρέπει λοιπόν κατά την εκμαίευση των απαιτήσεων να διερευνηθούν και οι προσδοκίες ποιότητας των πελατών.

Από τεχνικής άποψης τα χαρακτηριστικά ποιότητας οδηγούν στην λήψη σημαντικών αποφάσεων που έχουν να κάνουν με την αρχιτεκτονική και την σχεδίαση του συστήματος. Μπορεί να κριθεί αναγκαίο να διαχωριστεί το σύστημα σε διαφορετικούς υπολογιστές έτσι ώστε να αποκομισθεί η μέγιστη δυνατή και επιθυμητή απόδοση. Είναι πολύ πιο δύσκολο και δαπανηρό να επιτευχθούν οι στόχοι ποιότητας με επανεξέταση της αρχιτεκτονικής ενός ολοκληρωμένου συστήματος απο ότι αν γίνει στον αρχικό σχεδιασμό.

Οι πελάτες συνήθως δεν παρουσιάζουν ρητά τις προσδοκίες ποιότητας που έχουν. Ζητάνε γενικά το σύστημα να είναι φιλικό προς τον χρήστη, να είναι αξιόπιστο, γρήγορο και ισχυρό και είναι δουλειά του αναλυτή να αποκωδικοποιήσει αυτές τις πληροφορίες σε απαιτήσεις. Η τελική ποιότητα του συστήματος ορίζεται τόσο από τους πελάτες, όσο και από τους αναλυτές.

***Ποιοτικά Χαρακτηριστικά***

Υπάρχουν αρκετά χαρακτηριστικά των προϊόντων που μπορούν να οριστούν και σαν χαρακτηριστικά ποιότητας, αλλά για τα περισσότερα έργα θα πρέπει να εξεταστούν μόνο μερικά από αυτά. Ένας τρόπος να ταξινομηθούν τα χαρακτηριστικά είναι τα διακριθούν σε εκείνα τα οποία γίνονται αντιληπτά κατά τον χρόνο εκτέλεσης του συστήματος και σε εκείνα που δεν γίνονται. Ένας άλλος τρόπος είναι να διαχωριστούν εκείνα τα χαρακτηριστικά γνωρίσματα που είναι ορατά από τον χρήστη και σε εκείνα τα οποία δεν είναι, αλλά ωστόσο είναι σημαντικά για το τεχνικό προσωπικό.

Ο Πίνακας 9 παραθέτει διάφορα χαρακτηριστικά ποιότητας που θα πρέπει να εξεταστούν τόσο από την πλευρά των πελατών όσο και από την πλευρά των αναλυτών. Μερικά χαρακτηριστικά όπως η αποδότικοτητα,η αξιοπιστία και η επεκτασιμότητα είναι ζωτικής σημασίας για όλα τα έργα, άλλα όπως η διαθεσιμότητα, η ακεραιότητα και η συντηρισμότητα είναι ιδιαίτερα σημαντικά όταν έχουμε να κάνουμε για εφαρμογές διαδυκτίου, ενώ όταν το λογισμικό θα πρέπει να συνεργαστεί με ήδη υπάρχοντα στην επιχείρηση συστήματα τότε θα πρέπει να είναι συμβατό και διαλειτουργικό.

| **Σημαντικό κυρίως για χρήστες** | **Σημαντικό κυρίως για αναλυτές** |
|---|---|
| Διαθεσιμότητα | Συντηρησιμότητα |
| Αποδοτικότητα | Φορητότητα |
| Ευελιξία - Επεκτασιμότητα | Επαναχρησιμοποίηση |
| Ακεραιότητα | Ελεγξιμότητα |
| Διαλειτουργικότητα | |
| Αξιοπιστία | |
| Ευρωστία | |
| Χρηστικότητα | |

**Πίνακας 9.** Χαρακτηριστικά Ποιότητας Λογισμικού

Σε μία ιδανική περίπτωση το σύστημα θα έπρεπε να εμφάνιζε την μέγιστη δυνατή αξία σε όλα τα παραπάνω χαρακτησιστικά. Θα έπρεπε να ήταν διαθέσιμο ανά πασα χρονική στιγμή, να μην αποτύγχανε, να έδινε πάντα τα σωστά αποτελέσματα και να ήταν εύχρηστο και φιλικό. Επειδή όμως αυτό είναι σχεδόν ανέφικτο θα πρέπει να προσδιοριστούν ποιά χαρακτηριστικά είναι περισσότερο σημαντικά για την επιτυχία του έργου και ποιά λιγότερο. Στην συνέχεια θα πρέπει οι χρήστες και οι αναλυτές να ορίσουν τους στόχους τους σχετικά με αυτές τις παραμέτρους έτσι ώτσε οι σχεδιαστές να κάνουν τις σωστές επιλογές.

Διάφορα τμήματα του προϊόντος μπορεί να χρειάζονται συνδιασμούς χαρακτηριστικών ποιότητας. Θα πρέπει να διαφοροποιηθούν τα χαρακτηριστικά που είναι κοινά για όλο το σύστημα από εκείνα που θα πρέπει να ισχύον σε συγκεκριμένα κομμάτια του, σε ορισμένες κατηγορίες χρηστών και εκείνα που χρειάζονται σε ειδικές συνθήκες χρήσης. Στην συνέχεια αφού τα ποιοτικά χαρακτηριστικά τεκμηριωθούν θα πρέπει να συσχετιστούν με τους επιχειρηματικούς στόχους, τις περιπτώσεις χρήσης και τις λειτουργικές απαιτήσεις του συστήματος.

Οι αναλυτές θα πρέπει να συνεργαστούν με τους χρήστες για να διασφαλιστεί ότι οι μη λειτουργικές απαιτήσεις είναι συγκεκριμένες, μετρήσιμες και επαληθεύσιμες διότι σε διαφορετική περίπτωση δεν θε ξέρουμε αν τελικά έχουν επιτευχθεί. Εάν δεν μπορούν να ποσοτικοποιηθούν θα πρέπει να καθοριστούν οι προτεραιότητες και οι προτιμήσεις των πελατών τουλάχιστον όσον αφορά τις κύριες παραμέτρους της ποιότητας. Το πρότυπο IEEE για Μεθοδολογίες Ποσοτικοποίησης Παραμέτρων Ποιότητας (Software Quality Metrics Methodology) παρουσιάζει μία προσέγγιση για τον καθορισμό των απαιτήσεων της ποιότητας του λογισμικού στα παλίσια ενός συνολικού παλισίου ποσοτικοποίησης ποιότητας.

***Χαρακτηριστικά ποιότητας που είναι σημαντικά στους χρήστες***
Οι χρήστες θεωρούν ιδιαίτερα σημαντικά τα παρακάτω χαρακτηριστικά ποιότητας:

**Διαθεσιμότητα.** Η διαθεσιμότητα του πληροφοριακού συστήματος αποτελεί ένα ιδιαίτερα κρίσιμο και σύνθετο παράγοντα εκτίμησης της ποιότητας της λειτουργίας αυτού. Τυπικά αναφέρεται στο ποσοστό εκείνο του χρόνου κατά το οποίο το σύστημα επιτυγχάνει να προσφέρει τις υπηρεσίες του στον εξυπηρετούμενο στο επίπεδο ποιότητας που αναμένεται να τις παρέχει. Αιτίες αποτυχίας παροχής των υπηρεσιών του Πληροφοριακών Συστημάτων, μεταξύ άλλων είναι και οι εξής:

- Αστοχία υλικού ή τηλεπικοινωνιακής υποδομής, η οποία θέτει εκτός λειτουργίας κάποιο πόρο στον οποίο βασίζει τη λειτουργία του και την παροχή υπηρεσιών του ΠΣ.
- Υπερφόρτωση λογισμικού, υλικού ή τηλεπικοινωνιακών υποδομών με αποτέλεσμα ολική άρνηση εξυπηρέτησης του ή μειωμένη απόδοση κάτω των επιτρεπτών ορίων.

- Μερική ή ολική καταστροφή των δεδομένων.
- Απώλεια της εμπιστοσύνης σε λειτουργίες ή δεδομένα του συστήματος.
- Ανάγκες προγραμματισμένης ή μη προγραμματισμένης συντήρησης συστημάτων υλικού, λογισμικού ή τηλεπικοινωνιών.

Οι αιτίες των παραπάνω μπορεί να είναι:

- Στατιστική πιθανότητα αστοχίας υλικού.
- Κακόβουλη ενέργεια.
- Εσφαλμένη ακούσια ενέργεια.
- Κακή σχεδίαση.
- Φυσική καταστροφή.

**Αποδοτικότητα.** Εάν ένα σύστημα καταναλώνει πολλούς από τους διαθέσιμους πόρους τότε οι χρήστες θα αντιμετωπίσουν μειωμένη απόδοση και αυτό είναι μια ορατή ένδειξη της αναποτελεσματικότητας**.** Το σύστημα λοιπόν θα πρέπει να παρέχει τις υπηρεσίες του στον χρήστη με τρόπο αποδοτικό. Αυτό αναφέρεται σε διάφορες τεχνικές οι οποίες θα πρέπει να εφαρμοστούν:

- Καλορυθμισμένος κώδικας και δομή δεδομένων για τα στοιχεία που θα αναπτυχθούν.
- Κατάλληλη διαμόρφωση και παραμετροποίηση του υλικού και λογισμικού συστήματος που θα φέρει τις υπηρεσίες του Πληροφοριακού Συστήματος.
- Κατάλληλη διαμόρφωση και παραμετροποίηση του λογισμικού εφαρμογών.
- Κατάλληλη σχεδίαση για ελαχιστοποίηση του δικτυακού φόρτου.
- Η διεπαφή χρήστη θα προσφέρει στο χρήστη ένα ιδιαίτερα αποδοτικό περιβάλλον εκμετάλλευσης του συστήματος χωρίς την απαίτηση εκτέλεσης ενεργειών που εισάγουν πρόσθετα βήματα και καθυστερήσεις στην επίτευξη κάποιου στόχου.

Οι χρήστες δεν θα αναφέρουν τις απαιτήσεις απόδοσης με μια τέτοια τεχνική άποψη. Θα αναφέρονται κυρίως σε ό,τι αφορά τους χρόνους απόκρισης ή την χρησιμοποίηση του χώρου του δίσκου. Μετά είναι δουλειά των αναλυτών και των σχεδιαστών να αποδικοποιήσουν τις πληροφορίες των πελατών και να τις ανάγουν σε τεχνικά θέματα.

**Ευελιξία – Επεκτασιμότητα.** Πολύ σημαντικοί παράγοντες για τη διάρκεια ζωής του πληροφοριακού συστήματος που δημιουργείται αλλά και την επιτυχία του είναι η προσαρμοστικότητά του στις ανάγκες των επιμέρους χρηστών και χώρων εφαρμογής και τις μεταβολές που προκύπτουν σε αυτούς, αλλά και η επεκτασιμότητά του ώστε να μπορεί να καλύψει νέα δεδομένα του χώρου που καλείται να εξυπηρετήσει.

Οι δύο παράγοντες αυτοί είναι στενά συνδεμένοι, καθώς η κατάλληλη πρόβλεψη και δυνατότητα παραμετροποίησης του συστήματος μειώνει τις ανάγκες επέκτασής του. Όμως καθώς σε καμία περίπτωση δεν μπορεί κανείς να προβλέψει το σύνολο των μεταβολών που θα προκύψουν κατά τη διάρκεια ζωής ενός Πληροφοριακού Συστήματος αυτό καλείται να είναι δομημένο κατάλληλα ώστε να είναι δυνατή η επέκτασή του.

Η παραμετρική σχεδίαση και εκμετάλλευση όλων των στοιχείων τα οποία δεν αποτελούν φυσικές σταθερές της λειτουργικής περιοχής αποτελεί βασικό παράγοντα μεγιστοποίησης της προσαρμοστικότητας. Εξυπακούεται η δυνατότητα μεταβολής παραμέτρων από διαχειριστές και χρήστες ανάλογα με το επίπεδο εξουσιοδότησης και κρισιμότητας του στοιχείου χωρίς παρέμβαση του κατασκευαστή και λεπτομερής καταγραφή όλων των παραμέτρων που επηρεάζουν τη λειτουργία του συστήματος. Σε συνέχεια της προσαρμοστικότητας, αναφορικά με την επεκτασιμότητα, το σύστημα θα καλύπτει τους παρακάτω κανόνες:

- Ανάπτυξη βασισμένη σε συνιστώσες (components).
- Υποστήριξη περιορισμένου αριθμού προτύπων διεπαφής σε όλο το εύρος της εφαρμογής τα οποία θα κατονομάζονται και δεν θα αλληλο-καλύπτονται μεταξύ τους.
- Οι αναφερόμενες εδώ διεπαφές θα πρέπει να παρουσιάζουν ομοιογένεια στον τρόπο χρήσης τους.
- Οι συνιστώσες θα πρέπει να εμπεριέχουν τη βασική πληροφορία χρήσης τους ή να επισυνάπτεται στην τυπική μορφή που προβλέπεται από το εκάστοτε πρότυπο .
- Χρήση ανοιχτών πρωτοκόλλων και προδιαγραφών.

**Ακεραιότητα.** Η ακεραιότητα, που έχει να κάνει ουσιαστικά με την ασφάλεια, είναι ένας σημαντικός παράγοντας ιδιαίτερα για λογισμικά που δραστηριοποιούνται στο διαδύκτιο και δεν έχει καμμιά ανοχή σε σφάλματα. Η αποτελεσματική διαχείριση της ασφάλειας προϋποθέτει πολλά περισσότερα από ότι η εγκατάσταση πυλών ασφαλείας (Proxy Servers, Firewalls) ή λύσεων ανίχνευσης δικτυακών επιθέσεων. Η υιοθέτηση νέων τεχνολογιών και επιχειρηματιών ευκαιριών, απαιτεί αλλαγές στη δικτυακή υποδομή και η διαχείριση της ασφάλειας πρέπει να συνοδεύεται από συνεχή βελτίωση.

Ο στόχος αυτός προσεγγίζεται μέσα από μία δομημένη και βασισμένη σε standards διαδικασίες καθορισμού, δημιουργίας και υλοποίησης πολιτικής ασφαλείας. Όταν αυτή η προσέγγιση γίνεται πράξη, η διαχείρισης της ασφάλειας μπορεί να γίνει μέσα από προδιαγεγραμμένες διαδικασίες, οι οποίες θα διασφαλίζουν τη ζητούμενη ασφάλεια.

Για την κάλυψη των λειτουργικών αναγκών σύγχρονων συστημάτων Πληροφορικής που αξιοποιούν τις τεχνολογίες του Internet, είναι επιβεβλημένη η αντιμετώπιση μιας σειράς από προκλήσεις που αφορούν την ασφάλεια του συστήματος, οι οποίες είναι πολύ πιο σύνθετες και

δύσκολες από τις αντίστοιχες προκλήσεις που αντιμετώπιζαν οι παραδοσιακές client/server λύσεις. Μεταξύ άλλων, πρέπει να ληφθούν υπόψη παράγοντες όπως:

- Η προστασία των ευαίσθητων δεδομένων, τόσο κατά τη διακίνησή τους πάνω από το δίκτυο, όσο και κατά την αποθήκευσή τους σε κάποια Βάση Δεδομένων (end to end).
- Η πιστοποίηση της ταυτότητας των χρηστών των παρεχόμενων υπηρεσιών / εφαρμογών (authentication), ώστε αυτοί να είναι αναγνωρίσιμοι για λόγους απόδοσης δικαιωμάτων πρόσβασης και καταγραφής των ενεργειών τους (auditing).
- Η εφαρμογή αποτελεσματικών πολιτικών ασφάλειας για τον έλεγχο της πρόσβασης των χρηστών στις εφαρμογές και τα δεδομένα (authorization) με βάση συγκεκριμένα δικαιώματα και σε πολλαπλά επίπεδα.
- Η λεπτομερής καταγραφή των ενεργειών των χρηστών, έτσι ώστε να είναι δυνατός ο έλεγχος για τον εντοπισμό προβλημάτων ασφάλειας και την απόδοση ευθυνών.
- Η ευκολία κεντρικού ελέγχου και διαχείρισης όλων των υπηρεσιών ασφάλειας που διατίθενται από την πλατφόρμα τεχνολογικής υποδομής και τις εφαρμογές, αλλά και των χρηστών και των δικαιωμάτων τους, ανεξάρτητα από τον αριθμό τους.
- Η ευκολία υλοποίησης εφαρμογών οι οποίες θα αξιοποιούν όλες τις υπηρεσίες ασφάλειας που διατίθενται από την πλατφόρμα τεχνολογικής υποδομής.
- Η χρήση ανοικτών προτύπων ασφάλειας και η πιστοποίηση της πλατφόρμας τεχνολογικής υποδομής από διεθνείς οργανισμούς με τα πλέον πρόσφατα κριτήρια ασφάλειας.
- Η αξιοποίηση σύγχρονων τεχνολογιών και αρχιτεκτονικών ασφάλειας σε επίπεδο δικτυακού εξοπλισμού και hardware.
- Η εφαρμογή διαδικασιών και πρακτικών σε επίπεδο οργάνωσης, λειτουργίας, αλλά και χωροταξίας και φυσικής πρόσβασης των συστημάτων που διασφαλίζουν την προστασία τους.

Όσον αφορά την διασύνδεση των χρηστών με το Σύστημα Αποφάσεων, η μετάβαση των πληροφοριών θα γίνεται κρυπτογραφημένα με πιστοποιητικά. Οι χρήστες που θα μπορούν να επεξεργάζονται τα δεδομένα θα πρέπει να αυθεντικοποιούνται με την χρήση username και κωδικών πρόσβασης.

**Διαλειτουργικότητα.** Υπό το πρίσμα της ολικής ηλεκτρονικοποίησης των εργασιών μιας επιχείρησης, θα πρέπει να ληφθούν μέτρα ώστε να επιτυγχάνεται η διασυνδεσιμότητα και η διαλειτουργικότητα των πληροφοριακών συστημάτων. Η διασυνδεσιμότητα αποτελεί τεχνολογικής φύσεως απαίτηση η οποία απαιτεί από ένα πληροφοριακό σύστημα να παρέχει καλά δομημένες διεπαφές εκμετάλλευσής του, βασισμένες σε ευρέως αποδεκτά πρότυπα. Είναι ουσιαστικά απαραίτητη για την επίτευξη της διαλειτουργικότητας των Πληροφοριακών Συστημάτων.

Για την επίτευξη της απαραίτητης διασυνδεσιμότητας, πληροφοριακό σύστημα θα παρέχει καλά δομημένα και τεκμηριωμένα APIs (διεπαφές προγραμματισμού) τα οποία θα επιτρέπουν την μελλοντική εκμετάλλευσή τους από μηχανικούς λογισμικού. Επίσης, όπως απαιτείται από ένα σύγχρονο πληροφοριακό σύστημα θα πρέπει να καλύπτουν τη δυνατότητα εισαγωγής και εξαγωγής στοιχείων μέσω του προτύπου XML και στην ιδανική περίπτωση μέσω του προτύπου Web Service.

Σε κάθε περίπτωση η χρήση ανοιχτών πρωτοκόλλων και προδιαγραφών επιβάλλεται στο σύνολο του συστήματος και για την επίτευξη της διασυνδεσιμότητας. Από την άλλη πλευρά η διαλειτουργικότητα αποτελεί πιο αφηρημένη έννοια και αναφέρεται στη συνεργασία των υπηρεσιών με στόχο την βελτίωση της ολικής λειτουργίας. Η διαλειτουργικότητα παρουσιάζει το τεχνολογικό σκέλος και το επιχειρησιακό σκέλος.

Το επιχειρησιακό σκέλος αφορά στην περιγραφή των διαδικασιών και διεπαφών διαφορετικών. Το τεχνολογικό σκέλος αφορά στην υιοθέτηση ηλεκτρονικών διεπαφών και διαδικασιώνγια την ανταλλαγή πληροφοριών μεταξύ των πληροφοριακών συστημάτων, με στόχο την υποβοήθηση της διαλειτουργικότητας σε επιχειρησιακό επίπεδο.

**Αξιοπιστία.** Η πιθανότητα της εκτέλεσης μίας λειτουργίας του λογισμικού χωρίς βάβη για μια συγκεκριμένη χρονική περίοδο λέγεται αξιοπιστία. Επίσης σαν αξιοπιστία ορίζεται και η ανθεκτικότητα του συστήματος κάτω από συγκεκριμένες συνθήκες. Τα συστήματα που απαιτούν υψηλή αξιοπιστία θα πρέπει επίσης να έχουν σχεδιαστεί με υψηλού επιπέδου ελεγξιμότητα για να καταστήσει ευκολότερη τη εξεύρεση ελαττώματων που θα μπορούσαν να θέσουν σε κίνδυνο την αξιοπιστία.

**Ευρωστία.** Η ευρωστία είναι ο βαθμός στον οποίο ένα σύστημα συνεχίζει να λειτουργεί κανονικά ακόμα και όταν έχει να αντιμετωπίσει άκυρα δεδομένα εισόδου και απροσδόκητες συνθήκες λειτουργίας. Ένα εύρωστο – ισχυρό λογισμικό επανέρχεται γρήγορα από προβληματικές καταστάσεις και λανθασμένους χειρισμούς των χρηστών. Θα πρέπει να ζητήται από τους χρήστες να περιγράψουν συνθήκες σφάλματος τις οποίες μπορεί να αντιμετωπίσει το σύστημα, έτσι ώστε ληφθούν μέτρα να αντιμετωπιστούνάμεσα και γρήγορα.

**Χρηστικότητα**. Ένας πολύ σημαντικός παράγοντας επιτυχίας της εφαρμογής του Πληροφοριακού Συστήματος είναι αυτός της χρηστικότητάς του, αφού αυτή θα αποτελεί βασική απαίτηση για την αποδοχή του από τους χρήστες και την αύξηση της παραγωγικότητας.
Η υιοθέτηση της τυπικής διεπαφής περιηγητή διαδικτύου δίνει μεγάλες δυνατότητες εξατομίκευσης και σαφώς απαιτεί πολύ μικρό χρόνο εξοικείωσης του χρήστη με το σύστημα. Πάνω σε αυτό το μοντέλο, η μέγιστη χρηστικότητα του συστήματος μπορεί να επιτευχθεί αν υπάρχουν τα εξής χαρακτηριστικά:

- Απόδοση, η οποία μετράται αφηρημένα με το λόγο του Έργου που μπορεί να διεκπεραιώσει μια συγκεκριμένη κατηγορία χρηστών, μέσα από μια δεδομένη υποδομή σε κάποιο συγκεκριμένο χρονικό διάστημα. Καθώς μια τέτοια αξιολόγηση δεν είναι ούτε εύκολο αλλά ούτε σκόπιμο να πραγματοποιηθεί, μπορούν να τεθούν κάποιοι γενικοί κανόνες της διεπαφής οι οποίοι θα κατευθύνουν προς εξασφάλιση ενός επιθυμητού επιπέδου απόδοσης, όπως:
- Ο περιορισμός του αριθμού των φορμών, για παράδειγμα, που θα πρέπει να συμπληρώσει ένας χρήστης προκειμένου να προχωρήσει στην επεξεργασία μιας εγγραφής.
- Εκμετάλλευση σύγχρονων στοιχείων ελέγχου (controls) τα οποία προσφέρουν προηγμένες δυνατότητες απεικόνισης και αλληλεπίδρασης.
- Εύκολη πρόσβαση σε εργασίες σχετικές με το απεικονιζόμενο αντικείμενο ή την εκτελούμενη εργασία (contextsensitivecommands). Παράλληλη εκτέλεση πολλαπλών εργασιών (πολλαπλά παράθυρα / πλαίσια).
- Διαισθητικότητα, η οποία αναφέρεται στο χαρακτηριστικό μιας εφαρμογής να είναι εύκολα εκμεταλλεύσιμη από ένα χρήστη ακόμα και αν αυτός δεν είναι εξειδικευμένος στο αντικείμενό της.
- Πλήρης εκμετάλλευση των δυνατοτήτων διεπαφής του συστήματος πελάτη (εικόνα, ήχος, ποντίκι, πληκτρολόγιο)
- Αποκρισιμότητα, η οποία αναφέρεται στην ταχύτητα απόκρισης της διεπαφής χρήστη στις διάφορες εντολές αυτού, η οποία υποστηρίζεται αλλά δεν περιορίζεται από την αποκρισιμότητα του υποκείμενου μηχανισμού προσπέλασης στα δεδομένα.
- Συμμόρφωση με διεθνώς αποδεκτά πρότυπα εμφάνισης και αλληλεπίδρασης με σύστημα.
- Εκμετάλλευση από ΑΜΕΑ (Άτομα Με Ειδικές Ανάγκες) είτε σαν εσωτερικοί χρήστες του Πληροφοριακού Συστήματος είτε σαν εξυπηρετούμενοι.

**Χαρακτηριστικά ποιότητας που είναι σημαντικά στους αναλυτές**

Εδώ περιγράφονται τα χαρακτηριστικά ποιότητας που είναι ιδιαίτερα σημαντικά για τους προγραμματιστές λογισμικού και τους συντηρητές.

**Συντηρησιμότητα.** Η συντηρησιμότητα δείχνει πόσο εύκολο είναι να διορθωθεί ένα ελάττωμα ή να τροποποιηθεί το λογισμικό. Είναι στενά συνδεδεμένη με την ευελιξία και την ελεγξιμότητα. Είναι μία ιδιαίτερα κρίσιμη απαίτηση σε συστήματα που υποβάλλονται σε συχνή αναθεώρηση και σε συστήματα που πρέπει να δημιουργηθούν σε σύντομο χρονικό διάστημα, με αποτέλεσμα να μειωθεί ο τομέας της ποιότητας. Μπορεί να μετρηθεί στον χρόνο που κάνει ένα σύστημα να διορθωθεί και στο ποσοστό του κώδικα που διορθώθηκε στον χρόνο αυτό.

Θα πρέπει να συνεργαστούν οι αναλυτές με τους προγραμματιστές έτσι ώστε να επιλεχθεί ο κατάλληλος κώδιακς που θα μπορεί να τροποποιηθεί και διορθωθεί αποτελεσματικά και

γρήγορα. Στην συνιστώσα της συντηρησιμότητας θα πρέπει να μπουν και οι συσκευές Hardware που έχουν ενσωματωμένο λογισμικό και επηρεάζουν στην πορεία το συνολικό λογισμικό του έργου.

**Φορητότητα.** Φορητότητα είναι η προσπάθεια που απαιτείται για να μεταφερθεί ένα κομμάτι λογισμικού από ένα λειτουργικό περιβάλλον σε ένα άλλο. Οι σχεδιαστικές προσεγγίσεις που καθιστούν ένα σύστημα φορητό είναι πορόμοιες με εκείνες που το καθιστούν επαναχρησιμοποιήσιμο. Η φορητότητα είτε θα είναι αδιάφορη είτε κρίσιμη για ένα πληροφοριακό σύστημα. Θα πρέπει να εντοπιστούν εκείνα τα κομμάτια του προϊόντος που θα χρειαστεί να μεταφερθούν σε άλλα περιβάλλοντα. Υπάρχουν προσεγγίσεις σχεδιασμού και κωδικοποίησης που ενισχύουν την δυνατότητα φορητότητας ενός λογισμικού και αυτές θα πρέπει να χρησιμοποιηθούν.

**Επαναχρησιμοποίηση**. Η επαναχρησιμοποίηση έχει να κάνει με την χρήση του λογισμικού σε άλλες εφαρμογές. Κοστίζει πολύ περισσότερο από ότι η δημιουργία ενός στοιχείου που θα χρησιμοποιηθεί σε μία μόνο εφαρμογή. Το επαναχρησιμοποιήσιμο λογισμικό θα πρέπει να είναι καλά τεκμηριωμένο και ανεξάρτητο από μία συγκεκριμένη εφαρμογή και ένα λειτουργικό περιβάλλον. Είναι μία διαδικασία που ποσοτικοποιείται πολύ δύσκολα. Τα τμήματα του λογισμικού που πρόκειται να επαναχρησιμοποιηθούν θα πρέπει να οριστούν από την αρχή του έργου.

**Ελεγξιμότητα.** Είναι επίσης γνωστή και ως επαληθευσιμότητα και αναφέρεται στα στοιχεία εκείνα του συστήματος που μπορούν να ελεγχθούν για να δούμε τα ελαττώματά τους. Η ελεγξιμότητα είναι ιδιαίτερα κρίσιμη για σύνθετα προϊόντα που έχουν να κάνουν με πολύπλοκους αλγόριθμους και για προϊόντα που θα πρέπει να τροποποιηθούν, διότι υποβάλλονται σε συχνές δοκιμές ώστε να προσδιοριστεί αν οι αλλαγές καταστρέφουν οποιαδήποτε υπάρχουσα λειτουργικότητα.

**Απαιτήσεις Επίδοσης.** Οι απαιτήσεις επίδοσης καθορίζουν πόσο καλά και πόσο γρήγορα εκτελεί το σύστημα συγκεκριμένες λειτουργίες. Περιλαμβάνουν την ταχύτητα και τον χρόνο απόκρισης του συστήματος, την διακίνηση των συναλλαγών που κάνει ανά δευτερόλεπτο και την δυναμικότητα του συστήματος. Οι απαιτήσεις επίδοσης επηρεάζουν τις αποφάσεις για τον σχεδιασμό του λογισμικού και για το υλικό που θα χρησιμοποιηθεί, αλλά ποικίλουν ανάλογα με την φύση του λογισμικού προς κατασκευή. Θα πρέπει να συνυπολογιστούν οι απαιτήσεις του συστήματος σε μία υπερφορτωμένη κατάσταση ή σε μία κατάσταση έκτακτης ανάγκης [5,11].

***Ορισμός μη λειτουργικών απαιτήσεων με τη χρήση της Planguage***

Συνήθως τα χαρακτηριστικά ποιότητας των απαιτήσεων είναι δύσκολο να διευκρινιστούν, με αποτέλεσμα να καθιστούν δύσκολη και την αξιολόγησή τους. Για την αντιμετώπιση του προβλήματος της ασάφειας των μη λειτουργικών απαιτήσεων έχει αναπτυχθεί, από τον Tom

Gilb, η γλώσσα προγραμματισμού Planguage. Είναι μία γλώσσα με ένα σύνολο από λέξεις κλειδιά που μας επιτρέπουν να δηλώσουμε με ακρίβεια τα ποιοτικά χαρακτηριστικά, αλλά και άλλους στόχους του έργου. Παρακάτω υπάρχει ένα παράδειγμα μίας εκδοχής μίας απαίτησης απόδοσης με την χρήση της Planguage [21]:

"*Το 95% των ερωτημάτων της βάσης δεδομένων θα πρέπει να ολοκληρωθεί μέσα σε 3 δευτερόλεπτα χρησιμοποιώντας 1.1 GHz Intel Pentium 4 με λειτουργικό σύστημα MicrosoftWondowsXP με ελεύθερο τουλάχιστον το 60% των πόρων του συστήματος*".

**TAG**. Επίδοση.ΧρονοςΑπόκρισηςΕρωτήματος (Performance.QueryResponseTime).
**AMBITION.** Γρήγορος χρόνος απόκρισης στα ερωτήματα της Βάσης Δεδομένων.
**SCALE.** Ο χρόνος που μεσολαβεί από την έναρξη της διεργασίας έως την εμφάνιση των αποτελεσμάτων.
**METER.** Χρονοετρημένες δοκιμές εκτέλεσης 250 ερωτημάτων που αντιπροσωπεύουν μία επιχειρησιακή διεργασία.
**MUST**. Όχι περισσότερα από 10 δευτερόλεπτα για το 98% των ερωτημάτων.
**PLAN.** Όχι πειρσσότερα από 3 δευτερόλεπτα για μία κατηγορία ερωτημάτων και 8 δευτερόλπετα για όλα τα ερωτήματα.
**WISH**. Όχι περισσότερα από 2 δευτερόλεπτα για όλα τα ερωτήματα.
**BaseUserPlatformDEFINED.** 1.1-GHzIntelPentium 4, 128 MBRAM, MicrosoftWindowsXP, QueryGen 3.3, τουλάχιστον 60% των πόρων του συστήματος ελευθέρα, να μην τρέχει καμμία άλλη εφαρμογή.

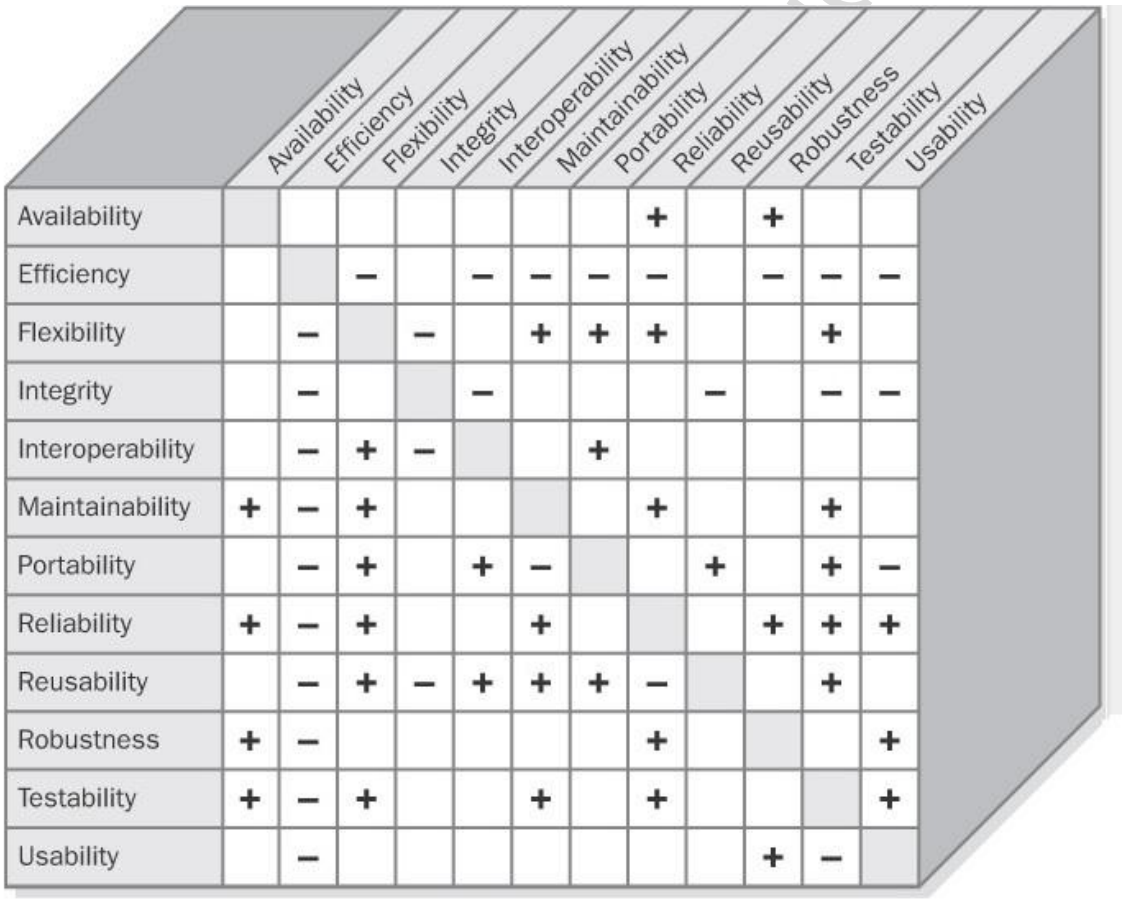
Η κάθε απαίτηση χαρακτηρίζεται με μία μοναδική ετικέτα (TAG, LABEL). Το AMBITION είναι ο στόχος του συστήματος που οδηγεί στην συγκεκριμένη απαίτηση. Το SCALE ορίζει τις μονάδες των μετρήσεων και το METER πώς ακριβώς θα γίνουν οι μετρήσεις αυτές. Το MUST είναι το ελάχιστο αποδεκτό επίπεδο που πρέπει να επιτευχθεί. Το PLAN είναι ο ονομαστικός στόχος και το WISH είναι το ιδανικό επιθυμητό αποτέλεσμα.

Η Planguage περιλαμβάνει και άλλες πρόσθετες λέξεις κλειδιά για να επιτραπεί μεγαλύτερη ευελιξία και ακρίβεια όσον αφορά τις προσδιαγραφές των απαιτήσεων. Παρέχει ένα σαφή καθορσιμό των χαρακτηριστικών των απαιτήσεων ποιότητας και απόδοσης βοηθώντας έτσι τόσο στον καλύτερο προσδιορισμό των απαιτήσεων, όσο και στην ευκολότερη ανάπτυξή τους [34].

***Συμβιβασμοί Χαρακτηριστικών (AttributeTrade-Offs)***
Ορισμένοι συνδιασμοί χαρακτηριστικών θα οδηγήσουν αναπόφευκτα σε ορισμένους συμβιβασμούς. Οι χρήστες και οι προγραμματιστές θα πρέπει να δώσουν προτεραιότητες στα πιο σημαντικά χαρακτηριστικά έτσι ώστε να ληφθούν οι σωστές αποφάσεις υλοποίησης. Το Σχήμα 43 δείχνει μερικές αλληλοεξαρτήσεις των χαρακτηριστικών, που θα βοηθήσουν στο να δωθούν προτεραιότητες. Το θετικό πρόσημο σε ένα κελί δείχνει πως ενα αυξηθεί το

συγκεκριμένο χαρακτηριστικό, τότε θα δημιουργηθεί μία θετική επίδραση και στα υπόλοιπα χαρακτηριστικά της στήλης. Αντιθέτα στα κελιά που έχουν αρνητικό πρόσημο θα δημιουργηθούν αρνητικές επιπτώσεις. Αν για παράδειγμα αυξηθεί η φορητότητα ενός στοιχείου λογισμικού, τότε αυτομάτως θα γίνει πιο ευέλικτο, πιο διαλειτουργικό, ευκολότερα επαναχρησιμοποιήσιμο και ευκολότερα ελέγξιμο.

Για να επιτευχθεί η βέλτιστη ισορροπία των χαρακτηριστικών του προϊόντος, θα πρέπει κατά τη διάρκεια εκμαίευσης των απαιτήσεων να προσδιορίστούν, καθοριστούν, και να δοθούν προτεραιότητες στα σχετικά ποιοτικά χαρακτηριστικά [21].




| | Availability | Efficiency | Flexibility | Integrity | Interoperability | Maintainability | Portability | Reliability | Reusability | Robustness | Testability | Usability |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Availability | | | | | | | | + | | + | | |
| Efficiency | | | − | | − | − | − | − | | − | − | − |
| Flexibility | | − | | − | | + | + | + | | | + | |
| Integrity | | − | | | − | | | | − | | − | − |
| Interoperability | | − | + | − | | | + | | | | | |
| Maintainability | + | − | + | | | | | + | | | + | |
| Portability | | − | + | | + | − | | | + | | + | − |
| Reliability | + | − | + | | | + | | | | + | + | + |
| Reusability | | − | + | − | + | + | + | − | | | + | |
| Robustness | + | − | | | | | | + | | | | + |
| Testability | + | − | + | | | + | | + | | | | + |
| Usability | | − | | | | | | | | + | − | |

**Σχήμα 43.**Θετικές και αρνητικές σχέσεις ανάμεσα στα χαρακτηριστικά πιότητας

**Εφαρμογή μη λειτουργικών απαιτήσεων**

Οι σχεδιαστές και οι προγραμματιστές θα πρέπει να καθορίσουν τον καλύτερο τρόπο να ικανοποιήσουν κάθε παράμετρο ποιότητας και κάθε απαίτηση απόδοσης. Τα χαρακτηριστικά ποιότητας είναι μη λειτουργικές απαιτήσεις δεν αποκλείεται όμως να οδηγήσουν σε λειτουργικές απαιτήσεις, οδηγίες σχεδιασμού και άλλες τεχνικές πληροφορίες που θα συνυπολογισοτύν κατά την υλοποίηση. Ο Πίνακας 10 δείχνει τις κατηγορίες αυτών των τεχνικών πληροφοριών που

στην ουσία καθορίζονται από τα χαρακτηριστικά ποιότητας. Αυτός ο προσδιορισμός είναι μέρος της ανάλυσης των απαιτήσεων.

| **Τύποι Ποιοτικών Χαρακτηριστικών.** | **Κατηγορίες Τενικών Πληροφοριών.** |
|---|---|
| Ακεραιότητα, Διαλειτουργικότητα, Αξιοπιστία, Ευχρηστία, Ασφάλεια. | Λειτουργική Απαίτηση. |
| Διαθεσιμότητα, Αποτελεσματικότητα,Ευελιξία, Απόδοση, Αξιοπιστία. | Αρχιτεκτονική του Συστήματος. |
| Διαλειτουργικότητα, Χρηστικότητα. | Περιορισμοί στον Σχεδιασμό. |
| Ευελιξία, Συντηρησιμότητα, Φορητότητα, Αξιοπιστία, Επαναχρησιμοποίηση, Ελεγξιμότητα, Χρηστικότητα. | Κατευθυντήρια Γραμμή Σχεδιασμού. |
| Φορητότητα. | Περιορισμοί της Εφαρμογής. |

**Πίνακας 10**. Οι Τύποι Ποιοτικών Χαρακτηριστικών και οι Κατηγορίες Τεχνικών Πληροφοριών που αναφέρονται

### 3.2.3 Πρωτοτυποποίηση

Η πρωτοτυποποίηση του λογισμικού κάνει τις απαιτήσεις πιο πραγματικές, αναπαριστά τις περιπτώσεις χρήσης και μειώνει τα κενά κατανόησης των απαιτήσεων. Είναι σαν μία μακέτα ενός κομματιού του νέου συστήματος με σκοπό να κατανοηθούν οι διεργασίες απο τους χρήστες και να ακουστούν οι ενστάσεις που έχουν πάνε σε αυτές. Βοηθά τους ενδιαφερόμενους να καταλήξουν στις απαιτήσεις και μειώνει τον κίνδυνο δυσαρέσκειας των πελατών.

Ακόμα και αν εφαρμοστούν οι πρακτικές ανάλυσης των απαιτήσεων, που περιγράφτηκαν σε προηγούμενα κεφάλαια της εργασίας, ορισμένες από αυτές μπορεί να εξακολουθούν να είναι ασαφής για τους πελάτες ή ακόμα και τους αναλυτές. Είναι δύσκολο να απεικονιστεί με ακρίβεια το πώς θα συμπεριφερθεί το λογισμικό απλά και μόνο με την ανάγνωση κειμένων και την μελέτη των μοντέλων ανάλυσης. Οι χρήστες θα ανταποκριθούν περισσότερο στην δοκιμή ενός πρωτοτύπου παρά στην ανάγνωση της SRS. Η συνηθισμένη απάντηση των χρηστών στο αν τους ικανοποιεί το σύστημα το οποίο βλέπουν στα χαρτιά είναι: *"Θα σας πω όταν δω τι κάνει (I'll know it when I see it – IKIWISI)*. Και αυτήν τυς την προσδοκία έρχεται να ικανοποιήσει το πρωτότυπο.

Το πρωτότυπο έχει πολλαπλές ερμηνείες και οι συμμετέχοντες μπορεί να έχουν διαφορετικές προσδοκίες από αυτό. Ένα πρωτότυπο λογισμικού είναι ένα μέρος του πραγματικού συστήματος, μπορεί να είναι ένα μοντέλο, σχέδια οπτικής απεικόνισης, μία προσομοίωση ή μια εξομοίωση. Θα περιγραφούν τα διαφορετικά είδη των πρωτοτύπων,πώς αυτά μπορούν να χρησιμοποιηθούν κατά την διάρκεια της ανάπτυξης των απαιτήσεων και πώς καθίστανται αναπόσπαστο τμήμα της Μηχανικής των Απαιτήσεων [21,25,36,37].

***Σημασία Πρωτοτύπων***

Ένα Prototype λογισμικού είναι μια μερική ή πιθανή εφαρμογή ενός προτεινόμενου νέου προϊόντος. Και εξυπηρετεί τρεις βασικούς σκοπούς:

**Αποσαφήνιση και να ολοκλήρωση των απαιτήσεων.** Χρησιμοποιείται ως εργαλείο των απαιτήσεων αφού στην ουσία είναι μία προκαταρτική εφαρμογή ενός τμήματος του συστήματος που δεν έχει πλήρως κατανοηθεί. Οι χρήστες αξιολογώντας το πρωτότυπο επισημαίνουν τα προβλήματα που έχουν οι απαιτήσεις και έτσι μπορούν να διορθωθούν με χαμηλό κόστος πριν κατασκευαστεί το συγκεκριμένο σύστημα.

**Εξερευνηση εναλλακτικών προτάσεων σχεδιασμού.** Χρησιμοποιείται και σαν εργαλείο σχεδιασμού καθώς επιτρέπει στους ενδιαφερόμενους να διερευνήσουν διαφορετικές αλληλεπιδράσεις και να αξιολογήσουν πιθανές τεχνικές προσεγγίσης βελτιώνοντας έτσι την χρηστικότητα του συστήματος. Με τον τρόπο αυτό μπορούμε να ελέγξουμε την σκοπιμότητα των απαιτήσεων.

**Ανάπτυξη ολόκληρου του προϊόντος**. Χρησιμοποιείται και σαν εργαλείο κατασκευής. Το prototype είναι μία λειτουργική εφαρμογή ενός υποσυνόλου του συστήματος.Με οδηγό λοιπόν τα Prototypes μπορεί να υλοποιηθεί ολόκληρο το σύστημα.
Ο πρωταρχικός λόγος για τη δημιουργία ενός πρωτοτύπου είναι να εξαλειφθούν οι αβεβαιότητες νωρίς στη διαδικασία ανάπτυξης. Με τον τρόπο αυτό δεν θα αποσαφηνιστούν μόνο οι τυχόν αβεβαιότητες αλλά θα κατανοηθούν πλήρως οι απαιτήσεις τόσο από την μεριά των αναλυτών και των σχεδιατών, όσο και από την πλευρά των υπόλοιπων ενδιαφερομένων.

*Οριζόντια πρωτότυπα*
Όταν αναφέρουμε πρωτότυπο λογισμικού συνήθως συνήθως εννοούμε το οριζόντιο πρωτότυπο μίας πιθανής διεπαφής χρήστη. Το οριζόντιο πρωτότυπο αναφέρεται επίσης και σαν πρωτότυπο συμπεριφοράς ή μακέτα. Λέγεται οριζόντιο γιατί δεν μπαίνει σε όλα τα επίπεδα της αρχιτεκτονικής και στις λεπτομέρειες του συστήματος, αλλά επειδή απεικονίζει ένα τμήμα της διεπαφής του χρήστη. Αυτό το είδος των πρωτοτύπων μας επιτρέπει να εξερευνήσουμε κάποιες συγκεκριμένες συμπεριφορές του συστήματος με απώτερο σκοπό την τελειοποίηση των απαιτήσεων. Βοηθά τους χρήστες να κρίνουν εάν το σύστημα θα τους επιτρέψει να υλοποιήσουν τις διεργασίες που επιθυμούν.

Εμφανίζει τις προσόψεις των διεπαφών του χρήστη και επιτρέπει σε ένα βαθμό την πλοήγηση μεταξύ τους, αλλά δεν περιέχει καμμία πραγματική λειτουργικότητα. Υποδεικνύει τις λειτουργικές επιλογές που θα έχει ο χρήστης στην διάθεσή του, όπως την εμφάνιση, την διάταξη και τα χρώματα και του επιτρέπει να δει μία δομή πλοήγησης. Η πλοήγηση φτάνει έως ένα σημείο και από εκεί και μετά εμφανίζεται ένα μύνημα που περιγράφει τι θα γινόταν πραγματικά.

Η προσομοίωση είναι συχνά αρκετά καλή και επιτρέπει στους χρήστες να καταλάβουν εάν κάποια λειτουργικότητα λείπει,  είναι λάθος, ή είναι περιττή. Αντιπροσωπεύουν την σκοπιά του

αναλυτή για το πώς θα μπορούσε να εφαρμοστεί μία περίπτωση χρήσης. Μπορεί επίσης να επισημάνει εναλλακτικούς τρόπους υλοποίησης μίας διεργασίας.
Το πρωτότυπο σε καμμία περίπτωση δεν είναι το τελικό παραδοτέο σύστημα. Είναι απλά ένας τρόπος να διευκρινιστούν οι απαιτήσεις και να προσδιοριστεί η γενική δομή του συστήματος και θα πρέπει να αντιμετωπίζεται με τον αντίστοιχο τρόπο.

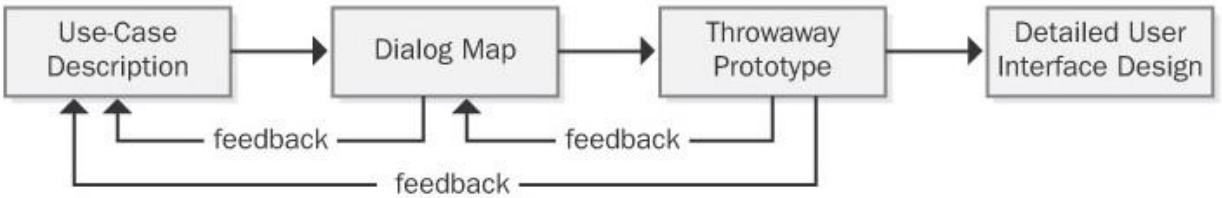
*Κάθετα πρωτότυπα*
Το κάθετο πρωτότυπο, γνωστό και σαν διαρθρωτικό, υλοποιεί ένα κομμάτι της λειτουργικότητας της εφαρμογής μέσα από τα τεχνικά στρώμαρα υπηρεσιών. Αγγίζει όλα τα επίπεδα εφαρμογής και υποτίθεται ότι λειτουργεί όπως και το πραγματικό σύστημα. Αναπτύσσεται όταν δεν είμαστε σίγουροι ότι μία προτεινόμενη αρχιτεκτονική προσέγγιση είναι εφικτή ή όταν θέλουμε να βελτιστοποίησουμε τους αλγορίθμους ή τις βάσεις δεδομένων του συστήματος. Χρησιμοποιούνται για τη διερεύνηση κρίσιμων διεπαφών χρήσης, χρονικών απαιτήσεων και για την μείωση του κινδύνου κατά την διάρκεια του σχεδιασμού.

ThrowawayPrototypes (Πρότυπα μιας χρήσης).
Πριν την κατσκευή ενός πωτοτύπου θα πρέπει να παρθεί μία απόφαση εάν θα απορριφθεί μετά την αξιολόγησή του ή εάν θα γίνει μέρος του παραδοτέου προϊόντος. Η κατασκευή ενός πρωτότυπου μιας χρήσης ή διερευνητικού πρωτότυπου με σκοπό την επίλυση των αβεβαιοτήτων και την βελτίωση των απαιτήσεων ποιότητας θα πρέπει να είναι γρήγορη και φθηνή. Όσο μεγαλύτερη προσπάθεια δίνεται για την δημιουργία του, τόσο πιο δύσκολη θα είναι η απόφαση της απόρριψής του από τους ενδιαφερόμενους.

Ένα πρωτότυπο μιας χρήσης βασίζεται στην γρήγορη εφαρμογή και τροποίηση παρά στην αξιοπιστία, τις επιδόσεις και την μακροπρόθεσμη διατηρησιμότητα. Για τον λόγο αυτό δεν επιτρέπεται σε ένα τέτοιο πρωτότυπο να γίνει η βάση του νέου συστήματος.

Προορίζεται για καταστάσεις που η ομάδα αντιμετωπίζει προβλήματα ασάφειας, ατέλειας ή αβεβαιότητας των απαιτήσεων. Βοηθά τους χρήστες και τους αναλυτές να κρίνουν κατά πόσο οι απαιτήσεις επιτρέπουν τις αναγκαίες επιχειρηματικές διεργασίες.

Το Σχήμα 44 δείχνει την διαδρομή των δραστηριοτήτων ανάπτυξης από τις περιπτώσεις χρήσης στις διεπαφές χρήστη με την βοήθεια ενός πρωτοτύπου μιας χρήσης. Η κάθε περίπτωση χρήσης περιλαμβάνει μία σειρά από ενέργειες χρηστών και απόκρισης του συστήματος που μπορούν να χρησιμοποιηθούν για να δημιουργηθεί ένα μοντέλο μίας πιθανής διεπαφής χρήστη. Όταν οι χρήστες με την σειρά τους αξιολογούν το πρωτότυπο μπορούν να προτείνουν ορισμένες αλλαγές. Αφού οριστούν οι απαιτήσεις η κάθε διεπαφή μπορεί να βελτιστοποιηθεί ως προς την χρηστικότητα.



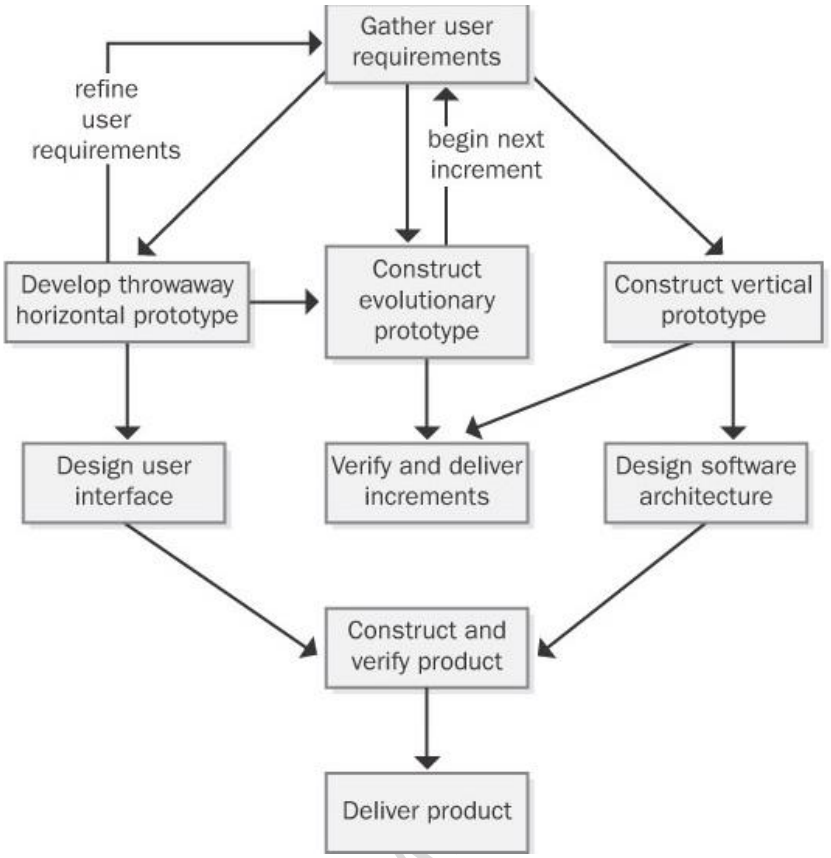

**Σχήμα 44**.Ακολουθία δραστηριοτήτων από περιπτώσεις χρήσης για τη σχεδίαση διεπαφών χρήστη χρησιμοποιώντας ένα πρωτότυπο μιας χρήσης

Εξελικτικό πρωτότυπο

Σε αντίθεση με το πρωτότυπο μίας χρήσης, το εξελικτικό παρέχει μία σταθερή βάση για την αρχιτεκτονική κατασκευή του προϊόντος. Η εξελικτική πρωτυποποίηση είναι συστατικό του μοντέλου σπειροειδής ανάπτυξης του κύκλου ζωής του λογισμικού και ορισμένων αντικειμενοστραφών διαδικασιών ανάπτυξης. Πρέπει να οικοδομηθεί από την αρχή σύμφωνα με τις απαιτήσεις ποιότητας του έργου. Ως εκ τούτου χρειάζεται περισσότερο χρόνο για να δημιουργηθεί από το αντίστοιχο μίας χρήσης που προσομειώνει τις ίδιες δυνατότητες του συτσήματος. Πρέπει να σχεδιάζεται με σκοπό την εύκολη και συχνή αναβάθμιση και δεν υπάρχει χώρος για οποιαδήποτε έκπτωση στην ποιότητα.

Ένα εξελικτικό πρωτότυπο είναι μία πιλοτική απελευθέρωση του συστήματος και η εξέλιξή του είναι και το τελικό προϊόν που θα παραδοθεί στους πελάτες. Τα πρωτότυπα αυτά δίνουν ένα κομμάτι της τελικής λειτουργικότητας στους χρήστες και είναι χρήσιμα για έργα που θα αυξάνονται με την παροδο του χρόνου, ενσωματώνοντας σταδιακά διάφορα συστήματα πληροφοριών.

Το Σχήμα 45 δείχνει διάφορους τρόπους που μπορούν να συνδιαστούν τα διαφορετικά είδη πρωτοτύπων. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε τις γνώσεις που αποκτήθηκαν από μία σειρά πρωτοτύπων μίας χρήσης, για την βελτίωση των απαιτήσεων και να βασιστούμε σε αυτές για την δημιουργία ενός εξελικτικού πρωτοτύπου. Όπως φαίνεται και στο σχήμα μπορούμε να χρησιμοποιήσουμε ένα οριζόντιο πρωτότυπο μιας χρήσης για να αποσαφηνίσουμε πριν την οριστικοποίηση του σχεδιασμού διεπαφής χρήστη, ενώ ταυτόχρονα να δημιουργήσουμε κάθετα πρωτότυπα για να επικυρώσουμε την αρχιτεκτονική, τα στοιχεία της εφαρμογής και τους αλγόριθμους [38,39,40].




**Σχήμα 45.**Πιθανοί τρόποι συνδιασμού πρωτοτύπων κατά την διαδιακασία ανάπτυξης λογισμικού

Ο Πίνακας 11 συνοψίζει μερικές τυπικές εφαρμογές διάφορων ειδών των πρωτοτύπων:

| | **Μίας Χρήσης** | **Εξελικτικά** |
|---|---|---|
| **Οριζόντια** | • Αποσαφήνιση και τελειοποίηση περιπτώσεων χρήσης και λειτουργικών απαιτήσεων.<br>• Προσδιορισμός λειτουργιών που λείπουν.<br>• Εξερεύνηση προσεγγίσεων διεπαφών χρήστη. | • Εφαρμογή κύριων περιπτώσεων χρήσης.<br>• Εφαρμογή πρόσθετες περιπτώσεις χρήσης με βάση την προτεραιότητα.<br>• Προσαρμογή του συστήματος στις ταχέως μεταβαλλόμενες ανάγκες των επιχειρήσεων.<br>• Εφαρμογή και βελτίωση ιστοσελίδων. |
| **Κάθετα** | • Επίδειξη τεχνικής σκοπιμότητας. | • Εφαρμογή και ανάπτυξη λειττουργικότητας με |

| | | |
|---|---|---|
| | | βάση όλων των επιπέδων της αρχιτεκτονικής.<br>• Εφαρμογή και βελτιστοποίηση αλγορίθμων.<br>• Δοκιμή του συστήματος. |

**Πίνακας 11**.Τυπικές Εφαρμογές Πρωτοτύπων

**Χειρόγραφα Και Ηλεκτρονικά Πρωτότυπα**

Για να επιλυθούν οι αβεβαιότητες των απαιτήσεων τα πωτότυπα δεν είναι ανάγκη πάντα να είναι εκτελέσιμα. Ένα έγγραφο πρωτότυπο είναι ένας φθηνός, γρήγορος και αλλά και χαμηλής τεχνολογίας τρόπος να εξερευνήσουμε πώς μοιάζει ένα μέρος του συστήματος. Μας βοηθούν να φτάσουμε σε μία χαμηλού κινδύνου πιθανή λύση πριν τη διαδιακασία της ανάπτυξης. Στα πρωτότυπα αυτά σχεδιάζονται στο χαρτί πώς περίπου θα είναι οι οθόνες των διεπαφών του χρήστη. Οι χρήστες στην συνέχεια δίνουν τις δικές τους ιδέες για την τελική υλοποίηση. Με τα πρωτότυπα, για την κατασκευή των οποίων έχουν χρησιμοποιηθεί ελάχιστοι πόροι, οι χρήστες εκφράζονται πιο εύκολα από ότι στα άλλα που έχουν δαπανηθεί αρκετός χρόνος και πόροι για την υλοποίησή τους.

Η αξιολόγησή τους γίνεται μέχω συνεργασίας των αναλυτών και των χρηστών. Ο αναλυτής έχει τον ρόλο του υπολογιστή και ο χρήστης εκτελεί μία διεργασία ενός σεναρίου αξιολόγησης. Ο χρήστης λέει προφορικά στον αναλυτή τα βήματα που θέλει να ακολουθήσει και ο αναλυτής εμφανίζει μία σελίδα που αντιπρωσοπεύει την οθόνη που θα έβλεπε ο χρήστης. Έτσι μπορεί να κριθεί αν αυτή όντως θα ήταν η αναμενόμενη απόκριση του συστήματος ή όχι και διενεργηθούν οι απαραίτητες αλλαγές.

Ανεξάρτητα από το πόσο αποτελεσματικά είναι τα εργαλεία πρωτοτυποποίησης, η σκιαγράφηση σε χαρτί είναι μία πολύ πιο γρήγορη διαδικασία. Διευκολύνει την ταχύτερη επανάληψη που είναι καθοριστικός παράγοντας στην επιτυχία του έργου. Είναι μία πολύ καλή τεχνική για τον ορισμό των απαιτήσεων πριν την λεπτομερή κατασκευή των διεπαφών χρήστη και των εξελεκτικών μοντέλων πρωτυποποίησης.

Για την δημιουργία ενός ηλεκτρονικού πρωτοτύπου μίας χρήσης είναι κατάλληλα τα παρακάτω εργαλεία:

- Γλώσσες προγραμματισμού όπως Microsoft Visual Basic, IBM Visual Age Small talk, και Inprise Delphi.
- Οι γλώσσες Perl, Python, και Rexx.
- Σχεδιαστικά εργαλεία όπως το Microsoft Visio και το Microsoft Power Point.
- Και εμπορικά εργαλεία δημιουργίας πρωτοτύπων.

Υπάρχουν και προσεγγίσεις με την χρήση HTML (Hypertext Markup Language) που ναι μεν είναι χρήσιμες για την γρήγορη δημιουργία πρωτοτύπων, αλλά δεν βοηθούν στην λεπτομερή σχεδίαση των διεπαφών.

**Αξιολόγηση Πρωτοτύπων**
Για την αξιολόγηση των οριζόντιων πρωτοτύπων θα πρέπει να δημιουργηθούν σενάρια που θα καθοδηγούν τους χρήστες μέσα από μία σειρά βημάτων και μέσω ερωτήσεων που θα αποσπάσουν τις πληροφορίες που χρειάζονται. Αυτές οι ερωτήσεις μπορεί να είναι :

- Το πρωτότυπο εφαρμόζει την ζητούμενη λειτουργικότητα;
- Λειπει κάποια λειτουργικότητα;
- Υπάρχουν περιττές λειτουργίες;
- Η πλοήγηση που γίνεται είναι λογική ή είναι πολύπλοκη;

Η αξιολόγηση θα πρέπει να γίνεται τόσο από έμπειρους, όσο και από άπειρους χρήστες έτσι ώστε να ακουστούν όλες πιθανές γνώμες. Είναι όμως απαραίτητη προϋπόθεση να αντιμετωπίζεται και να αξιολογείται για την συγκεκριμένη λειτουργικότητα, για την οποία έχει κατασκευαστεί, και όχι σαν το τελικό παραδοτέο προϊόν.

Στην συνέχεια θα πρέπει να καταγραφούν οι πληροφορίες που εξορύχθηκαν από την αξιολόγηση και να χρησιμοποιηθούν για την βελτίωση των απαιτήσεων της SRS. Αν η αξιολόγηση οδήγησε στην λήψη αποφάσεων για την δημιουργία διπαφών χρήστη θα πρέπει να καταγραφεί και η λογική που οδήγησε σε αυτές. Τέλος, θα πρέπει να διερευνηθεί εάν υπάρχουν συγκρούσεις μεταξύ του Εγγράφου Προδιαγραφής Απαιτήσεων και των πρωτοτύπων και να επιλυθούν.

Σε κάθε περίπτωση τα πρωτότυπα δεν είναι τα τελικά παραδοτέα του συστήματος. Είναι μία σημαντική βοήθεια για την δημιουργία του προϊόντος και θα πρέπει να χρησιμοποιούνται ως εργαλεία λήψης αποφάσεων και υπό προϋποθέσεις ως βάση για τον σχεδιασμό του συστήματος. Ένα πρωτότυπο δεν αντικαθιστά την Έγγραφο Προδιαγραφής Απαιτήσεων. Αυτό που στην ουσία κάνει είναι να δημιουργεί τις βάσεις για να γίνει το Έγγραφο ακόμα πιο τεκμηριωμένη [21,36,41].

### 3.2.4 Ορισμός Προτεραιοτήτων Των Απαιτήσεων
Για κάθε πρόγραμμα λογισμικού, με περιορσιμένους πόρους, θα πρέπει να καθορίζονται οι σχετιλές προτεραιότητες των απαιτούμενων δυνατοτήτων του. Ο ορισμός των προτεραιοτήτων βοηθά τους διεχειριστές του έργου να πάρουν τις κατάλληλες αποφάσεις και να κάνουν τους αναγκαίους συμβιβασμούς κατά τη διάρκεια της ανάλυσης των απαιτήσεων.

Καταλαβαίνουμε πως όταν οι προσδοκίες του πελάτη είναι υψηλές και το χρονοδιάγραμμα μικρό, θα πρέπει να βεβαιωθούμε πως το προϊόν θα πρέπει προσφέρει τις κατάλληλες

λειτουργίες στον συντομότερο δυνατό χρόνο. Θέτοντας προτεαριότητες στις απαιτήσεις μπορούμε να κατνήμουμε κατάλληλα τον χρόνο και να προσφέρουμε την υψηλότερη αξία στο χαμηλότερο δυνατό κόστος. Στην μεθοδολογία του ExtremeProgramming οι πελάτες μπορούν να ζητήσουν πόσες λειοτυργίες επιθυμούν να έχουν γίνει σε μία συγκεκριμένη χρονική περίοδο και οι αναλυτές με την σειρά τους να ορίσουν πόσες από αυτές μπορούν να συμπεριληφθούν σε κάθε έκδοση.

Ένας Project Manager θα πρέπει να ισορροπήσει το πεδίο εφαρμογής του έργου σε σχέση με τους περιορισμούς, τον προϋπολογισμό, το διαθέσιμο προσωπικό και τους στόχους ποιότητας. Ένας τρόπος να επιτευχθεί αυτό είναι να αναβληθεί η υλοποίηση των απαιτήσεων χαμηλής προτεραιότητας σε κάποια μεταγενέστερη έκδοση του έργου. Για να αποφευχθούν τυχόν διαφωνίες θα πρέπει και οι πελάτες να παίρνουν μέρος στην κατηγοριοποίηση των προτεραιοτήτων. Κάτι τέτοιο είναι πολύ δύσκολο να γίνει, αφού ο κάθε ένας θέτει προτεραιότητες από την δική του σκοπιά, αλλά είναι ζωτικής σημασίας για την επιτυχία του έργου. Οι σχεδιαστές με την σειρά τους μπορούν να υλοποιήσουν και κάποιες απαιτήσεις χαμηλής προτεραιότητας, εάν τους αναγκάσει η αρχιτεκτονική του συστήματος.

Ο καθορισμός των προτεραιοτήτων θα πρέπει να γίνεται νωρίς για να μπορεί να γίνει η επανεκτίμησή τους σε συνάρτηση με τις μεταβαλλόμενες ανάγκες των πελατών, τις συνθήκες της αγοράς και τους επιχειρηματικούς κανόνες που ισχύουν σε κάθε περίοδο. Εάν δοθεί υψηλή προτεραιότητα σε όλες σχεδόν τις απαιτήσεις, τότε το έργο κινδυνεύει να οδηγηθεί σε αποτυχία ή να μην πληρεί τα ποιοτικά χαρακτηριστικά που έχουν οριστεί.

**Κλίμακα Προτεραιοτήτων**

Μία τεχνική είναι η κατηγοριοποίηση των προτεραιοτήτων των απαιτήσεων σε τρεις κύριες κατηγορίες. Στις απαιτήσεις υψηλής, μεσαίας και χαμηλής προτεραιότητας. Η τεχνική αυτή είναι υποκειμενική και θα πρέπει όλοι οι ενδιαφερόμενοι να συμφωνήσουν στο πώς θα γίνει η ιεράρχηση. Ένας τρόπος να γίνει αυτή η ιεράρχηση είναι να αξιολογηθεί η προτεραιότητα σε σχέση με την σημασία που έχει η κάθε απαίτηση και το πόσο επείγουσα είναι. Μία απαίτηση λοιπόν μπορεί να είναι σημαντική ή καθόλου σημαντική και επείγουσα ή όχι επείγουσα. Στον Πίνακα 12 φαίνονται οι πιθανοί συνδιασμοί που καθορίζουν την κλίμακα των προτεραιοτήτων.

| | **Σημαντική** | **Καθόλου Σημαντική** |
|---|---|---|
| **Επείγουσα** | Υψηλή Προτεραιότητα | Μεσαία Προτεραιότητα |
| **Όχι Επείγουσα** | Μεσαία Προτεραιότητα | Χαμηλή Προτεραιότητα |

**Πίνακας 12.** Προτεραιότητες των απαιτήσεων βάση του πόσο σημαντική και επείγουσα είναι

Η προτεραιότητα των απαιτήσεων θα πρέπει να συμπεριλαμβάνεται στις περιγραφές των περιπτώσεων χρήσης. Θα πρέπει επίσης να αποφασιστεί εάν η προτεραιότητα θα κληρονομείται στις δευτερογενείς απαιτήσεις ή εάν θα ορίζεται για κάθε μία ξεχωριστά. Ακόμα και ένα

μεσαίου μεγέθους έργο μπορεί να έχει εκατοντάδες λειτουργικές απαιτήσεις και για τον λόγο αυτό η ιεράρχηση θα πρέπει να γίνει αναλυτικά και με συνέπεια. Θα πρέπει να τεκμηριωθούν ακόμα και οι απαιτήσεις χαμηλής προτεραιότητας γιατί καθώς αλλάζουν οι εξωτερικές συνθήκες μπορεί να αλλάξει και η αρχική ιεράρχηση.

Για τα μεγάλα έργα ανάπτυξης λογισμικού έχουν αναπτυχθεί μαθηματικοί μέθοδοι που ορίζουν προτεραιότητες στις απαιτήσεις.Μία τεχνικήεκτιμά την συνολική αξία που δίνει η κάθε απαίτηση στο έργο και το κόστος υλοποίησής της. Δίνεται έτσι υψηλή προτεραιότητα στις απαιτήσεις που δίνουν την μεγαλύτερη αξία στο σύστημα, ενώ παράλληλα έχουν το χαμηλότερο συνολικό κόστος. Μία άλλη εναλλακτική πρόταση είναι η Quality Function Deployment (QFD), η οποία ιεραρχεί τις προτεραιότητες σε σχέση με την αξία που παίρνει ο πελάτης από κάθε λειτουργία του συστήματος. Η τρίτη και τελευταία τεχνική βαθμολογεί την κάθε απαιτήση σε σχέση με τα διάφορα κριτήρια επιτυχίας του έργου και στο τέλος αθροίζει τις βαθμολογίες για να ταξινομήσει τις προτεραιότητες των απαιτήσεων [21,63,42,43].

### 3.3 Προδιαγραφή Απαιτήσεων

Το αποτέλεσμα της ανάπτυξης των απαιτήσεων είναι μία τεκμηριωμένη συμφωνία μεταξύ των πελατών και των develepes για το προϊόν που πρόκειται να δημιουργηθεί. Όπως είδαμε οι επιχειρηματικές απαιτήσεις υπάρχουν στο έγγραφο οράματος και πεδίου εφαρμογής του έργου, ενώ οι λειτουργικές απαιτήσεις υπάρχουν στις περιπτώσεις χρήσης. Οι λεπτομερείς λειτουργικές και μη λειτουργικές απαιτήσεις υπάρχουν στο Έγγραφο Προδιαγραφής Απαιτήσεων. Αν όμως δεν έχουν γραφεί με έναν οργανωμένο και ευανάγνωστο τρόπο θα είναι πολύ δύσκολο οι ενδιαφερόμενοι φορείς να συμφωνήσουν σε αυτές. Οι απαιτήσεις μπορούν να αναπαρασταθούν με διάφορους τρόπους:

- Σε καλά δομημένα έγγραφα γραμμένα σε φυσική γλώσσα.
- Σε γραφικά μοντέλα που απεικονίζουν τις διεργασίες, τις καταστάσεις του συστήματος, τις σχέσεις των δεδομένων, τις κλάσεις αντικειμένων και τις σχέσεις μεταξύ τους.
- Σε προδιαγραφές που ορζίζουν τις απαιτήσεις χρησιμοποιώντας μαθηματικές γλώσσες.

Η τελευταία κατηγορία ορίζει τις απαιτήσεις με ακρίβεια, αλλά είναι πολύ δύσκολο να γίνουν κατανοητές από τους πελάτες. Έτσι ο ορισμός τους σε μία δομημένη φυσική γλώσσα που εμπλουτίζεται με γραφικά μοντέλα είναι ο καλύτερος και πιο κατανοητός τρόπος τεκμηρίωσης των απαιτήσεων [9].

Η τεκμηρίωση των απαιτήσεων του λογισμικού πολλές φορές ονομάζεται και λειτουργική τεκμηρίωση, τεκμηρίωση του προϊόντος, έγγραφο απαιτήσεων ή τεκμηρίωση του συστήματος. Το Έγγραφο Προσδιορισμού Απαιτήσεων παρέχει με ακρίβεια τις λειτουργίες και τις δυνατότητες που πρέπει να παρέχει ένα σύστημα λογισμικού, όπως επίσης και τους περιορισμούς που πρέπει να υπάρξουν. Είναι η βάση για τον μετέπειτα σχεδιασμό, την

σχεδίαση, την κωδικοποίηση και τον έλεγχο του έργου. περιγράφει την συμπεριφορά του συστήματος κάτω από διάφορες συνθήκες. Πάνω σε αυτό βασίζονται όλοι ενδιφερόμενοι του έργου. Είναι σημαντικό γιατί:

- Οι πελάτες, το τμήμα Marketing και το τμήμα προσωπικού θα πρέπει να ξέρουν προϊόν θα τους παραδοθεί.
- Οι Διαχειριστές του έργου βασίζουν τις εκτιμήσεις στο χρονοδιάγραμμα, στους πόρους και στην περιγραφή του συστήματος.
- Το Έγγραφο Προδιαγραφής Απαιτήσεων στην ουσία καθοδηγεί την ομάδα ανάπτυξης στο τι θα δημιουργήσει.
- Η ομάδα δοκιμών το χρησιμοποιεί για να αναπτύξει σχέδια δοκιμών.
- Σε αυτό βασιστεί και ομάδα που θα αναλάβει την εγγαρφή των εγχειριδίων χρήσης.
- Το νομικό προσωπικό διασφαλίζει ότι οι απαιτήσεις είναι συμβατές με τους ισχύοντες νόμους και κανονισμούς.

Είναι ο τελικός αποδέκτης των απαιτήσεων του έργου. Αν κάποια απαίτηση δεν εμφανίζεται στο Έγγραφο Προδιαγραφής Απαιτήσεων, τότε δεν θα εμφανιστεί και στο έργο. Δεν θα πρέπει να γραφτούν όλες οι απαιτήσεις που αφορούν το έργο πριν ξεκινήσει η ανάπτυξη, αλλά θα πρέπει κάθε απαίτηση που αναπτύσσεται πρώτα να έχει καταγραφεί. Η επαυξητική ανάπτυξη είναι η καταλληλότερη όταν οι ενδιαφερόμενοι δεν μπορούν να κατγράψουν εκ των προτέρων όλες τις αναγκαίες απαιτήσεις και όταν είναι επιτακτική ανάγκη να δημιουργηθεί γρήγορα κάποια λειτουργικότητα. Ωστόσο θα πρέπει πρώτα να συμφωνηθούν οι απαιτήσεις και μετά να υλοποιηθούν [2,11,21].

### 3.3.1 Σωστά ΔομημένόΈγγραφο Προδιαγραφής Απαιτήσεων

Ένα σωστά δομημένο δομημένο Έγγραφο Προδιαγραφής Απαιτήσεων θα πρέπει να έχει τα ακόλουθα χαρακτηριστικά [44]:

- Ενότητες και υποενότητες.
- Πίνακα περιεχομένων και ευρετήριο, ώστε να βρίσκονται έυκολα οι απαιτούμενες πληροφορίες.
- Αριθμημένους πίνακες και εικόνες.
- Παραπομπές και αναφορές σε άλλα σημεία του εγγράφου.

**Labeling Requirements.** Για να γίνουν πιο εύκολες οι διαδικασίες ιχνηλασιμότητας και τροποποίησης θα πρέπει η κάθε λειτουργική απαίτηση να αναπαριστάται μοναδικά. Έτσι θα κερδίζουμε χρόνο κάθε φορά που θέλουμε να κάνουμε μία αλλαγή, μία τροποποίηση και μία παραπομπή. Υπάρχουν διάφορες τεχνικές, που αναλύονται παρακάτω, για να χαρακτηρίσουμε μοναδικά κάθε απαίτηση.

**Sequence Number**. Η πιο απλή προσέγγιση είναι να χαρακτηρίσουμε την κάθε απαίτηση με έναν μοναδικό αύξοντα αριθμό, όπως UR-9. Τα εμπορικά εργαλεία διαχείρησης απαιτήσεων το κάνουν αυτόματα όταν εισάγεται μία νέα απαίτηση. Το πρόθεμα δηλώνει το είδος της απαίτησης, για πράδειγμα UR (όταν πρόκειται για απαίτηση χρήστη). Αυτή η απλή μέθοδος δεν ομαδοποιεί τις απαιτήσεις που συσχετίζονται, απλά διαοφροποιεί μοναδικά την κάθε απίτηση από τις άλλες.

**Ιεραρχική Αρίθμηση.** Η σύμβαση που χρησιμοποιείται συχνότερα είναι όταν οι λειτουργικές απαιτήσεις εμφανίζονται στην ενότητα 3.2 του Εγγράφου Προδιαγραφής Απαιτήσεων να έχουν ετικέτες που θα ξεκινούν από 3.2, δηλαδή να είναι της μορφής 3.2.3.4. Είναι μία μέθοδος απλή και αποδοτική. Είναι μία διαδικασία που μπορεί να κάνει ένας οποισδήποτε επεξεργαστής κειμένου. Ούτε αυτή η προσέγγιση δεν ομαδοποεί τις απαιτήσεις.

**Ιεραρχική επισήμανση απαιτήσεων**. Υπαρχει και ένας τρόπος ιεραρχικής επισήμανσης των απαιτήσεων. Αν δηλαδή υπαάρχει η απαίτηση " Το σύστημα να ζητήσει από τον χρήστη την επιβεβαίωση οποιασδήποτε εκτύπωσης μεγαλύτερης από 10 αντιγράφων" ("The system shall ask the user to confirm any request to print more than 10 copies"), τότε θα καταγράφεται σαν "Print.ConfirmCopies". Είναι μία δομημένη προσέγγιση και δεν επηρεάζεται από την προσθήκη ή την διαγραφή άλλων απαιτήσεων. Το μειονέκτημα που έχει είναι ότι αυτή η κατηγοριοποίηση είναι πιο ογκώδης από εκείνες με τις αριθμητικές ετικέτες. Είναι πάντως πιο αποδοτική και συνεπής από ότι οι προηγούμενες δύο.

Μερικές φορές μπορεί να λείπει ένα κομμάτι πληροφορίας για κάποια απαίτηση. Σε αυτή την περίπτωση θα πρέπει να ρωτηθεί η γνώμη των πελατών επί του θέματος ή να κατασκευαστεί ένα Prototype για να διευκρινιστούν οι αβεβαιότητες. Επίσης θα πρέπει αυτή η απαίτηση να επισημανθεί σαν TBD (To Be Determined - Προς Καθορισμό). Οι απαιτήσεις που είναι προς TBD θα πρέπει να διευκρινιστούν πριν υλοποιηθούν γιατί σε αντίθετη περίπτωση ο προγραμματιστής είτε θα τις εμηνεύσει με την δική του εκδοχή που μπορεί να μην είναι η επιθυμητή, είτε θα αναβάλει την υλοποίησή τους αναβάλοντας έτσι και την ολοκλήρωση του έργου.

Η ενσωμάτωση των διεπαφών χρήστη στο Έγγραφο Προδιαγραφής Απαιτήσεων έχει και πλεονεκτήματα και μειονεκτήματα. Το μειονέκτημα είναι πως οι εικόνες της οθόνης και οι διεπαφές χρήστη περιγράφουν λύσεις και όχι απαιτήσεις. Το πλεονέκτημα είναι ότι διερευνώνται όλα οι πιθανές διεπαφές χρήστη και γίνονται ορατές οι απαιτήσεις και από την μεριά του χρήστη αλλά από την μεριά του προγραμματιστή. Οι διεπαφές χρήστη βοηθούν στον προγραμματισμό και την εκτίμηση του όλου έργου.

***Πρότυπο Προδιαγραφών των Απαιτήσεων***

Κάθε οργανισμός ανάπτυξης θα πρέπει να υιοθετήσει ένα ή περισσότερα πρότυπαΕγγράφων Προδιαγραφής Απαιτήσεων από τα πολλά που είναι διαθέσιμα. Οι πιο πολλοί χρησιμοποιούν το πρότυπο IEEE 830-1998 που είναι κατάλληλο για πολλά ήδη έργων αλλά έχει και μερικούς περιορισμούς. Αν η εταιρεία αναλαμβάνει την ανάπτυξη διάφορων ηδών λογισμικού, από μικρές βελτιώσεις σε ήδη υπάρχοντα συστήματα έως την δημιουργία ενός μεγάλου έργου, τότε θα πρέπει να έχει ένα Έγγραφο Προδιαγραφής Απαιτήσεων για κάθε κατηγορία [21].

Στο παρακάτω σχήμα φαίνεται ένα Έγγραφο Προδιαγραφής Απαιτήσεων που προασαρμόστηκε από το IEEE 830-1998 std. Το πρότυπο θα πρέπει να τροποποείται και να προσαρμόζεται στις ανάγκες και την φύση των έργων που αναπτύσσονται. Έαν ένα τμήμα του προτύπου δεν ισχύει για κάποιο συγκεκριμένο έργο θα πρέπει να το αφήνουμε όπως έχει και να υποδεικνύουμε ότι δεν είναι εφαρμόσιμο. Με τον τρόπο αυτό γίνονται πιο ξεκάθαρα τα πράγματα στον αναγνώστη και δεν αναρωτιέται αν έχει παραλειφθεί κάτι κατά λάθος. Αν η παράλειψη κάποιων σημείων του προτύπου είναι σύνηθες φαινόμενο για μία εταιρεία, τότε μπορούμε να αλάξουμε τον πίνακα περιεχομένων τουΕγγράφου Προδιαγραφής Απαιτήσεων αλλά θα πρέπει να αναφέρουμε τις αλλαγές που έγιναν, το άτομο που τις έκανε και τον λόγο που έγιναν. Μερικές υπάρχει η ανάγκη ένα κομμάτι πληροφορίας να καταγράφεται σε διάφορα τμήματα του προτύπου. Στην περίπτωση αυτή είναι καλύτερο να επαναλάβουμε την καταγραφή όσες φορές χρειαστεί παρά να δημιουργήσουμε σύγχυση γύρω από το πού ανήκει το συγκεκριμένο κομμάτι πληροφορίας.

Υπάρχει ένα κομμάτι αλληλοεπικάλυψης μεταξύ του Εγγράφου Προδιαγραφής Απαιτήσεων και του εγγράφου οράματος και πεδίου εφαρμογής. Αν χρησιμοποιούμε επιλέξουμε να χρησιμοποιήσουμε και τα δύο αυτά πρότυπα θα πρέπει να τα προσαρμόσουμε κατάλληλα για να αποφύγουμε τις επικαλύψεις και τις διπλοεγγραφές. Θα ήταν καλό να χρησιμοποιήσουμε το Έγγραφο Προδιαγραφής Απαιτήσεων για να επισημάνουμε κάποιες λεπτιμέρεις που εμφανίστηκαν στις πληροφορίες του εγγράφου οράματος και πεδίου εφαρμογής. Παρακάτω γίνεται περιγραφή των πληροφοριών που εμφανίζονται στο Έγγραφο Προδιαγραφής Απαιτήσεων.

- **Introduction (Εισαγωγή).**Εδώ γίνεται μία επισκόπηση που βοηθάει τον αναγνώστη να καταλάβει πώς έχει οργανωθεί το SRS και πώς να το χρησιμοποιήσει.

***Purpose (Σκοπός).*** Προσδιορίζεται το προϊόν ή η εφαρμογή των οποίων οι απαιτήσεις καθορίζονται στην SRS. Αν το SRS αναφέρεται σε ένα μόνο κομμάτι του συστήματος, τότε θα πρέπει να το αναφέρουμε.

***Document Conventions (Συμβάσεις εγγράφων*).** Περιγράφονται οι συμβάσεις που έχουν υιοθετηθεί και συμπεριλαμβάνουν τα στυλ του κειμένου, επισημάνσεις ή σημαντικές σημειώσεις.

***Intended Audience and Reading Suggestions*** (Προβλεπόμενοι αναγνώστες και προτάσεις διαβάσματος). Θα πρέπει να καταγραφούν οι πιθανοί αναγνώστες στους οποίους απευθύνεται η SRS. Επίσης δίνεται η κατάλληλη σειρά ανάγνωσης της SRS για τις διάφορες κατηγορίες των αναγνωστών.

***ProjectScope (Πεδίο εφαρμογής).***Δίνετε μία σύντομη περιγραφή του λογισμικού και περιγράφεται ο σκοπός του. Αφορά τους εταιρικούς στόχους και τις στρατηγικές της επιχείρησης. Αν υπάρχει ξεχωριστό έγγραφο οράματος και πεδίου εφαρμογής είναι καλύτερα να γίνονται αναφορές σε αυτό παρά να επαναλαμβάνεται η ίδια πληροφορία. Θα πρέπει το πεδίο εφαρμογής να δίνεται σαν υποσύνολο του μακροπρόσθεσμου στρατηγικού οράματος της επιχείρησης.

***References (Αναφορές).*** Καταγραφή των εγγράφων που εμαφνίζονται στην SRS και αν είναι δυαντόν και καταγραφή των διασυνδεσέων μεταξύ τους. Τα έγγραφα αυτά αφορούν συμβάσεις, πρότυπα, προσδιορισμό απαιτήσεων συστήματος ή ακόμα και το SRS ενός συσχετιζόμενου προϊόντος. Παρέχονται πληροφορίες με τις οποίες ο χρήστης είναι σε θέση να γνωρίζει τον τίτλο, την ημερομηνία δημιουργίας, τον συγγραφέα και την πηγή της κάθε αναφοράς.

- **Overall Description (Συνολική Περιγραφή).**

Αυτή η ενότητα περιέχει μία υψηλού επιπέδου επισκόπηση του προϊόντος και του περιβάλλοντος στο οποίο θα χρησιμοποιηθεί. Περιέχει επίσης τα προσδωκόμενα από τους χρήστες προϊόντα, τους περιορισμούς, τις παραδοχές και τις εξαρτήσεις που υπάρχουν.

***Product Perspective (Προοπτική του Προϊόντος).*** Περιγράφεται το πλαίσιο και η εφαρμογή του προϊόντος. αν δηλαδή το προϊόν που αναπτύσσεται είναι μέλος μία ευρύυερης ομάδας προϊόντων, αν είναι η επόμενη έκδοση ενός συστήματος, αν πρόκειται για αντικατάσταση ενός ήδη υπάρχοντος ή αν πρόκειται για ένα εντελώς νέο προϊόν. Αν πρόκεται για ένα κομμάτι ενός ευρύτερου συστήματος, τότε πρέπει να δηλωθεί πώς σχετίζεται με το λογισμικό του ευρύτερου προϊόντος και να προσδιορισοτύν οι βασικές διεπαφές μεταξύ τους.

***Product Features ( Χαρακτηριστικά του Προϊόντος).*** Γίνεται καταγραφή των κυριότερων χαρακτηριστικών του προϊόντος ή των σημαντικών λειτουργιών που εκτελεί. Στην ουσία πρόκειται για μία περίληψη των χαρακτηριστικών, αφού οι λεπτομέρεις θα δοθούν στο τρίτο κομμάτι του SRS. Καταγράφονται δηλαδή οι σημαντικότερες ομάδες των απαιτήσεων και πώς αυτές είναι συσχετισμένες. Γίνεται χρήση Διαγράμματων Ροής Δεδομένων (ΔΡΔ), διαγράμματα περίπτωσεων χρήσης και διαγράμματα κλάσεων.

***User Classesand Characteristics (Κατηγορίες και Χαρακτηριστικά Χρηστών).*** Εδώ προσδιορίζονται οι διάφορες κατηγορίες των χρηστών που θα χρησιμοποιήσουν το σύστημα και περιγράφονται τα χαρακτηριστικά τους. Ορισμένες απαιτήσεις αφορούν μόνο μερικές κατηγορίες χρηστών. Οι κατηγορίες αυτές είναι ένα υποσύνολο των ενδιαφερομένων που περιγράφονται στο έγγραφο οράματος και πεδίου εφαρμογής.

***Operating Environment (Περιβάλλον Λειτουργίας).*** Περιγράφεται το περιβάλλον στο οποίο θα λειτουργεί το λογισμικό. Συμπεριλαμβάνει την πλατφόρμα λειτουργίας, τις εκδόσεις των λειτουργικών συστημάτων, τις γεωγραφικές τοποθεσίες των χρηστών, τους εξυπηρετητές

(servers) και τις βάσεις δεδομένων. Καταγράφονται τα στοιχεία του λογισμικού και οι εφαρμογές με τις οποίες το σύστημα θα πρέπει να συνυπάρχει. Μερικές από αυτές τις πληροφορίες μπορεί να περιέχονται και στο έγγραφο οράματος και πεδίου εφαρμογής, αλλά εδώ γίνεται λεπτομερής αναφορά.

***Design and Implementation Constraints (Περιορισμοί σχεδιασμού και υλοποίησης).*** Παρουσιάζονται οι παράγοντες που περιορίζουν τις διαθέσιμες επιλογές της ομάδας ανάπτυξης. Οι περιορισμοί περιλαμβάνουν τα ακόλυθα:

- ✓ Ειδικές τεχνολογίες, εργαλεία, γλώωσες προγραμματισμού και βάσεις δεδομένων που πρέπει ή δεν πρέπει να χρησιμοποιηθούν.
- ✓ Περιορισμοί λόγω του περιβάλλοντος λειτουργίας του προϊόντος, όπως τα είδη και τις εκδόσεις των webbrowsers που θα χρησιμοποιηθούν.
- ✓ Απαιτούμενες συμβάσεις ή πρότυπα ανάπτυξης που πρέπει να ακολουθηθούν.
- ✓ Συμβασιμότητα με ήδη υπάρχοντα προϊόντα και συστήματα.
- ✓ Περιορισμοί που επιβάλονται από τους επιχειρηματικούς κανόνες.
- ✓ Περιορισμοί που αφορούν το hardware. Περιορισμοί δηλασδή που αφορούν τοην μνήμη, τον πεξαργαστή, το βάρος, το μέγεθος και το κόστος.
- ✓ Πρότυπα ανταλλαγής μορφών δεδομένων, όπως η XML.

***User Documentation (Τεκμηρίωση των χρηστών).***

Περιλαμβάνουν εγχειρίδια χρήσης και online ηλεκτρονική βοήθεια και τον προσδιορισμό των παραπάνω ργγράφων ή εργαλείων.

***Assumptions and Dependencies (Παραδοχές-Υποθέσεις και Εξαρτήσεις).*** Οι υποθέσεις είναι οι δηλώσεις που πιστεύεται ότι είναι αληθείς αλλά δεν υπάρχουν στοιχεία να τις τεκμηριώνουν. Αυτές μπορεί να οδηγήσουν σε λάθος γιατί οι διαφορετικοί αναγνώστες της SRS μπορεί να τους δώσουν διαφορετικές ερμηνείες. Οι εξαρτήσεις αναφέρονται σε εξωγενείς παράγοντες, όπως συστήματα που πρέπει να ενσωματωθούν και εξαρτώνται από αυτό που δημιουργείται. Θα πρέπει να γίνει πλήρης καταγραφή των εξαρτήσεων για να εξασφαλιστεί η σωστή δημιουργία και η συμβατότητα με τα άλλα υποσυστήματα.

- **System Features (Χαρακτηριστικά του Συστήματος).**

Το παραπάνω πρότυπο οργανώθηκε με βάση τα χαρακτηριστικά του συστήματος. Είναι ένας τρόπος να οριστούν οι λειτουργικές απαιτήσεις. Άλλοι τύποι οργάνωσης περιλαμβάνουν περιπτώσεις χρήσης, τον τρόπο λειτουργίας, τις κατηγορίες των χρηστών, τις κλάσεις των αντικειμένων, την λειτουργική ιεράρχηση ή των συνδιασμό τους. Δεν υπάρχει μία μοναδική σωστή επιλογή. Θα πρέπει η οργανωτική προσέγγιση να καθιστά εύκολο στους χρήστες να να κατανοήσουν τις δυνατότητες του συστήματος.

***3.x System Feature X (Χαρακτηριστικό Συστήματος X).*** Δήλωση του χαρακτηριστικού με λίγες λέξεις. Όταν δηλαδή έχουμε "3.1 Spell Check." θα πρέπει να γίνεται επανάληψη από το 3.x.1 έως το 3.x.3.

***3.x.1 Description and Priority (Περιγραφή Προτεραιότητας).*** Γίνεται σύντομη περιγραφή της λειτουργίας και δηλώνεται αν η προτεραιότητά της είναι υψηλή, μεσαία ή χαμηλή. Οι προτεραιότητες είναι δυναμικές και μπορούν να αλλάξουν κατά την πορεία του έργου.
***3.x.2 Stimulus / Respons eSequences (Ακολουθίες Απόκρισης Ερεθισμάτων).*** Καταγράφονται οι ακολουθίες των ερεθισμάτων εισόδου, από εξωτερικές συσκευές ή από ενέργειες χρηστών και οι αποκρίσεις που θα πρέπει να έχει το σύστημα σε αυτές. Αντιστοιχούν στις περιπτώσεις χρήσης ή σε εξωτερικά συμβάντα.
***3.x.3 Functional Requirements (Λειτουργικές Απαιτήσεις).*** Εδώ γίνεται η ταξινόμηση των λειουργικών απαιτήσεων που αντιστοιχούν σε ένα χαρακτηριστικό. Είναι οι δυαντότητες που θα πρέπει να έχει το λογισμικό για να ανταποκρίνεται στις διεργασίες του χρήστη και να εκτελέσει μία περίπτωση χρήσης. Επίσης γίνεται καταγραφή του πώς θα πρέπει να αντιδρά το σύστημα σε λανθασμένα δεδομένα εισόδου. Θα πρέπει να καθορίζεται μοναδικά η κάθε λειτουργική απαίτηση.

- **External Interface Requirements**

Οι απαιτήσεις των εξωτερικών διεπαφών (External Interface Requirements) είναι ο καθορισμός του hardware, του λογισμικού και των στοιχείων της βάσης δεδομένων με τα οποία το σύστημα θα πρέπει να επικοινωνεί. Εδώ παρέχονται πληροφορίες που διασφαλίζουν την σωστή επικοινωνία του συστήματος με τα εξωτερικά εξαρτήματα. Αν μερικά τμήματα του λογισμικού έχουν διαφορετικές εξωτερικές διασυνδέσεις θα πρέπει να καταγραφούν λεπτομερώς. Είναι ένας κρίσιμος παράγοντας για την επιτυχία του έργου και μπορεί να συμπεριλαμβάνει υλικό από άλλα έγγραφα και αναφορές.
***User Interfaces.*** Περιγράφονται τα λογικά χαρακτηριστικά κάθε διεπαφής χρήστη που χρειάζεται το σύστημα. Μερικά πιθανά στοιχεία είναι:

- ✓ Αναφορές στα πρότυπα GUI που πρόκειται να ακολουθηθούν.
- ✓ Πρότυπα για τις γραμματοσειρές, τις εικόνες και τα χρώματα.
- ✓ Διάταξη της οθόνης ή περιορισμοί ανάλυσης.
- ✓ Πλήκτρα συντόμευσης.
- ✓ Επιπρόσθετες λειτουργίες όπως πλοήγηση ή βοήθεια.
- ✓ Πρότυπα για την διευκόλυνση της εφαρμογής του λογισμικού.
- ✓ Προσαρμογή για άτομα με ειδικές ανάγκες.

Η τεκμηρίωση των λεπτομεριεών των διεπαφών των χρηστών (User Interfaces) θα πρέπει να γίνεται σε ένα ξεχωριστό έγγραφο και όχι στην SRS.

***Hardware Interfaces.*** Περιγραφή των χαρακτηριστικών της κάθε επαφής μεταξύ του λογισμικού και του hardware του συστήματος. Περιλαμβάνει τόσο τις αλληλεπιδράσεις μεταξύ τους, όσο και τα πρωτόκολλα επικοινωνίας που πρέπει να χρησιμοποιούνται.

***Software Interfaces.*** Περιγραφή των συνδέσεων μεταξύ αυτού και των άλλων στοιχείων του λογισμικού, συμπεριλαμβανομένων των βάσεων δεδομένων, των λειτουργικών συστημάτων και των εμπορικών εργαλείων. Γίνεται δήλωση των μηνυμάτων που ανταλλάσσονται μεταξύ τους, αλλά καταγράφεται και ο σκοπός των μηνυμάτων. Περιγράφονται οι υπηρεσίες που προσφέρει το κάθε λογισμικό.

***Communications Interfaces.*** Περιγράφονται οι απαιτήσεις για κάθε λειτουργία επικοινωνίας που θα εκτελέσει το σύστημα συμπεριλμβανομένων των e-mail, του webBrowser και των πρωτοκόλλων δικτύων και επικοινωνιών. Επίσης καθορίζονται τα θέματα ασφάλειας της επικοινωνίας όπως η κρυπτογράφηση, η ταχύτητα μεταφοράς των δεδομένων και οι μηχανισμοί συγχρονισμού.

- **Other Nonfunctional Requirements ( Υπόλοιπες μη Λειτουργικές Απαιτήσεις).**

Καθορίζονται οι μη λειτουργικές απαιτήσεις, εκτός των απαιτήσεων εξωτερικής διεπαφής (externalinterfacerequirements) και των περιορισμών που περιγράφθηκαν παραπάνω.

***Performance Requirements (Απαιτήσεις Απόδοσης).*** Καταγράφονται οι λειτουργίες που έχουν να κάνουν με την απόδοση των διάφορων λειτουργιών του συστήματος. Πρέπει να εξηγηθεί η σκέψη πίσω από κάθε απαίτηση για να πάρουν οι σχεδιαστές την κατάλληλη σχεδιαστική επιλογή. Να δηλωθούν δηλαδοί οι χρόνοι απόκρισης της βάσης δεδομένων, τον αριθμό των συναλλαγών ανά δευτερόλεπτο, ο καθορισμός της μνήμης. Είναι καλό να καθορίζονται οι απαιτήσεις απόδοσης στις συγκεκριμένες λειτουργικές απαιτήσεις που αναφέρονται. Καλό είναι να γίνεται και ποσοτικοποίηση των απαιτήσεων απόδοσης.

***Safety Requirements (Απαιτήσεις Ασφάλειας).*** Ορίζονται εγγυήσεις ή δράσεις που πρέπει να γίνουν και να προβλευθούν επικύνδυνες ενέργειες. Πρέπει να καταγραφούν όλες οι πιστοποιήσεις, οι πολιτικές και οι κανονισμοί ασφαλείας με τους οποίου πρέπει να συμμορφωθεί το σύστημα.

***Security Requirements.*** Εδώ ορίζονται όλες οι απαιτήσεις που έχουν να κάνουν με την ακεραιότητα και την διασφάλιση των στοιχείων των πελατών του συστήματος. Προέρχονται από επιχειρηματικούς κανόνες, πολιτικές ασφάλειας και προστασίας των στοιχείων των πελατών και από κανονιστικές διατάξεις με τις οποίες πρέπει να συμμορφώνεται το σύστημα.

***Software Quality Attributes (Χαρακτηριστικά Ποιότητας του Λογισμικού).*** Καταγραφή των χαρακτηριστικών ποιότητας που είναι χρήσιμα στους πελάτες και θα βοηθήσουν τους σχεδιαστές. Πρέπει να είναι συγκεκριμένα, ποσοτικά και επαληθεύσιμα. Πρέπει να γίνεται και αναφορά των προτεραιοτήτων των διάφορων χαρακτηριστικών όπως η ευκολία χρήσης και η ευκολία μεταφοράς.

- **Other Requirements**

Εδώ ορίζονται οι απαιτήσεις που δεν καλύπτονται από την SRS. Τέτοιες είναι οι απαιτήσεις διεθνοποίησης (internationalizationrequirements), που έχουν να κάνουν με το νόμισμα, την γλώσσα, τους διεθνείς κανονισμούς, τα πολιτιστικά και πολιτικά ζητήματα και οι νομικές απαιτήσεις. Μπορούν επίσης να φιλοξενηθούν κατηγορίες που έχουν να κάνουν με την

διαχείρηση, την συντήρηση, την παραμετροποίηση και την καταργραφή και παρακολούθηση των διεργασιών. Μπορούμε να παραλείψουμε την συγκεκριμένη κατηγορία αν οι απαιτήσεις καλύπτονται από άλλα τμήματα του Εγγράφου Προσδιορισμού Απαιτήσεων.

Όπως αναφέραμε και παραπάνω είναι καλό στοΕγγράφου Προσδιορισμού Απαιτήσεων να υπάρχουν και μερικά παραρτήματα τα οποία θα έχουν να κάνουν με:

**Appendix A: Glossary (Παράρτημα Α: Λεξικό - Γλωσσάρι).**
Εδώ ορίζονται οι ειδικευμένοι τους οποίους ο αναγνώστης θα πρέπει να ερμηνεύσει σωστά. θα πρέπει να οριστούν και να ερμηνευτούν τα ακρωνύμια που χρησιμοποιούνται. Το Έγγραφο Προδιαγραφής Απαιτήσεων θα πρέπει να συμπεριλαμβάνει όρους που έχουν να κάνουν αποκλειστικά με το έργο που υλοποιείται.

**Appendix B: Analysis Models.**
Είναι μία προαιρετική ενότητα που περιλαμβάνει τα μοντέλα ανάλυσης δεδομένων όπως τα διαγράμματα ροής (ΔΡΔ), τα διαγράμματα κλάσεων και τα διαγράμματα οντοτήτων συσχετίσεων.

**Appendix C: Issues List.**
Περιλαμβάνονται τα κάποια ανοιχτά ζητήματα που πρέπει να επιλυθούν. Τέτοια ζητήματα μπορεί να είναι τα στοιχεία που έχουν χαρακτηριστεί ως TBD. Το παράρτημα αυτό δεν είναι απαράιτητο να συμπεριλαμβάνεται στην SRS, αλλά είναι καλό να επισυνάπτεται μαζί της για να γίνεται έγκαιρη διαχείρηση των ζητημάτων που πρέπει να αντιμετωπιστούν.
Δεν υπάρχει κάποιος σωστός τρόπος τεκμηρίωσης των απαιτήσεων. Είναι όμως καλό να χρησιμοποιείται η ορολογία των χρηστών και όχι η υπολογιστική διάλεκτος. Παρακάτω υπάρχουν κάποιες προτάσεις για σωστότερη τεκμηρίωση απαιτήσεων:

- Θα πρέπει να υπάρχουν ολοκληρωμένες προτάσεις με σωστή γραμματική, ορθογραφία και στίξη.
- Οι παράγραφοι και οι προτάσεις θα πρέπει να είναι περιεκτικές και σύντομες.
- Θα πρέπει να χρησιμοποιείται ενεργητική φωνή.
- Θα πρέπει οι όροι να χρησιμοποιούνται όπως ορίζονται στο λεξικό – γλωσσάρι.
- Θα πρέπει μία υψηλού επιπέδου αόριστη απαίτηση να διευκρινίζεται λεπτομερώς για να μην υπάρχουν ασάφειες.
- Θα πρέπει να χρησιμοποιούνται λίστες, πίνακες και διγράμματα για να εμφανίζονται και οπτικά οι απαιτήσεις έτσι ώστε να γίνονται ευκολότερα κατανοητές και διαχειρίσιμες.

Οι απαιτήσεις πρέπει να τεκμηριώνονται με ακρίβεια και λεπτομερώς έτσι ώστε ο σχεδιαστής να κατανοεί πλήρως τις ανάγκες του πελάτη. Επίσης θα πρέπει οι απαιτήσεις που γράφονται να είναι ελέγξιμες. Όσο λιγότερους ελέγχους θα πρέπει να διενεργήσουμε για να βεβαιοθούμε για

την ορθότητα μίας απαίτησης, τόσο πιο καλή αυτή είναι. Οι ελέγξιμες απαιτήσεις έχουν προταθεί ως μέτρο για το μέγεθος του προϊόντος λογισμικού.

Θα πρέπει αν τηρείται το ίδιο επίπεδο λεπτομέρειας και να αποφεύγονται οι μεγάλες παράγραφοι που περιέχουν πολλές απαιτήσεις. Επίσης θα πρέπει να αποφεύγεται πολλαπλή δήλωση των ίδιων απαιτήσεων. Είναι προτιμότερο να γίνεται αναφορά σε άλλα σημεία της SRS, παρά να επαναλαμβάνονται τα ίδια κάθε φορά πράγματα. Η αποθήκευση των απαιτήσεων σε ένα εργαλείο διαχείρησης αποτρέπει την πολλαπλή και άσκοπη επαναχρησιμοποίησή τους [13,21,45,46]

### 3.3.2 Λεξικό Δεδομένων

Τα προβλήματα ολοκλήρωσης του έργου θα μειώνονταν αν η ομάδα ανάπτυξης λειτουργούσε σύμφωνα με τους ορισμούς στο λεξικό των δεδομένων. Το λεξικό αυτό συμπληρώνει το γλωσσάρι του έργου, που καθορίζει τους ατομικούς όρους που χρησιμοποιούνται. Μπορεί να υπάρχει είτε σαν ένα κομμάτι του SRS, είτε σαν ένα χωριστό έγγραφο.
Ένα ξεχωριστό λεξικό όρων βέβαια θα έκανε το SRS μικρότερο και πιο ευανάγνωστο αφού δεν θα περιλάμβανε πολλές λεπτομέρειες και επεξηγήσεις, αλλά θα καθιστούσε και πιο έυκολη την αναζήτηση της πληροφορίας που χρειαζόμαστε. Τα στοιχεία που θα υπάρχουν σε αυτό θα πρέπει παρουσιάζονται με απλούς συμβολισμούς. Υπάρχουν διάφορες κατηγορίες δεδομένων σε ένα λεξικό.
Τα πρωτογενή στοιχεία δεδομένων (Primitive Data Elements). Είναι τα δεδομένα για τα οποία δεν χρειάζεται περαιτέρω ανάλυση. Ο ορισμός που τους δίνεται προσδιορίζει τον τύπο δεδομένων, το μέγεθος και τις επιτρεπόμενες τιμές που μπορούν να πάρουν. Συνήθως καθορίζονται και με ενα σχόλιο το οποίο οριοθετήται με αστερίσκους.

**Request ID = * a 6-digit system-generated sequential integer,**
**beginning with 1, that uniquely identifies each**
**request ***

**Συνθεση (Composition)**. Είναι μία δομή που πειλαμβάνει πολλά στοιχεία δεδομένων. Αν ένα στοιχεία είναι προεραιτικό τότε περικλείεται σε παρενθέσεις.
***RequestedChemical = ChemicalID***
***+ Number of Containers***
***+ Grade***
***+ Amount***
***+ Amount Units***
***+ (Vendor)***

**Επανάληψη (Interation).** Αν ένα στοιχείο εμφανίζεται πολλές φορές σε μία δομή δεδομένων, τότε θα πρέπει να εμφανίζεται μέσα σε άγκυστρα {}. Ο αριθμός των επανλήψεων θα είναι της μορφής Κατώτατο_Όριο : Ανώτατο_Όριο πριν από τα άγκυστρα.

***Request = Request ID***
***+ Request Date***
***+ Charge Number***
***+ 1:10{RequestedChemical}***

**Επιλογή (Selection).**
Υπάρχουν στοιχεία δεδομένων που μπορούν να πάρουν διαφορετικές τιμές. Οι τιμές αυτές εμφανίζονται σε “” και διαχωρίζονται από κάθετες μπάρες ||.
***Quantity Units = ["grams" | "kilograms" | "milligrams" | "each"]***
**** 9-character text string indicating the units associated with the quantity of chemical requested ****

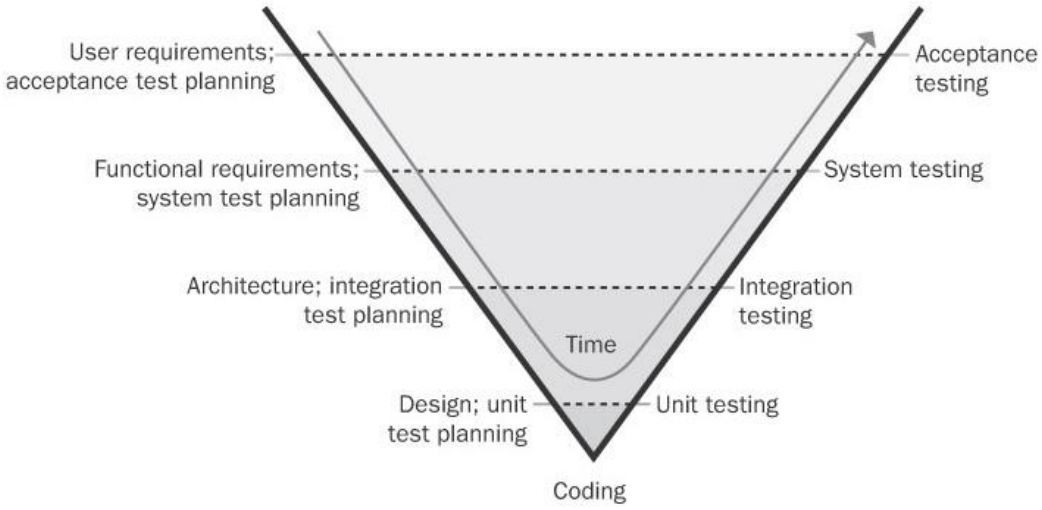
Ο χρόνος που αφιερώνεται στην δημιουργία ενός λεξικού δεδομένων είναι στην ουσία μία επένδυση που μειώσνει τον κίνδυνο παρερμηνείας των απαιτήσεων. Ένα σωστό λεξικό δεδομένων είναι ένα πολύτιμο εργαλείο που εξασφαλίζει τόσο την σωστή ανάπτυξη, όσο και την σωστή συντήρηση του έργου [10,11,21].

## 3.4 Επικύρωση Των Απαιτήσεων

Θα πρέπει όταν δίνονται οι απαιτήσεις στους προγραμματιστές του λογισμικού να έιναι σαφείς πλήρεις, διαφορετικά θα αναγκαστούν να τις ερμηνεύσουν από την δική τους οπτική γωνία η οποία δεν θα είναι απαραίτητα και η σωστή. Η προσπάθεια να διορθωθούν τα λάθη μετά την εκπλήρωση των εργασιών του έργου είναι πολύ μεγαλύτερη από την διόρθωση κατά την διάρκεια της ανάπτυξής του.

Σε πολλά έργα ο έλεγχος είναι μία από τις τελευταίες δραστηριότητες που γίνεται και έτσι τα προβλήματα παραμένουν ως έχουν μέχρι το τέλος. Εάν όμως ο έλεγχος γίνει από την αρχή της ανάπτυξης, τότε τα λάθη θα προσδιοριστούν και θα επιλυθούν νωρίς χωρίς μεγάλο κόστος [9].

Το Σχήμα 46 δείχνει το μοντέλο ανάπτυξης λογισμικού **V**, στο οποίο οι δοκιμές ελέγχου αρχίζουν παράλληλα με τις δραστηριότητες ανάπτυξης. Το μοντέλο αυτό δείχνει ότι οι δοκιμές βασίζονται στις απαιτήσεις των χρηστών, ο έλεγχος του συστήματος βασίζεται στις λειτουργικές απαιτήσεις και οι δοκιμές ολοκλήρωσης βασίζονται στην αρχιτεκτονική του συστήματος.




**Σχήμα 46.**Το μοντέλο ανάπτυξης λογισμικού V ενσωματώνει από νωρίς τις δοκιμές ελέγχου του συστήματος

Η επικύρωση είναι η τέταρτη συνιστώσα της ανάπτυξης των απαιτήσεων, μετά την εκμαίευση, την ανάλυση και τις προδιαγραφές. Οι δραστηριότητες της επικύρωσης των απαιτήσεων προσπαθούν να διασφαλίσουν ότι:

- Το Έγγραφο Προσδιορισμού Απαιτήσεων περιγράφει σωστά τις προβλεπόμενες δυνατότητες του συστήματος και τα χαρακτηριστικά που θα ικανοποιήσουν τις ανάγκες των διαφόρων ενδιαφερομένων.
- Οι απαιτήσεις του λογισμικού προέρχονται από τις απαιτήσεις του συστήματος απαιτήσειςκαι τους επιχειρισιακούς κανόνες.
- Οι απαιτήσεις είναι πλήρεις και υψηλής ποιότητας.
- Όλες οι απαιτήσεις είναι συνεπείς μεταξύ τους.
- Οι απαιτήσεις παρέχουν μία επαρκής βάση για να προχωρήσει το σχεδιασμός και η κατασκευή.

Η διαδικασία αυτή εξασφαλίζει ότι οι απαιτήσεις παρουσιάζουν όλα τα επιθυμητά σε αυτές χαρακτηριστικά όπως η πληρότητα, η ακρίβεια, η σαφήνεια, η επαληθευσιμότητα, το ότι είναι απαραίτητες, ότι έχουν ιεραρχηθεί με προτεραιότητες, ότι είναι τροποποιήσιμες αλλά και διασυνδεδεμένες μεταξύ τους με μονοπάτια ιχνηλασιμότητας. Για να συμβεί όμως η επικύρωση θα πρέπει να έχει προηγηθεί η διαδικασία της τεκμηρίωσης.

Συνήθως οι ενδιαφερόμενοι είναι απρόθυμοι να αφιερώσουν χρόνο στην διαδικασία της επικύρωσης των απαιτήσεων νομίζοντας πώς απλώς σπαταλούν πολύτιμο χρόνο και επιβραδύνουν την παράδοση του έργου. Στην ουσία όμως αυτή η διαδικασία είναι μία επένδυση που τελικά θα μειώσει το χρονοδιάγραμμα της παράδοσης αφού μειώνει την άσκοπη δουλειά που θα πρέπει να γίνει εφόσον οι απαιτήσεις δεν είναι οι κατάλληλες για την υλοποίηση. Η

πρόληψη των ελαττωμάτων των απαιτήσεων είναι στην ουσία η καλύτερη αντιμετώπισή τους. Οι σωστές απαιτήσεις οδηγούν σε προϊόντα υψηλότερης ποιότητας, μεγαλύτερη ικανοποίηση του πελάτη και μείωση του κόστους του προϊόντος.

Υπάρχουν διάφορες τεχνικές που μας βοηθούν να αξιολογήσουμε την ορθότητα και την ποιότητα των απαιτήσεων ενός συστήματος. Η μία προσέγγιση είναι να ποσοτικοποιηθεί η κάθε απαίτηση έτσι ώστε να είμαστε σε θέση να μπορούμε να μετρήσουμε σε τι ποσοστό μπορεί να ικανοποιηθεί μία προτεινόμενη λύση [5,10,21].

### 3.4.1 Επιθεώρηση Των Απαιτήσεων

Μία τεχνική της επικύρωσης είναι η επανεξέταση του συνόλου των απαιτήσεων έτσι ώστε να βρεθούν ποιες από αυτές είναι ασαφείς, διφορούμενες και θα δημιουργήσουν προβλήματα στην διαδικασία του σχεδιασμού και της υλοποίησης. Υπάρχουν οι επίσημες και οι ανεπίσημες επανεξετάσεις.Η ανεπίσημη μπορεί να γίνει με τρεις τρόπους:

- Να ζητηθεί από κάποιον συνάδελφο να επανεξετάσει το την εργασία που έχει ήδη γίνει (Peer Deskcheck).
- Να οριστεί μία ομάδα συναδέλφων να εξετάσει ένα παραδοτέο του συστήματος (Passaround).
- Να περιγράψει ο αναλυτής το παραδοτέο και ζητήσει παρατηρήσεις σχετικά με αυτό. Η μέθοδος αυτή ονομάζεται περιήγηση (Walkthrough).

Στα θετικά αυτής της διαδικασίας είναι δεν χρειάζεται κάποια ειδική προετοιμασία και μπορεί να ολοκληρωθεί γρήγορα.

Η επίσημη επανεξέταση από την άλλη, ακολουθεί μία συγκεκριμένη διαδικασία. Δημιουργείται μία έκθεση που προδιορίζει το υλικό, τους επιθεωρητές και τις αποφάσεις για το αν τελικά το προϊόν είναι αποδεκτό. Το κύριο παραδοτέο είναι μία περίληψη των ελαττωμάτων στα ζητήματα που τέθηκαν.

Ο πιο ενδεδειγμένος τρόπος για να γίνει μία επίσημη επανεξέταση είναι η επιθεώρηση (inspection). Ο καλύτερος τρόπος είναι να επιθεωρηθεί ξεχωριστά το κάθε έγγραφο απαιτήσεων που δημιουργείται. Εάν δεν υπάρχει χρόνος να επιθεωρηθούν όλα, γιατί είναι μία αρκετά χρονοβόρα διαδικασία, θα πρέπει να γίνει ανάλυση κινδύνου και να διαχωριστούν οι απαιτήσεις που χρειάζονται επιθεώρηση από εκείνες που είναι λιγότερο κρίσιμες και τους αρκεί μία άτυπη επανεξέταση.

***Ομάδα Επιθεώρησης***

Συγκροτείται μία ομάδα συμμετεχόντων που θα εξετάσουν προσεκτικά ένα προϊόν εργασίας για να βρουν ελαττώματα και δυαντότητες βελτίωσης. Οι συμμετέχοντες στην διαδικασία είναι ο μεσολαβητής (Moderator) ο οποίος ηγείται της επιθεώρησης και συντονίζει τις απαράιτητες ενέργεις, ο συντάκτης του προϊόντος και οι επιθεωρητές που μαζί με τον συντάκτη αναλαμβάνουν τον εντοπισμό των σφαλμάτων. Αναλυτικά οι ρόλοι των εμπλεκομένων είναι:

**Συντάκτης.** Ο συντάκτης είναι ο δημιουργός των απαιτήσεων που επιθεωρούνται. Οδηγεί την συζήτηση αλλά έχει έναν παθητικό ρόλο. Δουλειά του είναι να ακούει τα σχόλια των υπόλοιπων επιθεωρητών και να απαντά στις ερωτήσεις τους.

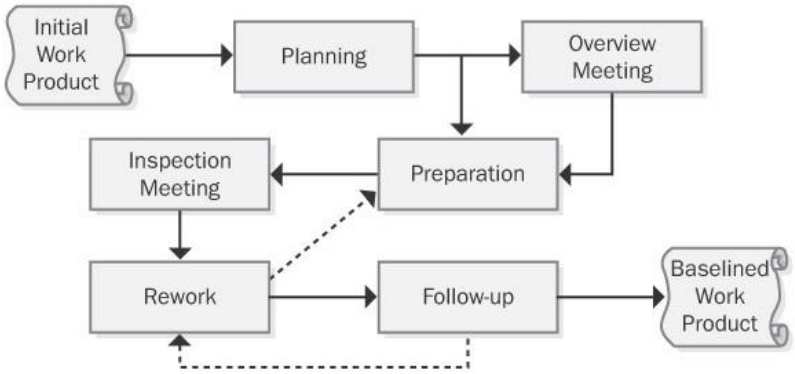
**Μεσολαβητής- Συντονιστής (Moderator).** Είναι εκείνος που συντονίζει τις δραστηριότητες της επιθεώρησης. Διανέμει το υλικό που θα επιθεωρηθεί και συντονίζει την συζήτηση.

**Αναγνώστης.** Εκφράζει με δικά του λόγια τις απαιτήσεις της SRS. Οι υπόλοιποι τον ακούνε και δίνουν την δική τους ερμηνεία έτσι ώστε να επισημανθούν πιθανές ατέλειες και προβλήματα.

**Εκείνος που καταγράφει.** Χρησιμοποιεί τυποποιημένα έντυπα για την καταγραφή των ζητημάτων που θίχτηκαν και τα ελαττωμάτων που βρέθηκαν κατά τη συνεδρίαση. Θα πρέπει να διαβάσει φωναχτά αυτό που έγραψε για να επιβεβαιωθεί η ακρίβεια της εγγραφής.

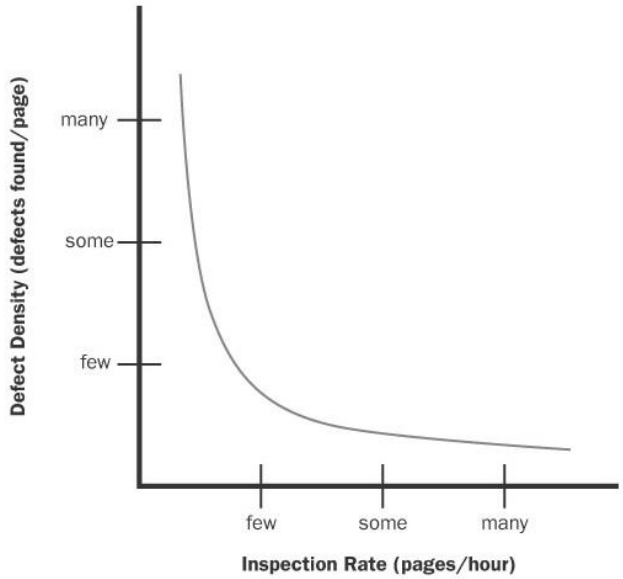
***Βήματα επιθεώρησης***

Όπως φαίνεται στο Σχήμα 47 επιθεώρηση είναι μία διαδικασία με πολλά βήματα. Το κάθε βήμα θα αναλυθεί στην συνέχεια.




**Σχήμα 47.** Τα βήματα της επιθεώρησης. Οι διακεκομμένες γραμμές δείχνουν τα βήματα της διαδικασίας ελέγχου που μπορούν να επαναληφθούν

**Σχεδιασμός.** Αφορά τους στόχους της επιυεώρησης, τον καθορισμό των προσώπων που θα παίξουν τον ρόλο των επιθεωρητών, τις τεχνικές που θα ακολουθηθούν και το χρονοδιάγραμμα της επιθεώρησης. Όσο περισσότερο χρόνο δίνουμε στην ανάγνωση κάθε σελίδας της SRS, τόσο περισσότερα σφάλματα θα ανιχνευθούν. Παρακάτω, στο Σχήμα 48, φαίνεται διαγραμματικά η συσχέτιση του χρόνου ελέγχου που αφιερώνεται με τα σφάλματα που εντοπίζονται.




**Σχήμα 48.** Αριθμός εντοπισμένων σφαλμάτων σε σχέση με τον χρόνο επιθεώρησης που αφιερώνεται

**Συνάντηση επισκόπησης.** Είναι μία κοινή συνεδρίαση στην οποία ο συντάκτης των απαιτήσεων παρουσιάζει στην ομάδα το περιέχομενο του προϊόντος προς συζήτηση, έτσι ώστε να αποκτήσουν όλοι την απαραίτητη οικειότητα με αυτό.
**Προετοιμασία.** Ο κάθε επιθεωρητής μελετά τις απαιτήσεις με στόνο την αναζήτηση σφαλμάτων. Τα σφάλματα αυτά μπορεί να είναι σφάλματα στις απαιτήσεις, λάθη στην σχεδίαση ή στον κώδικα.
**Συνεδρίαση επιθεώρησης.** Στην συνεδρίαση αυτή συλλέγονται τα σφάλματα που έχουν εντοπιστεί από κάθε επιθεωρητή και συζητούνται στην ομάδα. Η συνεδρίαση καταλήγει με έναν κατάλογο θεμάτων τα οποία θα πρέπει να διορθώσει ο συντάκτης των απαιτήσεων.
**Διόρθωση.** Ο συντάκτης προβαίνει στις κατάλληλες ενέργειες για τη διόρθωση των σφαλμάτων.
**Κλείσιμο.** Ο κύκλος της επιθεώρησης ολοκληρώνεται με μία τελική επαλήθευση. Είναι αποτέλεσμα συνεργασίας του μεσολαβητή με τον συντάκτη για να επιβεβαιώσει ότι έχουν γίνει όλες οι απαραίτητες αλλαγές για τη διόρθωση του προϊόντος.
Για να είναι επιτυχής η επιθεώρηση θα πρέπει να τηρούνται κάποια κριτήρια τόσο στην αρχή όσο και στο τέλος της διαδικασίας. Αρχικά είναι καλό να δημιουργείται μία λίστα με κάποια κριτήρια ελέγχου του εγγράφου των απαιτήσεων που θα πρέπει να ισχύουν πριν αρχίσει η όλη διαδικασία. Μερικά από τα κριτήρια αυτά μπορεί να είναι:

- Το έγγραφο είναι σύμφωνο με το καθορισμένο πρότυπο.
- Έχει ορθογραφικό έλεγχο.
- Είναι ευανάγνωστο.
- Όλες οι αμφισβητούμενες απαιτήσεις έχουν χαρακτηριστεί ως TBD.
- Υπάρχει αρίθμηση ώστε να διευκολύνονται οι αναφορές που γίνονται μέσα στην SRS.

Αντίστοιχα θα πρέπει κατά το πέρας της διαδικασίας να ισχύουν οι παρακάτω προϋποθέσεις:

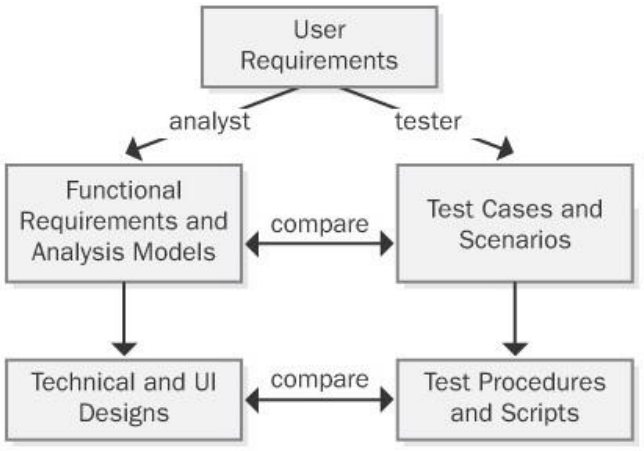
- Έχουν αντιμετωπιστεί όλα τα ζητήματα που τέθηκαν κατά τη διάρκεια της επιθεώρησης.
- Οι οποιεσδήποτε αλλαγές έγιναν με σωστό τρόπο.
- Όλες οι TBD απαιτήσεις έχουν επιλυθεί, καθοριστεί και τεκμηριωθεί.

Όπως αναφέρθηκε και παραπάνω η επιθεώρηση είναι μία επίπονη και χρονοβόρα διαδικασία. Θα πρέπει όμως να γίνει όσο το δυνατόν καλύτερα για να επιτευχθεί η ορθότητα του συστήματος προς υλοποίηση. Θα πρέπει να δοθεί ιδιαίτερη προσοχή όταν έχουμε να κάνουμε με μεγάλες SRS. Στην περίπτωση αυτή θα πρέπει να γίνει αναλύση κινδύνου στις απαιτήσεις και να επιθεωρηθούν σωστά τουλάχιστον οι σημαντικότερες. Επίσης καλό θα είναι να δημιουργηθούν πολλές ομάδες επιθεώρησης που σαρώνουν παράλληλα τις απαιτήσεις έτσι ώστε να εξοικονομηθεί πολύτιμος χρόνος. Η ομάδα επιθεώρησης θα πρέπει να απαρτίζεται από κατάλληλα άτομα και να είναι μικρή και ευέλικτη στις συναντήσεις της. Τέλος στα έργα που γίνονται σε διαφορετικές γεωγραφικές τοποθεσίες, η επιθεώρηση θα πρέπει να γίνεται με ακόμα μεγαλύτερη προσοχή. Καλό να είναι να υπάρχει οπτική επικοινωνία της ομάδας συμμετεχόντων και όχι απλά αποστολή τηλεφωνικών και διαδυκτιακών μυνημάτων [47,48].

### 3.4.2 Έλεγχος Των Απαιτήσεων

Είναι πολύ δύσκολο να απεικονικστεί πώς θα λειτουργεί ένα σύστημα κάτω υπό ορισμένες συνθήκες μόνο από την ανάγνωση της SRS. Για τον λόγο αυτό θα πρέπει να γίνουν δικομές ελέγχου στις απαιτήσεις, είτε είναι λειτουργικές είτε είναι του χρήστη, έτσι ώστε να δούμε την συμπεριφορά του συστήματος. Οι περιπτώσεις ελέγχου θα πρέπει να ξεκινούν μόλις σταθεροποιηθούν οι απαιτήσεις για να μπορέσουν να διορθωθούν χωρίς μεγάλο κόστος τα τυχόν προβλήματα. Οι δοκιμές αυτές μπορούν να αξιολογήσουν τόσο τις απαιτήσεις, όσο τα μοντέλα ανάλυσης και τα πρωτότυπα.Μπορούν να καλύψουν και τον κανονικό και τον εναλλακτικό δρόμο διεργασιών των σεναρίων χρήσης.

Ο αναλυτής θα γράψει τις λειτουργικές απαιτήσεις και η αμάδα ανάπτυξης δοκιμών ελέγχου θα δημιουργήσει περίπτωσεις δοκιμών για αυτές. Αντίστοιχα θα γίνει και για τις περιπτώσεις χρήστη, όπως φαίνεται στο Σχήμα 49. Όσο οι προγραμματιστές θα μετατρέπουν τις απαιτήσεις σε διεπαφές χρήστη και σε τεχνικό σχεδιασμό, θα μπορέσουν να γίνουν και λεπτομερείς διαδικασίες δοκιμών για τον τελικό έλεγχο του συστήματος.




**Σχήμα 49**.Διαδικασία Δοκιμών Ελέγχου Απαιτήσεων

***Στάδια δοκιμών***

Δοκιμές συστατικών στοιχείων (ή υπομονάδων)

- Κάθε συστατικό στοιχείο δοκιμάζεται ανεξάρτητα.
- Τα στοιχεία μπορεί να είναι συναρτήσεις, αντικείμενα ή συναφείς ομαδοποιήσεις τέτοιων οντοτήτων.

Δοκιμές συστήματος

- Δοκιμή του συστήματος στο σύνολό του. Είναι ιδιαίτερα σημαντική η δοκιμή των ανακυπτουσών ιδιοτήτων.

Δοκιμές αποδοχής

- Δοκιμή με δεδομένα που παρέχονται από τον πελάτη για να ελεγχθεί ότι το σύστημα πληροί τις ανάγκες του.


**Σχήμα 50.** Διάγραμμα διεργασιών ελέγχου και αποδοχής

Ανεξάρτητα από το επίπεδο ή τη φάση της διαδικασίας ανάπτυξης, η διαδικασία δοκιμών ή ελέγχων των προϊόντων αποτελείται από τρία συστατικά: τη δημιουργία μίας διαδικασίας

ελέγχου ή δοκιμών (που περιλαμβάνει συχνά τη δημιουργία των τύπων των δοκιμών), τη δημιουργία ενός σχεδίου δοκιμής (που περιλαμβάνει τις διαδικασίες για τη διενέργεια των δοκιμών - ελέγχων) και την εκτέλεση των ελέγχων - δοκιμών.

Η διαδικασία ελέγχου - δοκιμών είναι μια επίσημη περιγραφή για το πώς ένα προϊόν θα εξεταστεί. Για τη δημιουργία μιας τέτοιας διαδικασίας η ομάδα ελέγχου θα πρέπει να αναλύσει όλες τις απαιτήσεις του προϊόντος, να γράψει τη διαδικασία της δοκιμής και να εξετάσει την διαδικασία για τυχόν παραλείψεις. Το σχέδιο δοκιμής - ελέγχου μπορεί να συμπεριλάβει τις περιπτώσεις ελέγχων, τους όρους ελέγχων, το περιβάλλον δοκιμής, το κατάλογο των απαιτούμενων αποτελεσμάτων, τα κριτήρια επιτυχίας και αποτυχίας και κάποια διαχείριση κινδύνου.

Το σχέδιο δοκιμής ετοιμάζεται μετά την αναθεώρηση όλων των λειτουργικών απαιτήσεων του προϊόντος. Αυτές οι απαιτήσεις μπορούν να χωριστούν εύκολα σε συγκεκριμένες διαδικασίες δοκιμής. Οι διαδικασίες δοκιμής μπορούν να καθορίσουν τους όρους δοκιμής, τα στοιχεία που χρησιμοποιούνται για τη δοκιμή και τα αναμενόμενα αποτελέσματα. Το σχέδιο δοκιμής πρέπει να συμπεριλάβει τις περιπτώσεις ή τα σενάρια διεξαγωγής των δοκιμών, τα οποία θα πρέπει να έχουν ως σκοπό να αντιπροσωπεύσουν τις χαρακτηριστικές και ακραίες καταστάσεις που μπορούν να εμφανιστούν κατά τη διάρκεια της ζωής του προϊόντος.

Η εκτέλεση των δοκιμών πραγματοποιείται συστηματικά σύμφωνα με τα έγραφα δοκιμών. Τα έγγραφα δοκιμών είναι τα αποτελέσματα της δημιουργίας της διαδικασίας της δοκιμών και της ανάπτυξης των σχεδίων δοκιμών. Κάθε φορά που εκτελείται μία δοκιμή θα πρέπει να καταγράφονται τα αποτελέσματα της δοκιμής σε ένα αρχείο το οποίο ονομάζεται ημερολόγιο δοκιμών. Όλα τα αποτελέσματα των δοκιμών που σημειώνονται στο ημερολόγιο δοκιμών θα πρέπει να αξιολογηθούν από μηχανικούς να συγκριθούν με τα κριτήρια επιτυχίας και αποτυχίας που έχουν σημειωθεί στο σχέδιο δοκιμών. Οποιαδήποτε ελαττώματα στη λειτουργία του προϊόντος θα πρέπει να καθοριστούν προτού το προϊόν προχωρήσει στην τεχνική υλοποίηση και κατασκευή του. Υπάρχουν περιπτώσεις όπου τα ελαττώματα δεν καθορίζονται δεδομένου ότι σημειώνονται ως χαμηλής σημασίας στην αξιολόγηση του κινδύνου που έχει καταγραφεί στα προηγούμενα Επίπεδα ανάπτυξης του προϊόντος. Αφού ολοκληρωθούν όλες οι δοκιμές θα πρέπει να ετοιμαστεί μια περίληψη των δοκιμών και των αποτελεσμάτων και αυτή να παραδοθεί στον διευθυντή του προγράμματος ανάπτυξης, στο διευθυντή του τμήματος ποιοτικού ελέγχου και στον προϊστάμενο της ομάδας ελέγχου. Όταν όλες οι δοκιμές που συμπεριλαμβάνονται στην περίληψη δοκιμής πιστοποιηθούν το προϊόν προχωρά στο επόμενο επίπεδο ανάπτυξης.

Οι τελικοί κριτές του συστήμος θα είναι οι χρήστες. Θα πρέπει να διεξάγουν δοκιμές για να καθορίσουν εάν το σύστημα πληροί τα κριτήρια αποδοχής. Τα κριτήρια αυτά αξιολογούν κατά πόσο το σύστημα πληροί τις τεκμηριωμένες απαιτήσεις του και αν είναι κατάλληλο για το λειτουργικό περιβάλλον που προορίζεται. Όσο πιο γρήγορα γίνουν οι δοκιμές αποδοχής, τόσο

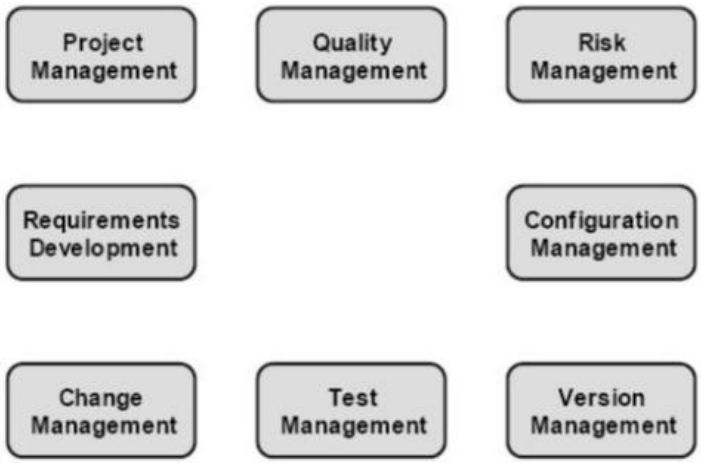
πιο γρήγορα θα διορθωθούν τα ελαττώματα των απαιτήσεων και τόσο πιο γρήγορα και εύκολα θα γίνει η υλοποίηση.

Οι δοκιμές αποδοχής θα πρέπει να αναφέρονται στα σενάρια χρήσης. Θα πρέπει να εξαταστούν τα σενάρια χρήσης και να αξιολογηθεί η αποδοχή του λογισμικού σε αυτά. Επικεντρώνονται τόσο στις κανονικές διαδρομές των περιπτώσεων χρήσης, όσο και στις εναλλακτικές και σε ειδικές συνθήκες χρήσης του συστήματος. Σε τέτοιες δοκιμές ελέγχου θα πρέπει να υπόκεινται και οι μη λειτουργικές απαιτήσεις για να διασφαλιστεί ότι έχουν επιτευχθεί οι στόχοι απόδοσης, επίδοσης και ποιότητας του προϊόντος και ότι έχουν εφαρμοστεί όλα τα πρότυπα και οι απαιτούμενοι περιορισμοί.

Είναι ακομα μία ευκαιρία να επικυρωθούν οι απαιτήσεις από τους χρήστες. Αν ο πελάτης δεν μπορεί να αξιολογήσει μία απαίτηση τότε αυτή δεν μπορεί να τεκμηριωθεί με σαφήνεια. Θα πρέπει, λοιπόν, να βεβαιωθούμε πως οι απαιτήσεις παρέχουν τις κατάλληλες προδιαγραφές για να χρησιμεύσουν ως θεμέλια σχεδιασμού, κατασκευής και διαχείρησης του έργου. Οι δοκιμές αποδοχής και οι επιθεωρήσεις των απαιτήσεων της SRS θα μας βοηθήσουν να οικοδομήσουμε ένα υψηλής ποιότητας σύστημα πιο γρήγορα και πιο οικονομικά [21,49,50].

# 4 Διαχείριση Απαιτήσεων

Τα μεγάλα και περίπλοκα έργα (projects) συνήθως διαιρούνται σε μικρότερα κομμάτια που το ένα ένα σχετίζεται με το άλλο. Μία τέτοια διαίρεση των κομματιών φαίνεται παρακάτω στο Σχήμα 51:



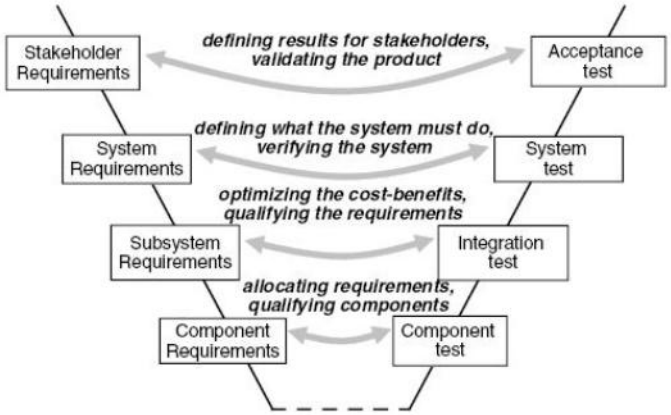

**Σχήμα 51.**Διαίρεση ενός έργου σε δρασηριότητες που σχετίζονται μεταξύ τους

Γενικά αυτός ο διαχωρισμός ονομάζεται Μηχανική των Απαιτήσεων (Requirements Engineering). Σκοπός της είναι να οδηγήσει το έργο να ολοκληρωθεί όσο τον δυνατόν πιο εύκολα και να μειώσει τα τυχόν λάθη. Είναι πιο αξιόπιστο και ευκολότερο να διαχειριστούμε ένα μεγάλο και σύνθετο έργο αφού προηγουμένως το έχουμε διαχωρίζει σε μικρότερα κομμάτια. Άνθρωποι με διαφορετικές εμπειρίες θα ανταποκρίνονται με διαφορετικές διαδικασίες και μεθόδους στις αντίστοιχες περιοχές.

Η Μηχανική των Απαιτήσεων είναι ένα σύνολο δραστηριοτήτων που λειτουργούν ως γέφυρα που ενώνει τις ανάγκες του συστήματος με τις ανάγκες των ενδιαφερόμενων μελών κατά τη διαδιακασία της ανάπτυξης της έργου, δίνοντας έμφαση στον εντοπισμό και την επικοινωνία των κομματιών του έργου που σταδιακά δημιουργούνται.

Στο κλασσικό μοντέλο **V**, του Σχήματος 52, φαίνονται οι σχέσεις μεταξύ των απαιτήσεων στα διάφορα στάδια ανάπτυξης του έργου. Η Μηχανική των Απαιτήσεων λαμβάνει χώρα σε κάθε επίπεδο. Αν και οι διαδικασίες που γίνονται διαφέρουν ελαφρώς σε κάθε επίπεδο, το βασικό

μοτίβο παραμένει το ίδιο. Η κύρια διαδικασία της Μηχανικής των Απαιτήσεων είναι ο ορισμόςτων απαιτήσεων, ο οποίος με την σειρά του χωρίζεται σε δύο υπορουτίνες, τον ορισμό του πεδίου εφαρμογής των απαιτήσεων και τον ορισμό των απαιτήσεων.




**Σχήμα 52**.Η Μηχανική των Απαιτήσεων σύμφωνα με το μοντέλο V

Η Διαχείρηση των Απαιτήσεων λειτουργεί σαν την συλλογή των συστημάτων διεργασιών που διασυνδέονται με την Μηχανική των Απαιτήσεων. Κατά την διαδικασία αυτή συγκεντρώνονται όλες οι αλλαγές και οι διαμορφώσεις που έχουν γίνει στις απαιτήσεις και την συνέχεια γίνονται οι κατάλληλεςς επικυρώσεις και επαληθεύσεις.

Η Διαχείρηση των απαιτήσεων, λοιπόν είναι ένα σύνολο δραστηριοτήτων που διασφαλίζουν πως τα χαρακτηριστικά των απαιτήσεων θα είναι πάντα ενημερωμένα και προσβάσιμα σε όποιον από την ομάδα θέλει να τα χρησιμοποιήσει. Με άλλα λόγια η Διαχείρηση των Απαιτήσεων ενσωματώνει τα σχετικά κομμάτια πληροφοριών από όλα τα συστημάτα της εφαρμοσμένης μηχανικής [11,28,51].

***Γιατί είναι σημαντική η Διαχείρηση των Απαιτήσεων;***

Η Μηχανική των Απαιτήσεων δεν περιλαμβάνει την αξιολόγηση των απαιτήσεων, αλλά "ενδιαφέρεται" μόνο για το πώς θα διατυπωθεί με απαιτήσεις το όραμα των ενδιαφερόμενων μελών που βάζουν το κεφάλαιο. Ενδιαφέρεται δηλαδή για το αν μία απαίτηση θα έχει μεγάλο κόστος ή όχι και για το αν μία απαίτηση θα έχει νόημα ή όχι. Έτσι κατά τη διάρκεια της ανάπτυξης του έργου μπορεί να δημιουργηθούν αρκετά πρόβλημα, εάν δεν συνοπολογιστεί η διαδικασία της Διαχείρησης της Απαιτήσεων. Τέτοια προβλήματα είναι τα παρακάτω:

- Οι απαιτήσεις να μην είναι εφικτές.
- Οι απαιτήσεις να μην είναι ελέγξιμες.
- Οι απαιτήσεις θα πρέπει να αξιολογούνται κάθε φορά που ένα εμπλεκόμενο μέλος προτείνει μία αλλαγή χωρίς αυτή να έχει τεκμηριωθεί.

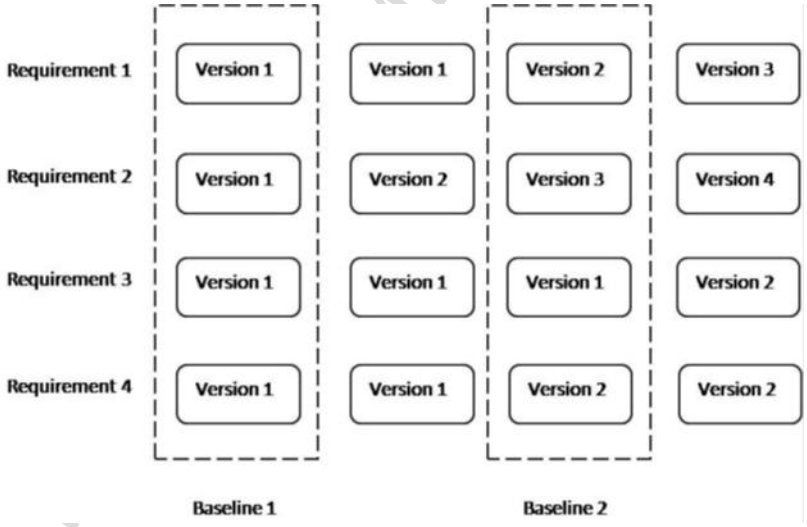
- Δεν είναι εφικτό να δωθεί η διαφορά μεταξύ των παλιών και νέων εκδόσεων των απαιτήσεων.
- Δεν μπορούμε να καταλάβουμε την βελτίωση των ενημερωμένων (Updated) πληροφοριών χωρίς να υπάρχει τεκμηρίωση.

Η Διαχείρηση των Απαιτήσεων έρχεται να δώσει λύση σε αυτά τα προβλήματα, να μειώσει το συνολικό ρίσκο και την πολυπλοκότητα του έργου και οργανώσει ακόμα περισσότερο την όλη διαδιακασία. Μερικά από τα κοινά κριτήρια Διαχείρησης απαιτήσεων δίνονται παρακάτω:

- **Identifiability** (Ικανότητα αναγνωρισιμότητας).
- **Filterability** (Ικανότητα να φιλτράρωνται).
- **Traceability**(Ικανότητα να ανιχνεύονται- Ιχνηλασιμότητα).
- **Linking** (Ικανότητα διασύνδεσης).
- **Userrights** (Δικαιώματα χρηστών).
- **Baselining** (Γραμμές βάσης).

**Identifiability (Ικανότητα Αναγνωρισιμότητας).** Κάθε απαίτηση θα πρέπει να είναι σε θέση να αναγνωριστεί μοναδικά. Αυτό επιτυχγάνεται δίνοντας σε κάθε απαίτηση έναν ξεχωριστό αριθμό. Προσθέτοντας ένα ξεχωριστό πρόθεμα σε κάθε ξεχωριστό χαρακτηριστικό, μας δίνεται η δυνατότητα να μπορούμε να ταυτοποιήσουμε κάθε απαίτηση.
**Filterability (Ικανότητα να Φιλτράρωνται).** Με την Διαχείρηση των Απαιτήσεων μαζεύουμε όλες τις πληροφορίες μαζί και πέρνουμε κάθε φορά όποια χρειαζόμαστε σε οποιαδήποτε στιγμή του έργου. Είναι σημαντικό όλοι να μοιράζονται τις πληροφορίες και όλοι να έχουν πρόσβαση σε αυτές.
**Traceability (Ικανότητα να Ανιχνεύονται - Ιχνηλασιμότητα).** Η ικανότητα να ανιχνεύονται καλύπτει δύο σημαντικές πτυχές. Η πρώτη είναι η δυνατότητα ανίχνευσης μεταξύ δύο διαφορετικών τύπων πληροφοριών την ίδια στιγμή αλλά και την δυνατότητα ανίχνευσης ενός κομματιού πληροφορίας κατά τη διάρκεια του χρόνου.
**Linking (Ικανότητα Διασύνδεσης).** Διασύνδεση είναι η τεκμηρίωση των σχέσεων μεταξύ διαφορετικών κομματιών πληροφορίας που σχετίζονται με τις απαιτήσεις. Για παράδειγμα η διασύνδεση των πληροφοριών των απαιτήσεων με τις πληροφορίες του ελέγχου (test) σημαίνει την τεκμηρίωση κατά κάποιο τρόπο για το ποιές απαιτήσεις θα ελεγχθούν, από ποιές περιπτώσεις δοκιμών (testcases) και ποιές περιπτώσεις δοκιμών καλύπτουν τις όποιες απαιτήσεις.
**User rights (Δικαιώματα Χρηστών).** Εφόσον όλα τα μέλη της ομάδας μοιράζονται τις ίδιες πληροφορίες, πρέπει να υπάρχουν κανόνες σχετικά με τη διαχείρηση της πληροφορίας. Οι υπεύθυνοι για τον έλεγχο δηλαδή θα πρέπει να έχουν πρόσβαση μόνο στο διάβασμα των απαιτήσεων. Ως εκ τούτου είναι σημαντικό να οριστούν κάποιες "γραμμές" πολιτικής σύμφωνα με τις οποίες τα δεδομένα θα μπορούν να τα δουν και να τα επεξεργαστούν μόνο τα εξουσιωδοτημένα μέλη της ομάδας, που θα οριστούν από τους παραπάνω κανόνες.

**Baselining (Γραμμές Βάσης).** Το Requirement Baseline είναι ένα σύνολο από λειτουργικές και μη λειτουργικές απαιτήσεις που η ομάδα ανάπτυξης έχει δεσμευτεί να εφαρμόσει σε κάποια συγκεκριμένη έκδοση. Ο ορισμός ενός Baseline επιτρέπει στους ενδιαφερόμενους να καταλάβουν τι ιδιότητες και δυνατότητες θα έχει το έργο όταν ολοκληρωθεί. Είναι σαν ένα στιγμιότυπο της συλλογής των απαιτήσεων σε μία συγκεκριμένη χρονική στιγμή. Όταν οι απαιτήσεις λοιπόν θα περάσουν την γραμμή αυτή υποτίθεται πως δεν θα οδηγηθούν σε περεταίρω αλλαγές και επεξεργασίες, αλλά θα ταυτιστούν με την έκδοση που υπάρχει εκείνη την στιγμή. Υπάρχει όμως η περίπτωση να δημιουργηθούν νέες εκδόσεις χωρίς να αλλάξουν τις Baselined εκδόσεις. Οι διαφορές μεταξύ των των διάφορων Baselines φαίνονται στις αναφορές των εκθέσεων όταν πραγματοποιούνται αξιολογήσεις. Στο Σχήμα 53 φαίνεται ένα παράδειγμα στο οποίο υπάρχουν τέσσερις απαιτήσεις με διάφορες εκδόσεις. Αρχικά όλες οι απαιτήσεις έχουν οριοθετηθεί με Baseline1 στην έκδοση Vesrion1 και η κάθε απαίτηση με την έκδοση Version1 δεν μπορεί να υποστεί επιπλέον αλλαγές. Όμως με τον καιρό έχουν δημιουργηθεί νέες εκδόσεις για όλες τις απαιτήσεις και στο σημείο που γίνεται το Basline2 η κάθε απαίτηση έχει νέα έκδοση εκτός από την απαίτηση Requirement3. Στο σημείο αυτό οι τρέχουσες απαιτήσεις με τις τρέχουσες εκδόσεις έχουν οριοθετηθεί σαν Baseline2. Το BaselinedSRS έγγραφο θα πρέπει να περιλαμβάνει μόνο τις απαιτήσεις που έχουν σχεδιαστεί για κάποια συγκεκριμένη έκδοση. Η αποθήκευσή τους σε ένα εργαλείο διαχείρησης απαιτήσεων διευκολύνει τον προσδιορισμό και τη διαχείρηση των αλλαγών που θα υποστούν σε αυτή τη βάση.




**Σχήμα 53.** Baselining με αναπτυσσόμενες απαιτήσεις

Είναι αρκετά συνηθισμένο πως η μικρή κατανόηση των απαιτήσεων των χρηστών οδηγεί πολλά προγραμμάτα λογισμικού στην αποτυχία. Πολλοί οργανισμοί ανάπτυξης λογισμικού βελτιώνουν τις μεθόδους τους που χρησιμοποιούν για την ανάλυση, την τεκμηρίωση και τη διαχείρηση των απαιτήσεων, έτσι ώστε να έχουν καλύτερα αποτελέσματα. Αυτοί οι οργανισμοί τεκμηριώνουν τις απαιτήσεις σε ένα δομημένο λογισμικό προδιαγραφών απαιτήσεων SRS

(Software Requirements Specification) βασισμένο σε φυσική γλώσσα. Παρόλα αυτά ένα τέτοιο SRS κείμενο έχει και κάποιους περιορισμούς.

### 4.1 Αστάθεια των απαιτήσεων

Οι ενδιαφερόμενοι είναι οι είναι οι κύριοι συντελεστές και δημιουργοί των απαιτήσεων σε ένα πληροφοριακό σύστημα ή σε ένα έργο γενικότερα. Στην αρχική φάση κάθε έργου οι απαιτήσεις μπορεί να αλλάξουν λόγω του γεγονότος ότι οι ενδιαφερόμενοι συνήθως έχουν διάφορες ανάγκες και στόχους οι οποίοι μπορεί να δημιουργήσουν ορισμένες συγκρούσεις με τις ήδη υπάρχουσες απαιτήσεις. Αυτές οι συγκρούσεις μπορεί να οδηγήσουν σε αλλαγές και λάθη που πρέπει να εκτιμηθούν. Όταν λοιπόν οι απαιτήσεις υποστούν αρκετές αλλαγές κατά την πάροδο του χρόνου, υπάρχει η πιθανότητα να γίνουν ασταθής.

Ένας ακόμα παράγοντας που οδηγεί στην αστάθεια των απαιτήσεων είναι πως καμμιά φορά τα ενδιαφερόμενα μέλη μπορεί να έχουν και αντίθετα μεταξύ τους συμφέροντα. Οι απαιτήσεις αυτές θα πρέπει να εξελιχθούν και αυτό αντανακλά και αλλαγές στο σύστημα ανάπτυξης. Επιπλέον η αστάθεια των απαιτήσεων εξαρτάται και από άλλους παράγοντες. Τέτοιοι παράγοντες θα μπορούσαν να είναι η οργανωτική πολυπλοκότητα, το στάδιο του κύκλου ζωής και η γενικότερη αστάθεια της αγοράς. Είναι λογικό ότι όσο πιο πολύπλοκο είναι ένα αναπτυσσόμενο σύστημα , τόσο μεγαλύτερη είναι και η αστάθεια εξ' αιτίας της αύξησης των αλληλεπιδρώντων συστατικών.

Το πλέον κατάλληλο για τον έλεγχο της αστάθειας των απαιτήσεων είναι το επαυξητικό οντέλο ανάπτυξης (Incremental Development). Όταν χρησιμοποιείται single increment είναι πολύ δύσκολο να αναβάλουμε τα αιτήματα των ενδιαφερομένων, τα οποία έχουν σαν αποτέλεσμα να δημιουργούν πολλές αλλαγές στο προϊόν και στο πρόγραμμα του έργου. Η Incremental ανάπτυξη όμως επιτρέπει στο πλάνο να είναι λιγότερο επιρρεπή σε αλλαγές παρόλα τα αιτήματα των ενδιαφερομένων. Οι απαιτήσεις βέβαια πάλι θα αλλάξουν αλλά με την μέθοδο αυτή θα γίνει πολύ ευκολότερο να διαχειριστούμε την αστάθεια των απαιτήσεων και να τις κρατήσουμε όσο το δυνατόν πιο αμετάβλητες [21].

#### 4.1.1 Χαρακτηριστικά Απαιτήσεων

Πρέπει να σκεφτόμαστε την κάθε απαίτηση σαν ένα αντικείμενο με ξεχωριστές ιδιότητες που την κάνουν να διαφέρει από τις υπόλοιπες. Κάθε λειτουργική απαίτηση, εκτός από την περιγραφή που γίνεται στα έγγραφα, θα πρέπει να έχει και κάποια πρόσθετα κομμάτια πληροφοριών ή χαρακτηριστικών που σχετίζονται με αυτήν. Αυτές οι πληροφορίες καθιερώνουν ένα πλαίσο και ένα υπόβαθρο σε κάθε απαίτηση που ξεφεύγει από την απλή περγραφή της προβλεπόμενης λειτουργίας της. Αυτές οι επιπρόσθετες πληροφορίες μπορούν να αποθηκευτούν σε ένα υπολογιστικό φύλλο, σε μία βάση δεδομένων ή σε ένα εργαλείο διαχείρησης απαιτήσεων. Τα εργαλεία αυτά μας επιτρέπεουν να φιλτράρουμε, να ταξινομούμε την βάση δεδομένων έτσι ώστε να δούμε το σύνολο των απαιτήσεων με βάση τιμές και τις ιδιότητές τους. Για παράδειγμα

θα είναι εφικτό να απαριθμήσουμε όλες τις απαιτήσεις υψηλής προτεραιότητας που έχουν εκχωρηθεί για υλοποίηση και έχουν κατάσταση Approved.

Είναι εξαιρετικά σημαντικό σε ένα σύνθετο και πολύπλοκο έργο να υπάρχεί μία πλούσια ποικιλία χαρακτηριστικών για κάθε απαίτηση. Είναι σημαντικό λοιπόν να προσδιορίζουμε τις απαιτήσεις σύμφωνα με τα εξής:

- Την ημερομηνία που δημιουργήθηκαν.
- Την τρέχουσα μορφή έκδοής τους.
- Τον συντάκτη που τις έγραψε.
- Το άτομο που είναι υπέυθυνο να κρίνει αν οι απαιτήσεις εκπληρώνουν το σκοπό τους.
- Τους ιδιοκτήτες των απαιτήσεων, ή μια λίτσα από τους ενδιαφερόμενους (stakeholders).
- Την κατάσταση των απαιτήσεων.
- Την καταγωγή ή την πηγή των απαιτήσεων.
- Το σκεπτικό με το οποίο έγιναν οι απαιτήσεις.
- Το υποσύστημα ή τα υποσυστήματα στα οποία εκχωρούνται οι απαιτήσεις.
- Το προϊόν στο οποίο εκχωρούνται.
- Μέθοδος επαλήθευσης που πρέπει να χρησιμοποιθεί για να αξιολογηθούν οι απαιτήσεις.
- Την προτεραιότητα της εφαρμογής τους.
- Την σταθερότητά τους.

Οι σχεδιαστές λογισμικού είναι συχνά πολύ αισιόδοξοι όταν αναφέρονται στο ποσοστό των λειτουργιών που έχουν ολοκληρώσει. Συνήθως στο ποσοστό αυτό συμπεριλαμβάνουν και λειτουργίες που ναι μεν έχουν ξεκινήσει αλλά δεν έχουν ολοκληρώσει. Αυτή η τάση υπερεκτίμησης της κατάστασης προόσου ενός έργου δεν μας επιτρέπει να καθορίσουμε ακριβώς τι πραγματικά συμβαίνει. Η παρακολούθηση της κατάστασης κάθε λειτουργικής απαίτησης κατά τη διάρκεια ανάπτυξης είναι ένα πιο ακριβές βαρόμετρο της πραγματικής προόδου του έργου.

Μερικοί αναλυτές προσθέτουν στις απαιτήσεις την κατάσταση Designed (δηλαδή πως τα σχεδιαστικά στοιχεία που αναφέρονται σε μία απαίτηση έχουν δημιουργηθεί και αξιολογηθεί) και Delivered (το λογισμικό δηλαδή που περιέχει η απαίτηση είναι στα χέρια των χρηστών για δοκιμή). Είναι χρήσιμο να τηρείται ένα αρχείο απαιτήσεων που έχουν απορριφθεί και ο λόγος που απορίφθηκαν. Η κατάσταση Rejected μας επιτρέπει να διατηρούμε μία προτεινόμενη απαίτηση για μια πιθανή μελλοντική αναφορά χωρίς να χαλάμε την συγκεκριμένη έκδοση με την οποία έχουν δεσμευτεί οι απαιτήσεις.

Η ταξινόμηση των απαιτήσεων σε διάφορες κατηγορίες είναι πιο αποτελεσματική από ότι προσπάθεια παρακολούθησης της ολοκλήρωσης κάθε απαίτησης σε κάθε Baseline. Η ενημέρωση της κατάστασης των απαιτήσεων θα πρέπει να γίνεται μόνο όταν τηρούνται συγκεκριμένες προϋποθέσεις μετάβασης. Οι αλλαγές των καταστάσεων των απαιτήσεων θα

μπορούν να οδηγήσουν σε ενημέρωση της ιχνηλασιμότητας των πληροφοριών έτσι ώστε να προσδιοριστεί ο σχεδιασμός, ο κώδικας και τα test που πρέπει να γίνουν σε κάθε απαίτηση.

***Προτεινόμενες καταστάσεις απαιτήσεων***

**Proposed.** Η απαίτηση έχει ζητηθεί από εξουσιοδοτημένη πηγή.

**Approved.** Η απαίτηση έχει αναλυθεί και έχουν προσδιοριστεί οι επιπτώσεις της στο έργο. Οι βασικοί ενδιαφερόμενοι έχουν συμφωνήσει να συμπεριληφθεί αυτή η απαίτηση και η ομάδα ανάπτυξης έχει δεσμευτεί να την εφαρμόσει.

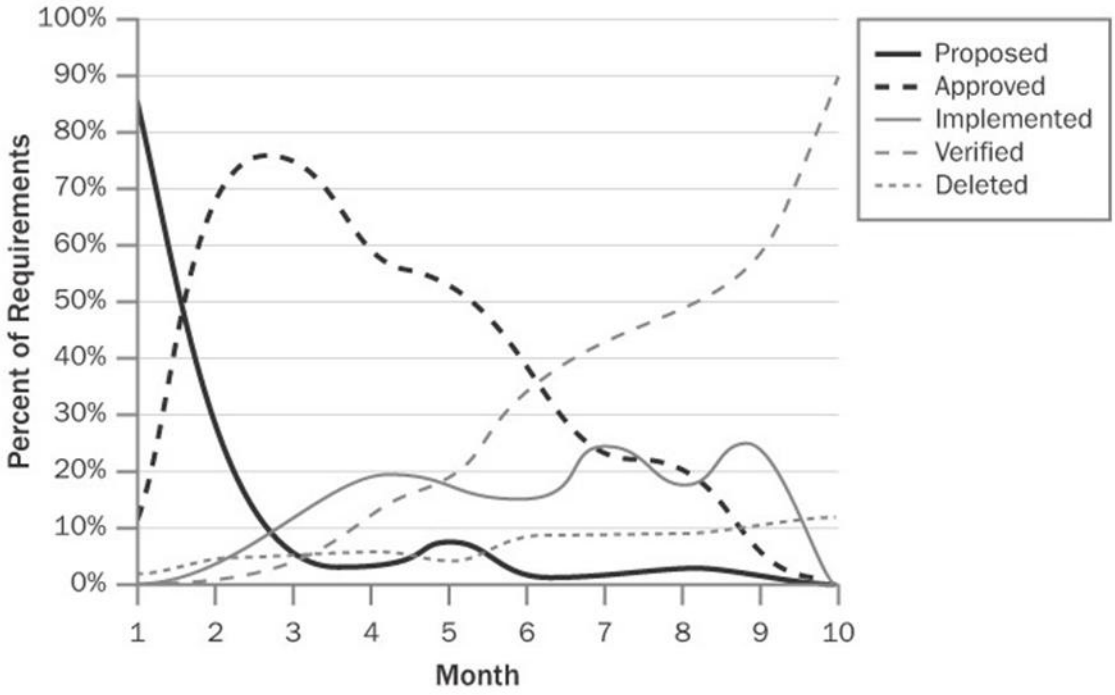
**Implemented.** Ο κώδικας υλοποιεί την απαίτηση έχει σχεδιαστεί, γραφτεί και ελεγχθεί.

**Verified.** Η σωστή λειτουργία της υλοποιημένης απαίτησης έχει επιβεβαιωθεί στο πλαίσιο του έργου που αναπτύσεται. Η απαίτηση τώρα θεωρείται πλήρης.

**Deleted.** Η συγκεκριμένη απαίτηση έχει διαγραφεί. Συμπεριλμβάνεται μία εξήγηση για το ποιός και το γιατί την διέγραψε.

**Rejected.** Η απαίτηση προτάθηκε αλλά τελικά δεν υλοποιήθηκε. Συμπεριλαμβάνετια μία εξήγηση για το ποιός και γιατί την απέριψε.

Το Σχήμα 54 δείχνει την παρακολούθηση των απαιτήσεων σε ένα υποθετικό έργο διάρκειας δέκα μηνών. Δείχνει το ποσοστό του συνόλου των απαιτήσεων, έχοντας την τιμή κάθε κατάστασης για το τέλος του κάθε μήνα. Οι καμπύλες δείχνουν πόσο πλησιάζει το έργο στον πλήρη έλεγχο των πιστοποιημένων (Approved) απαιτήσεων.




**Σχήμα 54.** Παρακολούθηση της κατανομής της κατάστασης των απαιτήσεων κατά τη διάρκεια του έργου

Για έναν οργανισμό που δεν έχει μετρήσει ποτέ μία οποιαδήποτε πτυχή του έργου του, του είναι πολύ δύσκολο να προσδιορίσει πώς κατανέμουν τον χρόνο τους οι ομάδες του. Η μέτρηση της πραγματικής ανάπτυξης ενός έργου και της προσπάθειας διεχείρησής της απαιτεί αλλαγή νοοτροπίας του οργανισμού και ατομική πειθαρχία έτσι ώτσε να καταγραφούν οι καθημερινές δραστηριότητες. Η παρακολούθηση αυτής της προσπάθειας δεν είναι τόσο χρονοβόρα όσο πιστεύεται. Ο οργανισμός θα αποκτήσει πολύτιμες πληροφορίες για το πώς έχει αφιερώσει τον χρόνο του σε διάφορες εργασίες.

Η παρακολούθηση αυτή υποδεικνύει επίσης αν η ομάδα έχει εκτελέσει όντως τις απαιτούμενες δράσεις γαι τη διαχείρηση των απαιτήσεων. Η αποτυχία της διαχείρησης των απαιτήσεων θα οδηγήσει σε άυξηση του κινδύνου του συνολικού έργου. Παρακάτω φαίνονται μερικές πρακτικές έτσι ώστε να γίνει η διαχείρηση πιο αποτελεσματική:

- Υποβολή αλλαγών των απαιτήσεων και πρόταση για νέες απαιτήσεις.
- Αξιολόγηση των προτεινόμενων αλλαγών.
- Ενημέρωση των απαιτήσεων είτε σε έγγραφα είτε σε βάσεις δεδομένων.
- Παρακολούθηση και αναφορά κατάστασης απαιτήσεων.
- Παρακολούθηση τυχόν αλλαγής δισυνδεδεμένων απαιτήσεων.
- Συλλογή πληροφοριών ιχνηλασιμότητας απαιτήσεων.

Η διαχείριση των απαιτήσεων βοηθά στο να εξασφαλιστεί πως η προσπάθεια που γίνεται για την συλλογή, την ανάλυση και την τεκμηρίωση των απαιτήσεων δεν πάει χαμένη. Αν για παράδειγμα δεν καταφέρουμε να διατηρήσουμε το σύνολο των απαιτήσεων ακαίρεο, παρόλες τις αλλαγές που γίνονται, οι ενδιαφερόμενοι δεν θα ξέουν τι θα περιμένουν να παραδοθεί σε κάθε έκδοση. Η αποτελεσματική διαχείριση των απαιτήσεων μπορεί να μειώσει το χάσμα προσδοκιών, διατηρώντας όλα τα ενδιαφερόμενα μέρη ενημερωμένα για την τρέχουσα κατάσταση των απαιτήσεων σε όλη τη διαδικασία ανάπτυξης του έργου.

Οι περισσότεροι προγραμματιστές έρχονται αντιμέτωποι με το φαινόμενο του ότι μία μικρή αλλαγή στις απαιτήσεις είναι τελικά πολύ πιο σύνθετη από ότι αναμενόταν, με αποτέλεσμα οι εκτιμήσεις του κόστους και των επιπτώσεων που γίνονται να μην είναι ρεαλιστικές. Οι πολλές και συχνές αλλαγές των απαιτήσεων κυρίως σε σύνθετα έργα δημιουργούν χάος και ελλατώνουν την ποιότητα του έργου. Ένας οργανισμός που θέλει να διαχειριστεί σοβαρά έργα λογισμικού θα πρέπει να διασφαλίσει ότι:

- Οι αλλαγές των απαιτήσεων αξιολογούνται εξονυχιστικά πριν οριστικοποιηθούν.
- Ότι τέτοιες αποφάσεις παίρνονται από κατάλληλα και εξουσιοδοτημένα άτομα.
- Οι εγκεκριμένες αλλαγές κοινοποιούνται σε όλους τους ενδιαφερόμενους συμμετέχοντες.
- Οι αλλαγές ενσωματώνονται στο έργο με σταθερό τρόπο.

Οι τροποποιήσεις του λογισμικού είναι προς όφελος του προϊόντος. Είναι σχεδόν αδύνατο μία ομάδα να προβλέψει όλες τις απαιτήσεις από την αρχή του έργου, με τις ανάγκες των χρηστών να αλλάζουν με την πάροδο του χρόνου. Μία αποτελεσματική ομάδα αναλυτών θα πρέπει να ανταποκριθεί άμεσα και με συνέπεια σε αυτές τις αλλαγές έτσι ώστε να δημιουργήσει αξία στο προϊόν που θα παράξει.

Όταν όμως γίνονται αλλαγές πάντα υπάρχει και ένα κόστος. Ακόμα και μία αλλαγή σε μία απλή ιστοσελίδα έτσι ώστε να γίνει πιο γρήγορη και αποτελεσματική, κάνοντας αλλαγές στον αλγόριθμο σχεδιασμού θα αυξήσει το συνολικό κόστος του έργου. Όταν γίνεται αίτημα απόρριψης μίας αλλαγής, θα καταναλωθούν πόροι για την υποβολή, την αξιολόγηση και την απόφαση για το αν τελικά θα απορριφθεί ή όχι. Πόροι οι οποίοι θα μπορούσαν να είχαν χρησιμοποιηθεί σε άλλες διαδικασίες που οδηγούσαν στην γρηγορότερη ολοκλήρωση του έργου. Αν τα ενδιαφερόμενα μέρη του έργου δεν διαχειριστούν τις αλλαγές που γίνονται κατά τη διάρκεια της ανάπτυξης, δεν θα ξέρουν τι πραγματικά θα πρέπει να περιμένουν όταν το έργο παραδοθεί και έτσι θα δημιουργηθεί ένα χάσμα προσδοκιών. Όσο πλησιάζουμε στην ημερομηνία παράδοσης οι επιπτώσεις των αποφάσεων των αλλαγών γίνονται πολύ πιο σοβαρές από ότι είναι στην αρχή της ανάπτυξης του έργου και ερχόμαστε εντιμέτωποι με προβλήματα καθυστέρησης παράδοσης.

Οι αναλυτές θα πρέπει να ενσωματώνουν τις εγκεκριμένες αλλαγές κατά την τεκμηρίωση των απαιτήσεων. Στην τεκμηρίωση περιγάφεται με ακρίβεια το προϊόν που πρόκειται να παραδοθεί και έτσι αν το Έγγραφο Προσδιορισμού Απαιτήσεων δεν ενημερώνεται κατά τη διάρκεια ανάπτυξης του έργου αυτομάτως η αξία του θα μειωθεί [21,52].

## 4.2 Διαχείρηση Αλλαγών Των Απαιτήσεων

Έχει διαπιστωθεί πως οι creepingrequirements δημιουργούν προβλήματα στο:

- 80% των έργων διαχείρησης πληροφοριακών συστημάτων.
- 70% των στρατιωτικών έργων λογισμικού.
- 45% έργων λογισμικού εξωτερικής ανάθεσης (outsourced).
-

Οι απαιτήσεις που αλλάζουν συχνά (Creeping Requirements) περιλαμβάνουν νέες λειτουργικότητες και σημαντικές τροποποιήσεις στις απαιτήσεις που παρουσιάζονται αφού έχουν γίνει Baselined. Οι απαιτήσεις σε έργα διαχείρησης πληροφοριών αυξάνονται κατά 1% ανά μήνα. Για ένα εμπορικό λογισμικό ο ρυθμός αύξησης των απαιτήσεων φτάνει το 3,5% ανά μήνα. Το πρόβλημα δεν είναι στην αλλαγή των απαιτήσεων, αλλά στο αντίκτυπο που έχουν αυτές όταν γίνονται στο τέλος του έργου. Αν εγκρίνεται η κάθε αλλαγή που προτείνεται το έργο διατρέχει τον κίνδυνο να μην παραδοθεί ποτέ [21].

Κάποιες από τις αλλαγές που γίνονται είναι αναπόφευκτες και συμφέρουσες. Καθώς οι επιχειρηματικές διεργασίες, οι ευκαιρίες της αγοράς, τα ανταγωνιστικά προϊόντα και οι

τεχνολογίες αλλάζουν με την πάροδου του χρόνου, έτσι και η διαχείρηση θα πρέπει να αναπροσανατολιστεί και να εφαρμόσει όποιες αλλαγές θεωρούνται απαραίτητες να γίνουν έτσι ώστε το τελικό έργο να συμβαδίζει με τις τρέχουσες καταστάσεις. Ένα ανεξέλεγκτο scopecreep το οποίο ενσωματώνει στον αρχικό σχεδιασμό νέες λειτουργίες χωρίς να συμμερίζεται τους πόρους που χρειάζονται για την προσαρμογή, τα χρονοδιαγράμματα και τα επίπεδα ποιότητας που έχουν οριστεί μπορεί να οδηγήσει τελικά στην αποτυχία του έργου. Όταν συστηματικά γίνονται μικρές ή μεγάλες αλλαγές και απρόσμενες τροποποιήσεις στις απαιτήσεις τότε σίγουρα θα οδηγηθούμε στην υπέρβαση του χρόνου παράδοσης του έργου και στην υποβάθμιση της ποιότητας που έχει συμφωνηθεί με τους πελάτες.

Το πρώτο βήμα για την διαχείρηση του scopecreep είναι η καταγραφή του οράματος, του πεδίου εφαρμογής και των περιορισμών του νέου συστήματος ως μέρος των επιχειρησιακών απαιτήσεων. Κάθε προτεινόμενη απαίτηση θα πρέπει να αξιολογείται σύμφωνα με τους στόχους της επειχείρησης, το όραμα του προϊόντος και το πεδίο εφαρμογής του έργου (projects cope). Οι πελάτες και οι χρήστες θα πρέπει να συμμετέχουν κατά τη διαδιακασία εκμαίευσης για να μην παραβλευθούν σημαντικές για το σύστημα απαιτήσεις. Το πρωτότυπο παρέχει μία προεπισκόπηση της ενδεχόμενης εφαρμογής η οποία βοηθά τους προγραμματιστές και τους αναλυτές να καταλάβαουν τις ανάγκες των χρηστών. Τέλος σε έργα που οι απαιτήσεις είναι αβέβαιες και συνεχώς ματαβαλλόμενες, η χρησιμοποιίηση μικρών κύκλων ανάπτυξης παρέχει τη δυνατότητα για ευκολότερες προσαρμογές στις απαιτήσεις [53].

### 4.2.1 Διαδικασία Ελέγχου Αλλαγής

Μία αλλαγή της διαδιακασίας ελέγχου (change control process) θα επιτρέψει στους Leaders του έργου να λάβουν τεκμηριωμένες αποφάσεις που θα παρέχουν αξία στους πελάτες ενώ παράλληλα θα ελέχθουν τον κύκλο ζωής του κόστους του έργου. Η διαδικασία αυτή μας επιτρέπει να παρακολουθούμε την κατάσταση των προτεινόμενων αλλαγών και να διασφαλίσουμε ότι δεν παραβλέπονται ή χανονται. Μόλις ένα σύνολο απαιτήσεων γίνει Baselined θα πρέπει να ακολουθήσουμε την διαδικασία για όλες τις απαιτήσεις μέχρι εκείνη την χρονική στιγμή.

Είναι ένας μηχανισμός που διοχετεύει και φιλτράρει τις αλλαγές έτσι ώστε να διασφαλιστεί ότι λαμβάνονται υπόψη μόνο οι πλέον κατάλληλες. Η διαδικασία αλλαγής θα πρέπει να είναι καλά τεκμηριωμένη, όσο το δυνατόν πιο απλή και πάνω από όλα αποτελεσματική για να μπορεί να χρησιμοποιηθεί από όπονδήποτε ενδιαφερόμενο. Αν εκείνοι θεωρήσουν πως η αλλαγή δεν είναι σημαντική, τότε δεν εξετάζεται καν υπό ένταξη.

Η διαχείριση των αλλαγών των απαιτήσεων μοιάζει αρκετά με την διαδικασία συλλογής και λήψης αποφάσεων ελλιπών αναφορών. Αυτές οι δύο δραστηριότητες μπορούν να γίνουν με τα ίδια εργαλεία. Ένα εργαλείο όμως δεν υποκαθιστά ποτέ μία διαδικασία. Η χρησιμοποίηση ενός εργαλείου παρακολούθησης και διαχείρησης προτεινόμενων αλλαγών των απαιτήσεων δεν

αντικαθιστά την γραπτή διαδικασία που περιγράφει τα περιεχόμενα και την επεξεργασία της αίτησης αλλαγής.

***Πολιτική Ελέγχου Αλλαγής***

Η διοίκηση θα πρέπει να χαράξει μία σαφής πολιτική για το πώς οι ομάδες του έργου θα χειριστούν τις προτεινόμενες αλλαγές των απαιτήσεων. Οι πολιτικές θα πρέπει να είναι ρεαλιστικές και να προσθέτουν αξία στο έργο. Για παράδειγμα:

- Όλες οι αλλαγές των απαιτήσεων θα πρέπει να ακολουθούν την διαδικασία, διαφορετικά δεν θα λαμβάνονται υπόψη.
- Δεν θα πραγματοποιείται κανένα έργο τόσο σχεδιασμού όσο και υλοποίησης, εκτός της μελέτης σκοπιμότητας, σε μη εγκεκριμένες αλλαγές.
- Ένα απλό αίτημα για αλλαγή δεν θα εγγυάται ότι εκείνη θα πραγματοποιηθεί. Το Project's Change Control Board (CCB) θα αποφασίζει αν τελικά θα εφαρμοστούν οι αλλαγές.
- Τα περιεχόμενα της βάσης δεδομένων των αλλαγών θα είναι προσβάσιμα από όλους τους συμμετέχοντες του έργου.
- Το πρωτότυπο κείμενο της αίτησς αλλαγής δεν θα πρέπει να τροποποιείται ή να διαγράφεται.
- Για κάθε αλλαγή θα πρέπει να διενεργείται και ανάλυση των επιπτώσεων που θα δημιουργηθούν στο έργο.
- Θα καταγράφεται στο σκεπτικό σύμφωνα με το οποίο εγκρίθηκε ή απορρίφθηκε μια οποιαδήποτε αλλαγή.

Όπως είναι λογικό θα υπάρχουν αλλαγές που θα επηράζουν από ελάχιστα έως καθόλου το έργο, αλλά θα υπάρχουν και άλλες που θα έχουν σημαντικές επιπτώσεις σε αυτό. Για τον λόγο αυτό όλες οι αλλαγές θα πρέπει να γίνονται μέσω της διαδικασίας ελέγχου αλλαγής (change-controlprocess). Αν τώρα μία αλλαγή είναι χαμηλού ρίσκου, ελάχιστου κόστους, δεν επηρεάζει το έργο σχεδόν καθόλου και δεν χρειάζεται να εργαστούν πάνω από ένα άτομα για να πραγματοποιηιεί, τότε θα δίνεται στην διακριτική ευχέρεια του προγραμματιστή να την υλοποίησει έτσι ώστε να εξοικονομηθεί χρόνος.

***Περιγραφή Διαδικασίας Ελέγχου Αλλαγής***

Στην συνέχεια βλέπου ένα πρότυπο (template) για την διαδικασία ελέγχου αλλαγής (Change-controlprocess) για να διαχειρσιτούμε τις τροποποιήσεις των απαιτήσεων αλλά και τις άλλες αλλαγές του έργου [21].

**1. Introduction**

1.1 Purpose

1.2 Scope
1.3 Definitions

**2.Roles and Responsibilities**

**3.Change-Request Status**

**4.Entry Criteria**

**5.Tasks**

5.1 Evaluate Request
5.2 Make Decision
5.3 Make Change
5.4 Notify All Affected Parties

**6.Verifl cation**

6.1 Verify Change
6.2 Install Product

**7.Exit Criteria**

**8.Change-Control Status Reporting**

**Appendix: Data Items Stored for Each Request**

- *Entry Criteria. Πρέπει να εκπληρώνονται όλες οι προϋποθέσεις πριν την εκτέλεση της διαδικασίας ελέγχου των αλλαγών.*
- *Οι διάφορες διεργασίες που εμπλέκονται στην διαδικασία, ο ρόλος των υπευθύνων και των άλλων συμμετεχόντων στην διαδικασία.*
- *Βήματα που εξασφαλίζουν πως όλες οι διεργασίες έχουν εκπληρωθεί με σωστό τρόπο.*
- *Exit Criteria. Οι προϋποθέσεις που διασφαλίζουν πως η διαδικασία έχει εκπληρωθεί επιτυχώς.*

**Introduction**. Εδώ περιγράφεται ο σκοπός της διαδικασίας και το οργανωσιακό πλαίσιο για το οποίο γίνεται. Σημειώνονται οι τομείς στους οποίους η διαδιακασία θα επιφέρει περαιτέρω αλλαγές, αλλά και οι οποιοδήποτε τομείς θα πρέπει να εξαιρεθούν από αυτές. Ορίζονται οι απαραίτητες επεξηγήσεις για την κατανόηση των αλλαγών.

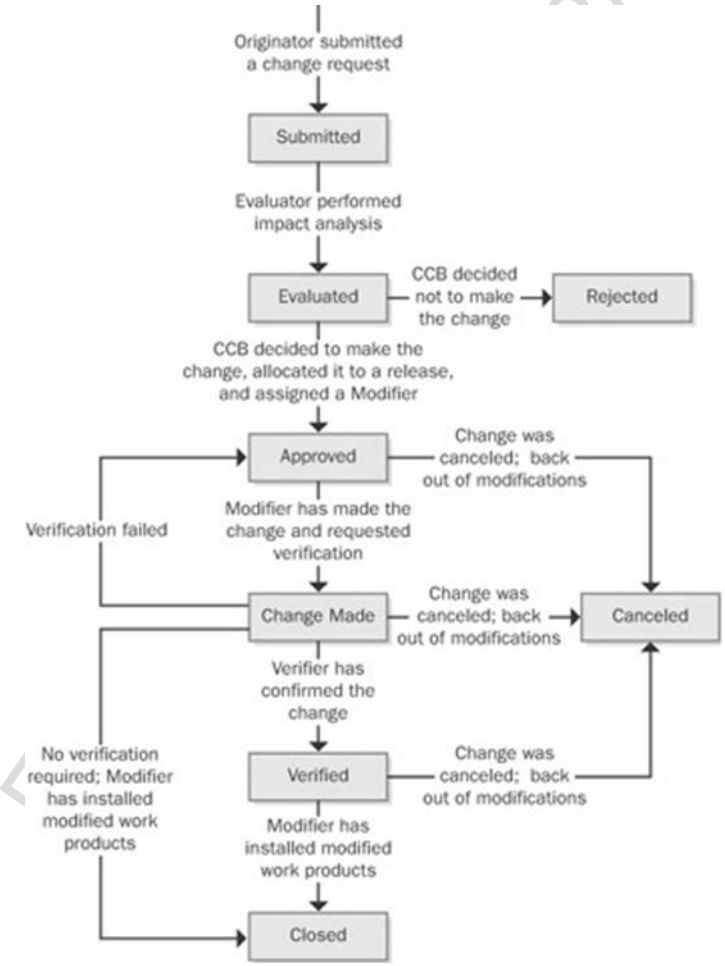
**Rolesand Responsibilities.** Ορίζεται η ομάδα των μελών του έργου, κατά ρόλο και όχι ονομστικά, ποιοί θα συμμετέχουν στις αλλαγές και οι γίνεται περιγραφή των δραστηριοτήτων τους. Ο παρακάτω πίνακας δείχνει μερικούς ενδεικτικούς ρόλους. Γίνεται διαχωρισμός των ρόλων του καθενός έτσι ώστε να μην υπάρχουν επικαλυπτόμενες διεργασίες.

**CCB Chair.** Ο πρόεδρος του Change Control Board. Είναι υπεύθυνος για την τελική λήψη των απαοφάσεων. Είναι εκείνος που θα επιλέξει τα πρόσωπα για τις θέσεις των Evaluator και Modifier.

**CCB.** Η ομάδα που αποφασίζει την έγκριση ή την απόρριψη των προτεινόμενων αλλαγών σε κάποιο συγκεκριμένο έργο (project).

**Evaluator**. Το άτομο από το οποίο ζητείται να αναλύσει την επίπτωση των προτεινόμενων αλλαγών. Θα μπορούσε να είναι κάποιος τεχνικός, ένας πελάτης, ένας υπάλληλος του τμήματος marketing ή και ένας συνδιασμός των παραπάνω.
**Modifier.** Ο υπέυθυνος να κάνει τις αλλαγές σε ένα έργο αφού εγκριθεί κάποιο σχετικό αίτημα.
**Originator**. Εκείνος που υπάλει μία νέα αίτηση αλλαγής.
**Request Reciever.** Το πρόσωπο στο οποίο υποβάλονται οι νέες αιτήσεις αλλαγής.
**Verifier.** Εκείνος που τελικά κρίνει έαν οι αλλαγές έγιναν σωστά.

**Change-Request Status**. Ένα αίτημα αλλαγής εκτελεί ένα προκαθορισμένο κύκλο ζωής, έχοντας διαφορετικές καταστάσεις κατά τη διάρκεια της ζωής του. Οι διαφορετικές καταστάσεις φαίνονται στο Σχήμα 55. Η ενημέρωση της κατάστασης του αιτήματος γίνεται μόνο αν πληρούνται τα απαραίτητα κάθε φορά κριτήρια.




**Σχήμα 55.** Διαγραμμα μετάβασης κατάστασης σε ένα αίτημα αλλαγής

**Entry Criteria.** Το βασικό κριτήριο εισόδου της διαδικασίας ελέγχου αλλαγής (Change-Control Process) είναι:

- Μία έγκειρη αίτηση αλλαγής που έχει διαβιβαστεί μέσω ενός εγκεκριμένου καναλιού.

Όλοι οι δυνητικοί Originators θα πρέπει να είναι σε θέση να υποβάλουν μία αίτηση αλλαγής, είτε αυτή γίνεται συμπληρώνοντας μία διαδυκτιακή φόρμα, είτε στέλνοντας ένα mail, είτε χρησιμοποιώντας κάποιο συγκεκριμένο Change-Control εργαλείο. Θα γίνεται καταχώρηση μία μοναδικής ετικέτας σε κάθε αίτημα αλλαγής και θα δρομολογείται κατευθείαν στον Request Receiver.

**Tasks**. Το επόμενο βήμα είναι να αξιολογηθεί η τεχνική εφικτότητα, το κόστος, οι πόροι και το αν τελικά η αλλαγή ευθυγραμμίζεται με το επιχειρησιακό πλάνο. Ενας Evaluator θα αναλύει τις επιτπωσεις της προκείμενης αλλαγής και το ρίσκο που θα επιφέρει αυτή στο τελικό έργο. Θα πρέπει επίσης να λάβει υπόψη και τις επιπτώσεις και το ρίσκο που θα δημιουργηθούν αν τελικά απορριφθεί η αλλαγή.

Αν τελικά ληφθεί απόφαση έγκρισης της αλλαγής, το CCB της προσδίδει ένα επίπεδο προτεραιότητας και μία ημερομηνία ολοκλήρωσης. Ενημερώνεται η κατάσταση του αιτήματος και ενημερώνονται όλες οι ομάδες που θα χρειστεί να κάνουν με την σειρά τους αλλαγές στις εργασίες τους. Ο Modifier θα ενημερώνει και θα τεκμηριώνει τις αλλαγές - συμπεριλαμβανομένων των προδιαγραφών των σχεδίων, του κώδικα, του ελέγχου και των manual - που θα προκύψουν σε όλο το έργο.

**Verification.** Όταν γίνεται η έγκριση των αλλαγών των απαιτήσεων, πρέπει να εξασφαλίζεται πως θα τροποποιηθούν όλα τα κομμάτια του έργου τα οποία επηρεάζονται. Έτσι χρησιμοποιούμε τις πληροφορίες ιχνηλασιμότητας να βρούμε ποια κομμάτια του συστήματος έχουν αυτές οι αλλαγές αγγίξει. Μετά την έγκριση ο Modifier εγκαθιστά τις ενημερωμένες εργασίες και επαναπροσδιορίζει το Baseline των αλλαγμένων απαιτήσεων, έτσι ώστε να είναι όλα διαθέσιμα στην υπόλοιπη ομάδα.

**Exit Criteria**. Για να ολοκληρωθεί σωστά η Change-Control Process, θα πρέπει να πληρούνται όλα τα παρακάτω κριτήρια:

- Η κατάσταση του αιτήματος είναι Rejected,Closed ή Canceled.
- Έχουν εγκατασταθεί όλα τα τροποποιημένα προϊόντα.
- Όλοι οι εμπλεκόμενοι του έργου έχουν ενημερωθεί για τις λεπομέρειες των αλλαγών και τις τρέχουσες κατατάσεις των αιτήσεων.
- Η βάση (matrix) ιχνηλασιμότητας έχει ενημερωθεί.

**Change-Control Status Reporting**. Πρέπει να ενημερωθούν τα διαγράμματα και οι αναφορές της Change-Control βάσης δεδομένων. Τα διαγράμματα αυτά δείχνουν τον αιθμό των αιτήσεων των αλλαγών ανά κατάσταση κατηγορίας συναρτήσει του χρόνου. Οι παραπάνω αναφορές χρησιμοποιούνται για την παρακολούθηση του έργου από τους Project Managers.

**Appendix:**Πληροφορίες που έχουν αποθηκευτεί για κάθε αίτηση αλλαγής.

Παρακάτω θα δούμε ορισμένα στοιχεία που θα πρέπει να αποθηκευτούν για ένα αίτημα αλλαγής.
**Change Origin**. Η λειτουργική ομάδα που ζήτησε την αλλαγή. Όπως η ομάδα marketing, η ομάδα διαχείρησης του έργου, οι πελάτες, οι μηχανικοί ή οι ομάδα ελέγχου.
**Change-Request ID.** Ετικέτα ταυτοποίησης αίτησης.
**Change Type.** Τύπος του αιτήματος αλλαγής. Αν είναι δηλαδή αλλαγή της απαίτησης, ή ενίσχυσή της ή κάποιο διαπιστωμένο λάθος σε αυτή.
**Date Submitted**. Ημερομηνία δημιουργίας του αιτήματος αλλαγής.
**Date Updated**. Ημερομηνία της τελευταίας τροποποίησης της αλλαγής.
**Description.** Περιγραφή της ζητούμενης αλλαγής.
**Implementation Priority**. Η προτεραιότητα υλοποίηση όπως αυτή έχει δοθεί από το CCB.
**Modifier.** Το όνομα του ατόμου που είναι υπεύθυνο για την υλοποίηση της αλλαγής.
**Originator.** Το όνομα του ατόμου που πρότεινε την αλλαγή.
**Planed Released.** Ημερομηνία ολοκλήρωσης της αλλαγής.
**Project.** Το Project στο οποίο γίνετια η αλλαγή.
**Status.** Η τρέχουσα κατάσταση της αλλαγής.
**Title.** Μία περίληψη μίας γραμμής της προτεινόμενης αλλαγής.
**Verifier.** Το όνομα του ατόμου που θα αποφανθεί αν η αλλαγή έγινε με επιτυχία.

***Η Επιτροπή Ελέγχου Αλλαγών (Change Control Board)***
Η ChangeControlBoard είναι μία πολύ καλή λύση γαι την σωστή διαχείρηση των απαιτήσεων. Είναι μία ομάδα ανθρώπων, σε μία ή σε ξεχωριστές ομάδες, που αποφασίουν για το ποιές από τις προτεινόμενες αλλαγές θα εγκριθούν σε ένα συγκεκριμένο έργο. Αποφασίζει επίσης για το πότε και αν θα διορθωθούν κάποια αναφερθέντα ελαττώματα. Οριμένα έργα έχουν ήδη μία ομάδα που παίρνει τις αποφάσεις για τις εκάστοτε αλλαγές, αλλά η δημιουργία της CCB επισιμοποιεί αυτή την ομάδα και ορίζει τις λειτουργίες της.

Ορίσμενα από τα μέλη της είναι εξουσιδοτημένα να παίρνουν αποφάσεις για αλλαγές και απλά να ενημρώνουν την διοίκηση για αυτές, ενώ άλλα μπορούν να κάνουν απλά συστάσεις για τυχόν αλλαγές. Στα μικρά έργα τα άτομα που παίρνουν τις αποφάσεις είναι λίγα, αλλά όταν ένα έγο είναι σύνθετο τότε απαιτέιται η CCB να έχει πολλά επίπεδα. Ορισμένοι σε αυτά τα επίπεδα είναι υπεύθυνοι να παίρνουν επιχειρισιακές αποφάσεις, ενώ άλλοι τις αντίστοιχες τεχνικές. Εκείνοι που είναι στα υψηλότερα επίπεδα ασχολούνται με τις αλλαγές που θα έχουν μεγάλη επίπτωση στο έργο. Αν για παράδειγμα ένα έργο περιλαμβάνει πολλά υποσυστήματα θα δημιουργηθούν CCBs για κάθε υποσύστημα, που θα ασχολούνται με τις αλλαγές που γίνονται σε κάθε ένα από αυτά, αλλά και μία CCB που θα ασχολείται με τις αλλαγές που επηρεάζουν συνολικά το έργο.

Η CCB δεν σημιουργείται για να προσθέσει πεισσότερη γραφειοκρατεία, αλλά για να βοηθήσει στην διαχείρηση των αλλαγών ακόμα σε μικρά έργα. Η δομής της δεν πρέπει να είναι χρονοβόρα και περίπλοκη, αλλά σαφής και αποτελεσματική. Η δουλειά της είναι να αξιολογήσει

όλες τις προτεινόμενες αλλαγές και να πάρει γρήγορες αποφάσεις που θα βασίζονται στην ανάλυση των πιθανών επιπτώσεων και των οφελών που αυτές θα επιφέρουν. Θα πρέπει επίσης να διασφαλίζει πως τα άτομα που θα κάνουν κάθε φορά την αξιολόγηση και θα παίρνουν τις αποφάσεις έγκριψης ή απόρριψης των αλλαγών θα είναι τα πλέον κατάλληλα.

***Σύνθεση της CCB***

Η CCB πρέπει να αντιπροσωπευει όλες τις ομάδες του έργου που συμμετέχουν στην λήψη των αποφάσεων κατά τη διάρκεια του έργου. Θα πρέπει να υπάρχει μία ομάδα ατόμων από συγκεκριμένους τομείς που θα παίρνουν τις αποφάσεις. Η επιλογή των εκπροσώπων μπορεί να γίνει από τους ακόλουθους τομείς [21]:

- Διαχείρηση του έργου ή του του προγράμματος.
- ProductManagement ή αναλυτές απαιτήσεων.
- Ομάδα ανάπτυξης.
- Ομάδα ελέγχου και διασφάλισης ποιότητας.
- Ομάδα Marketing και εκπρόσωπων των πελατών.
- Ομάδα τεχνικής υποστήριξης.
- Configuration management

Η CCB ενός έργου που πρέπει να δημιουργήσει και software αλλά και hardware θα πρέπει ενδεχομένως να συμπεριλάβει άτομα απο τα τμήματα Hardware και SystemEngineering. Κρατώντας την ομάδα όσο το δυαντόν μικρότερη την κάνουμε αυτομάτως πιο ευέλικτη και αποτελεσματική στις αιτήσεις αλλαγών. Οι μεγάλες ομάδες έχουν αντιμετωπίζουν συχνά προβλήματα στον προγραμματισμό συναντήσεων και στην λήψη αποφάσεων. Είναι καλό επίσης να συμμετέχουν στις συναντήσεις και κάποια εξειδικευμένα άτομα στους τομείς που θα γίνει συζήτηση για αλλαγές, έτσι ώστε να εξασφαλιστεί ότι η ομάδα έχει επαρκείς τεχνικές και επιχειρηματικές γνώσεις.

Ο σκοπός, το πεδίο εφαρμογής (scope authority), τα μέλη και οι διαδιακασίες της CCB περιγράφονται από ένα σύνολο κανόνων. Οι κανόνες αυτοί καθορίζουν τη συχνότητα των τακτικών συνατήσεων των μελών και τις συνθήκες που θα διέπουν την κάθε συνεδρίαση. Το πεδίο εφαρμογής (scope authority) είναι αυτό που καθορίζει ποιές αποφάσεις μπορούν να ληφθούν από τα χαμηλότερα επίπεδα και ποιες πρέπει να περάσουν στα υψηλότερα.

Όπως όλα τα όργανα λήψης αποφάσεων έτσι και η CCB θα πρέπει να ορίσει κάποιους κανόνες που θα την βοηθήσουν να πραγματοποιήσει το έργο της. Η περιγραφή της διαδικασίας λήψης αποφάσεων θα πρέπει να περιλαμβάνει τα ακόλουθα:

- Τον αριθμό των μελών και τους ρόλους που είναι απαραίτητοι για να ληφθεί μία απόφαση.
- Αν για την λήψη της απόφασης θα χρησιμοποιείται ψήφος, απλή συναίνεση ή κάποια άλλη διαδικασία.
- Εάν ο πρόεδρος της CCB θα μπορεί να ακυρώσει μία συλλογική απόφαση.
- Εάν ο πρόεδρος ή κάποιο άλλο μέλος θα πρέπει να επικυρώσει την απόφαση.

Η CCB θα πρέπει να ζυγίζει να αναμενόμενα οφέλη με τις εκτιμώμενες επιπτώσεις της αποδοχής μιάς προτεινόμενης αλλαγής. Τα πλεονεκτήματα από την βελτίωση του προϊόντος περιλαμβάνουν αύξηση των εσόδων, εξοικονόμηση πόρων, μεγαλύτερη ικανοποίηση των πελατών και ανταγωνιστικό πλεονέκτημα για το προϊόν που θα παραδοθεί. Οι επιπτώσεις της προκείμενης αλλαγής έχουν να κάνουν με αύξηση του κόστους ανάπτυξης, καθυστέρηση παράδοσης, υποβάθμιση της συμφωνημένης ποιότητας και μείωση της λειτουργικότητας του που έχει σαν αποτέλεσμα την δυσαρέσκεια των πελατών. Αν οι επιπτώσεις στο κόστος ή στο χρονοδιάγραμμα υπερβαίνουν τα ανώτατα όρια που έχει ορίσει η CCB τότε η λήψη της απόφασης θα πρέπει να γίνει από τα υψηλότερα επίπεδα.

Όταν η CCB πάρει μία απόφαση θα πρέπει αυτομάτως να ενημερωθεί η βάση δεδομένων των αλλαγών. Υπάρχουν εργαλεία που δημιουργούν μηνύματα ηλεκτρονικού ταχυδρομείου για να ενημερώσουν τον Originator και τα άλλα μέλη για την τρέχουσα κατάστση της αίτησης αλλαγής έτσι ώστε νε προβούν στις κατάλληλες διεργασίες.

***Επαναδιαπραγμάτευση των δεσμεύσεων***

Πριν την αποδοχή μίας σημαντικής αλλαγής απαιτήσεων θα πρέπει να αναδιαπραγματετούν οι δεσμεύσεις τις διοίκησης με τους πελάτες του προγράμματος. Μπορεί να γίνει διαπραγμάτευση για αύξηση του χρόνου παράδοσης του έργου, αύξηση του προσωπικού και αναβολή απαιτήσεων χαμηλότερης προτεραιότητας. Αν δεν ληφθούν ορισμένες δεσμεύσεις και δεν τεκμηριωθούν οι απειλές της επιτυχάς του έργου είναι πολύ πιθανό στο τέλος να μην έχουμε το επιθυμητό αποτέλεσμα.

***Εργαλεία Διαδικασίας Ελέγχου Αλλαγής***

Η διαδιακασία ελέγχου αλλαγής των απαιτήσεων μπορεί να γίνει ευκολότερα με την χρησιμοποίηση κάποιων ειδικών αυτοματοποιημέτων εργαλείων.Πολλές ομάδες χρησιμοποιούν εμπορικά εργαλεία για τον εντοπισμό, την συλλογή, την αποθήκευση και την διαχείρηση των αλλαγών των απαιτήσεων. Με τα εργαλεία αυτά μπορούμε να δούμε τις αιτήσεις για αλλαγή ανά κατηγορία ανά πάσα στιγμή [52,54].

Τα διαθέσιμα εργαλεία αλλάζουν συχνά αλλά συγκεντρώνουν κάποια κοινά χαρακτηριστικά:

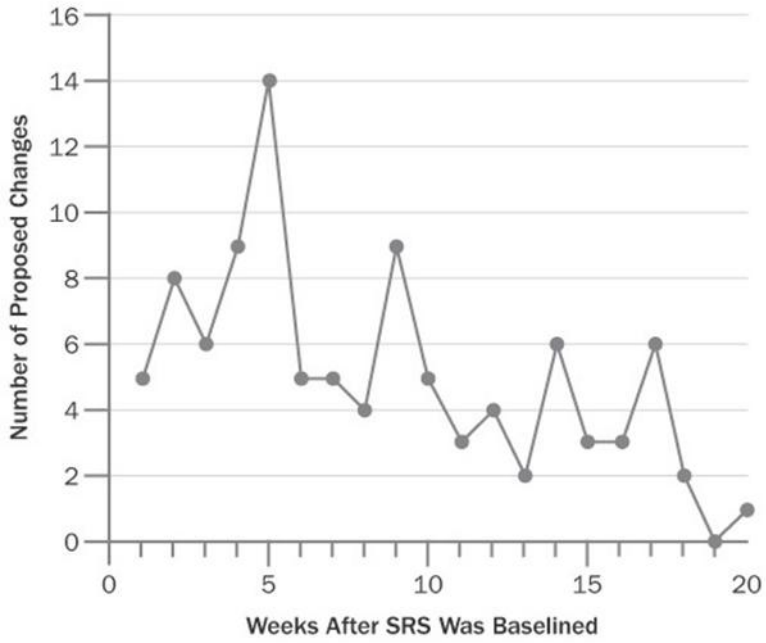
- Μας επιτρέπουν να ορίζουμε τις όλες τις πληροφορίες που περιλαμβάνονται σε μία αίτηση αλλαγής.
- Μας επιτρέπουν να ορίσουμε ένα State-Transition model για τον κύκλο ζωής του αίτηματος αλλαγής.
- Επιτρέπει τις αλλαγές της κατάστασης μόνο στους εξουσιοδοτημένους χρήστες.
- Καταγράφει την ημερομηνία αλλαγής κάθε κατάστασης και τον χρήστη που την έκανε.
- Μας ειδοποιεί όταν ένας Originator προτείνει μία αλλαγή.
- Μας ειδοποιεί όταν υπάρχει αλλαγή στην κατάσταση ενός αιτήματος.
- Δημιουργεί αναεφορές και διαγράμματα βοηθώντας στην καλύτερη διαχείρηση του έργου.

***Μέτρηση Αλλαγών Των Απαιτήσεων***

Η μέτρηση των αλλαγών των απαιτήσεων μας βοηθά να έχουμε μία γενικότερη επίβλεψη του έργου, δίνοντάς μας ακριβείς και υποκειμενικές πληροφορίες. Οι μετρήσεις θα πρέπει να ευθυγραμμίζονται στα ερώτηματα που πρέπει να απαντηθούν από τους διαχειριστές και στους στόχους που πρέπει να επιτευχθούν. Είναι ένας τρόπος για να αξιολογηθεί η σταθερότητα των απαιτήσεων. Βελτιώνουν την όλη διαδικασία και οδηγούν σε λιγότερα αιτήματα αλλαγών στην πορεία του έργου. Είναι καλό να ελέγχουμε τα εξής:

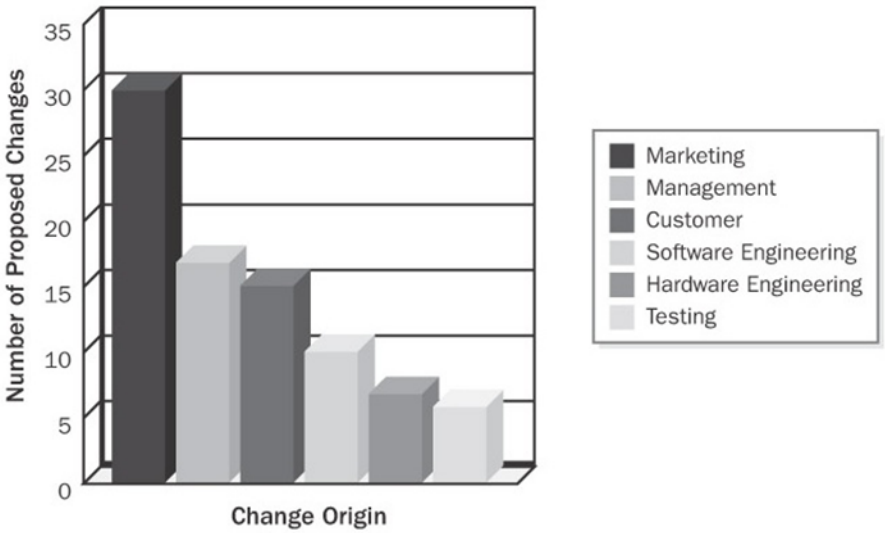
- Τον αριθμό των αιτήσεων των αλλγών που έχουν γίνει.
- Τον συνολικό αριθμό των αλλαγών που έγιναν συμπεριλαμβανομένων των απαιτήσεων που έγιναν προσθήκες, που διαγράφηκαν ή τροποποιήθηκαν.
- Τον αρθμό των αιτήσεων που έχει κάνει η κάθε πηγή.
- Τον αριθμό των αλλαγών που έγιναν σε απαιτήσεις που είχε γίνει Baseline.
- Την συνολική προσπάθεια που καταβάλλεται για την επεξεργασία και την εφαρμογή των αιτήσεων αλλαγών.

Πριν όμως ξεκινήσουμε να κάνουμε μητρήσεις θα πρέπει να καταλάβουμε καλά τους στόχους και όλα τα δεδομένα που έχουμε. Το Σχήμα 56 μας δείχνει έναν τρόπο να παρακολουθούμε τον αριθμό των αιτήσεων αλλαγών που γίνονται κατά τη διάρκεια του έργου. Δείχνει τον ρυθμό με τον οποίο προτείνονται νέες αλλαγές. Η συχνότητα αυτών των αλλαγών θα πρέπει να μηδενίζεται όσο πλησιάζει η ημερομηνία παράδοσης του έργου. Όσο η συχνότητα των αλλαγών δεν μειώνεται τόσο μεγαλώνει ο κίνδυνος να μην παραδοθεί έγκαιρα το έργο. Αν το διάγραμμα δείξει ότι οι απαιτήσεις που έχουν γίνει Baseline είναι ελλιπείς, τότε θα πρέπει να βελτιωθούν οι τεχνικές εκμαίευσης απαιτήσεων [55,56].




**Σχήμα 56**. Διάγραμμα δραστηριότητας αλλαγών απαιτήσεων

Ο διαχειριστής του έργου που ανησυχεί ότι οι συχνές αλλαγές θα έχουν επιπτώσεις στην ημερομηνία παράδοσης του έργου είναι καλό να παρακολουθεί τις πηγές των συχνών αιτημάτων αλλαγών. Το Σχήμα 57 μας δείχνει έναν τρόπο να ελέγχουμε τα αιτήματα που έχουν γίνει από τις διάφορες πηγές. Έτσι θα μπορεί να γίνει μία συζήτηση με το τμήμα που κάνει τα περισσότερα αιτήματα και να βρεθεί ένας τρόπος να μειωθούν. πό τις απαιτήσεις στην σχεδίαση του έργου.




**Σχήμα 57**.Διάγραμμα πηγής αιτημάτων αλλαγής απαιτήσεων

Η ανάλυση των επιπτώσεων είναι ένας σημαντικός παράγοντας της διαχείρησης των απαιτήσεων. Παρέχει ακριβή κατανόηση των επιπτώσεων των προτεινόμενων αλλαγών η οποία με την σειρά της βοηθά στην λήψη σωστών και τεκμηριωμένων αποφάσεων για το ποιές από αυτές θα εγκριθούν ή όχι. Η ανάλυση εξετάζει τις προτεινόμενες αλλαγές ώστε να καθοριστούν τα στοιχεία που πρέπει να δημιουργηθούν ή τροποποιηθούν και την προσπάθεια που πρέπει να καταβληθεί ώστε αυτή να υλοποιηθεί. Πριν ληφθεί λοιπόν μία οποιαδήποτε απόφαση έγκρισης ή απόρριψης θα πρέπει προηγουμένως να αφιερωθεί κάποιος χρόνος στην ανάλυση των επιπτώσεων.

Ο πρόεδρος της CCB ζητάει από κάποιο μέλος της να κάνει την ανάλυση των επιπτώσεων που θα επιφέρει η συγκεκριμένη αλλαγή. Αυτή η ανάλυση θα πρέπει να περιλαμβάνει τρεις κύριες πτυχές:

- Την κατανόηση των πιαθνών επιπτώσεων που θα επιφέρει η αλλαγή. Αν προσθέσουμε πολύ λειτουργικότητα σε ένα προϊόν μπορεί να μειώσουμε την απόδοσή του σε μη επιτρεπτά επιπεδα.
- Να προσδιοριστουν όλα τα αρχεία, τα μοντέλα και τα έγγραφα που θα πρέπει να τροποποιηθούν.
- Να προσδιορισούν οι απαραίτητες εργασίες αλλά και η προσπάθεια που θα πρέπει να καταβληθεί για να ολοκληρωθούν οι αλλαγές.

### 4.3 Ιχνηλασιμότητα

Η ιχνηλασιμότητα των απαιτήσεων είναι η τεκμηρίωση μεταξύ των διασυνδέσεων των διαφορετικών απαιτήσεων. Οι απαιτήσεις εκμειεύονται από διαφορετικές πηγές και πρέπει κάπως να διασφαλιστεί και η προέλευσή τους. Επίσης θα πρέπει να καταγράφονται και να τεκμηριώνονται και αλλαγές που κατά καιρούς κάποιος μπορεί να κάνει. Η ιχνηλασιμότητα ορίζεται ως [21]:

"*Η ιχνηλασιμότητα των απαιτήσεων είναι η ικανότητα να περιγράφουμε και να ακολουθούμε τη ζωή μίας απαίτησης και προς τις δύο κατευθύνσεις, δηλαδή από την προέλευσή της, την δημιουργία της και τον ορισμό της μέχρι την μεταγενέστερη εγκατάσταση και χρήση της ακόμα και σε περιόδους συνεχής αναβαθμισής της*."

Είναι ένας πολύ σημαντικός παράγοντας γιατί μας φανερώνει και τια αλλαγές που έχει δεχθεί μία απαίτηση με την πάροδο του χρόνου. Επίσης την ταυτοποιεί όταν επέλθει μία αλλαγή πάνω της. Επίσης μας παρέχει πληροφορίες που μας βοηθούν να εξακριβώσουμε αν ληφθεί υπ' όψην όλες οι δισυνδέσεις μεταξύ των απαιτήσεων.

Επιτρέπει στον διαχειριστή να ανιχνεύει μία συγκεκριμένη απαίτηση από την δημιουργία της μέχρι και τις συνολικές αλλαγές που έχει υποστεί κατά τη διάρκεια της ζωής της. Επιπλέον όταν μία απαίτηση υποστεί μία αλλαγή υπάρχει μεγάλη πιθανότητα να μεταδοθούν τυχόν αλλαγές και

σε άλλες απαιτήσεις που είναι διασυνδεδεμένες μαζί της. Η ιχνηλασιμότητα μας επιτρέπει να ελέγξουμε το που θα γίνουν πιθανές αλλαγές και μας βοηθάει έτσι να τις εκτιμήσουμε και τις προγραμματίσουμε. Τέλος μας δίνει τη δυνατότητα να κατηγοριοποιούμε και να δίνουμε προτεραιότητα σε κάποιες απαιτήσεις οι οποίες με τη σειρά τους θα μας βοηθήσουν να εκμαιεύσουμε τις υπόλοιπες απαιτήσεις από τα ενδιαφερόμενα μέλη.

Ένα εργαλείο διαχείρησης εμπορικών απαιτήσεων που αποθηκεύει τις απαιτήσεις και τις συναφείς πληροφορίες σε μία Multi-user βάση δεδομένων παρέχει μία πολύ καλή λύση. Τα εργαλεία αυτά παρέχουν συναρτήσεις για τη διαχείρηση και την προβολή των δεδομένων της βάσης, την εισαγωγή και την εξαγωγή των απαιτήσεων, τον καθορισμό των συσχετίσεων μεταξύ των απαιτήσεων και την σύνδεση των απαιτήσεων με τα υπόλοιπα εργαλεία ανάπτυξης λογισμικού.

Παρακάτω θα περιγραφθούν ορισμένα πλεονεκτήματα που μπορεί ένα εργαλείο διαχείρησης απαιτήσεων να παρέχει. Θα δούμε κάποια βασικά χαρακτηριστικά που μας δίνουν τέτοια εργαλεία και θα παρουσιαστούν τέσσερα εργαλεία διαχείρησης εμπορικών απαιτήσεων (TBI's Caliber-RM, QSS's DOORS, Rational's RequisitePro, and Integrated Chipware's RTM). Τα παραπάνω εργαλεία δεν θα μας βοηθήσουν να συγκεντρώσουμε τις απαραίτητες απαιτήσεις για ένα πληροφοριακό σύστημα και δεν έχουν σκοπό να αντικαταστήσουν την διαδικασία διαχείρησης απαιτήσεων. Ένα εργαλείο δεν είναι μία διαδικασία από μόνο του αλλά βοηθάει στην υποστήριξη και στην υλοποίηση της διαδικασίας.

Τα μέλη της ομάδας ανάπτυξης θα πρέπει να έχουν πρόσβαση στις απαιτήσεις κατά τη διάρκεια του έργου, ίσως μέσω μίας διαδυκτιακής εφαρμογής που θα τους δίνει πρόσβαση σε εργαλεία διαχείρησης. Η διαχείρηση των απαιτήσεων περιλαμβάνει όλες τις δραστηριότητες που εξασφαλίζουν την ακεραιότητα, την ακρίβεια και την συμφωνία των απαιτήσεων κατά την εξέλιξη του έργου. Στο παρακάτω σχήμα βλέπουμε και διαγραμματικά τι ακριβώς περιλαμβάνεται στην Διαχείρηση των Απαιτήσεων:

- Έλεγχος των αλλαγών των απαιτήσεων.
- Διατήρηση του πλάνου σύμφωνα με την αλλαγή και την ενημέρωση των απαιτήσεων.
- Έλεγχος των εκδόσεων τόσο των μεμονομένων απαιτήσεων όσο και των εγγράφων τους.
- Παρακολούθηση της κατάστασης των απαιτήσεων κατά τη διάρκεια του έργου.
- Διαχείρηση των διασυνδέσεων μεταξύ των απαιτήσεων και των άλλων διεργασιών του έργου.

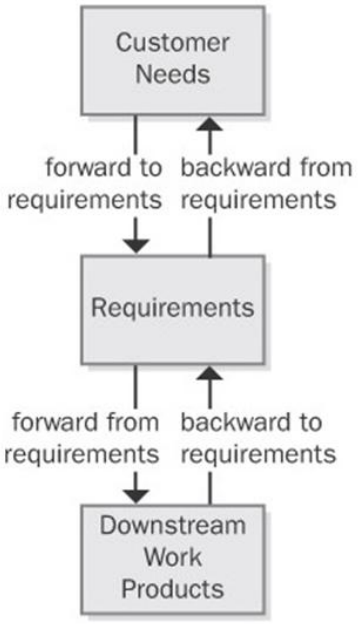
Μία ομάδα που της προτείνεται να γίνουν αλλαγές στις απαιτήσεις μπορεί να μην είναι σε θέση να ανταποκριθεί και να μην εκπληρώσει το υπάρχον πρόγραμμα και τις δεσμεύσεις ποιότητας που υπάρχουν. Είναι δουλειά του Project Manager να διαπραγματευτεί τις αλλαγές αυτές με

τους υπόλοιπους Managers, τους πελάτες και τα εμπλεκόμενα στο έργο μέλη. Το έργο μπορεί να ανταποκριθεί στις νέες ή αλλαγμένες απαιτήσεις με διάφορους τρόπους:

- Καθυστέρηση εκπλήρωσης απαιτήσεων χαμηλής προτεραιότητας.
- Απόκτηση επιπλέον προσωπικού.
- Ολίσθηση του προγράμματος έτσι ώστε να συμβιβαστεί με τις νέες λειτουργικότητες.
- Υποβάθμιση της ποιότητας έτσι ώστε να τηρηθεί το πρόγραμμα.

### 4.3.1 Ανίχνευση Απαιτήσεων

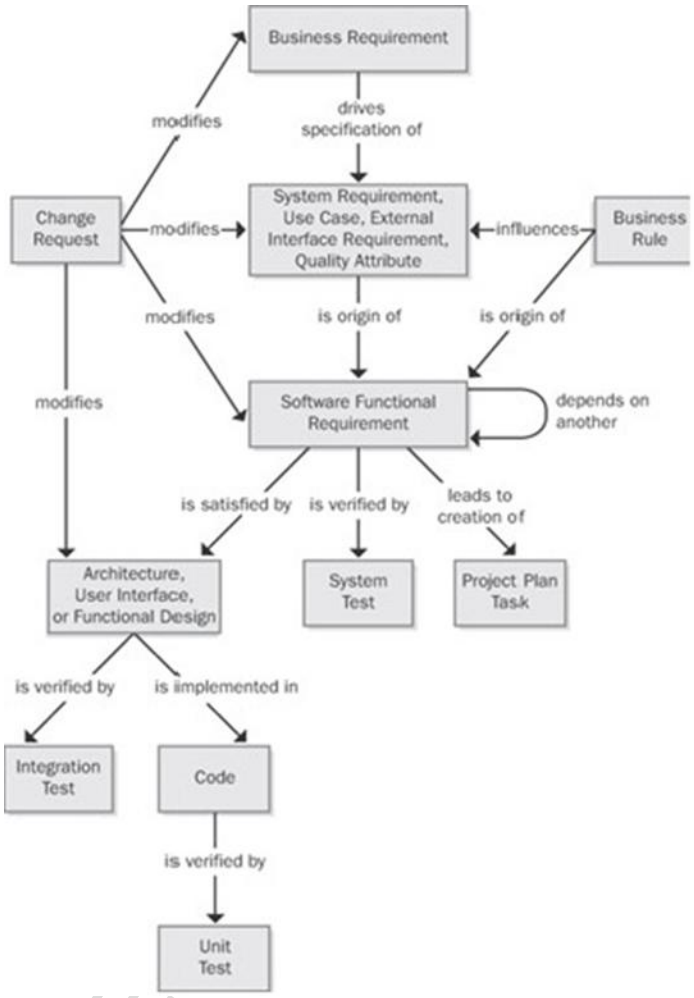
Στο Σχήμα 58 φαίνονται τέσσερις δεσμοί ιχνηλασιμότητας απαιτήσεων. Φαίνεται διαγραμματικά πώς οι ανάγκες των πελατών επηρεάζουν τις απαιτήσεις (Forward) , αλλά και πώς η αλλαγή των απαιτήσεων επηράζει τις ανάγκες των πελατών (Backward).




**Σχήμα 58.** Τέσσερις τύποι ιχνηλασιμότητας απαιτήσεων

Αν οι ανάγκες των χρητών έχουν τεκμηριωθεί σε usecases, το πάνω μισό του σχήματος μας δείχνει τις διασυνδέσεις μεταξύ των usecases και των λειτουργικών απαιτήσεων. Το κάτω μισό συσχετίζει τις απαιτήσεις με την ροή των παραδοτέων κομματιών κατά τη διάρκεια της ανάπτυξης. Αυτός ο τύπος σύνδεσης εξασφαλίζει πως έχει ικανοποιηθεί η κάθε απαίτηση. Με τον τρόπο αυτό είμαστε κάθε φορά σε θέση να γνωρίζουμε τόσο για ποιό στοιχείο έχει δημιουργηθεί η κάθε απαίτηση όσο και το αντίστροφο.

Το Σχήμα 59 δείχνει πολλά είδη άμεσης ιχνηλασιμότητας που ορίζονται σε ένα έργο. Είμαστε έτσι σε θέση να γνωρίζουμε ποιές διασυνδέσεις είναι αυτές που θα μας βοηθήσουν στην επιτυχή ανάπτυξη και αποτελεσματική συντήρηση του έργου.




**Σχήμα 59.** Πιθανά είδη διασυνδέσεων ιχνηλασιμότητας απαιτήσεων

### 4.3.2 Οφέλη ιχνηλασιμότητας

- **Certification.** Μπορούμε να χρησιμοποιήσουμε τις πληροφορίες ιχνηλασιμότητας για να πιστοποιήσουμε πως έχουν επιτευχθεί όλες οι απαραίτητες για μία αλλαγή απαιτήσεις.
- **Change impact analysis**. Χωρίς αυτές υπάρχει ο κίνδυνος να παραλείψουμε ένα ή και περισσότερα στοιχεία του συστήματος κατά την προσπάθεια να προσθέσουμε, τροποποιήσουμε ή διαγράψουμε κάποια απαίτηση.
- **Maintenance.** Με αξιόπιστες πληροφορίες ιχνηλασιμότητας αυξάνουμε την παραγωγικότητά μας. Πολλές φορές οι αλλαγές στις επιχειρηματικές πολιτικές ή άλλων εξωτερικών παραγόντων επιβάλουν αλλαγές στον τρόπο λειτουργίας του λογισμικού.

Ένας πίνακας, λοιπόν, που μας δείχνει από ποιές απαιτήσεις εκπλκηρώνονται οι πολιτικές αυτές μας βοηθάει να κάνουμε τις αλλαγές που κάθε φορά χρειάζονται.

- **Project tracking.** Μία επιμελής καταγραφή των πληροφοριών ιχνηλασιμότητας κατά της ανάπτυξης, μας βοηθάει να έχουμε μία ακριβή εικόνα της κατάστασης του έργου. οι διασυνδέσεις που υπολείπονται μας υποδεικνύουν ποιά προϊόντα εργασίας δεν έχουν ακόμη δημιουργηθεί.
- **Reengineering.** Μπορούμε να κάνουμε μία λίστα με τις λειτουργίες ενός συστήματος που θέλουμε να αντικαταστήσουμε και να τις αντιστοιχήσουμε στις απαιτήσεις του νέου συστήματος. Ο καθορισμός των συνδέσεων ιχνηλασιμότητας ένας τρόπος να κατανοήσουμε τον τρόπο που θα λειτουργεί το νέο σύστημα.
- **Risk reduction.** Η τεκμηρίωση των πληροφοριών διασύνδεσης μειώνει τον κίνδυνο αποτυχίας αν για κάποιο λόγο ένα κύριο μέλος της ομάδας ανάπτυξης παραιτηθεί από το έργο.
- **Testing**. Αν μία δοκιμή δεν δώσει το επιθυμητό αποτέλεσμα, οι διασυνδέσεις μεταξύ των δοκιμών , των απαιτήσεων και του κώδικα θα μας επισημάνουν ποιά τμήματα πρέπει να επανεξετάσουμε γλιτωνοντάς μας από περιτές και χρονοβόρες δοκιμές.

Τα παραπάνω είναι κάποια μακροπρόθεσμα οφέλη που μπορεί να αυξάνουν το κόστος ανάπτυξης του έργου, αλλά εγγυώνται την μείωση του ρίσκου αποτυχίας. Οι ιχνηλασιμότητα των απαιτήσεων είναι στην ουσία μία επένδυση που αυξάνει τις πιθανότητες έγκαιρης παράδοσης του έργου και ικανοποίησης των πελατών. Αν και είναι δύσκολο να ποσοτικοποιηθεί, η επένδυση αυτή θα αποδίδει κάθε φορά που θα θέλουμε να τροποποιήσουμε, επεκτείνουμε ή αντικαταστήσουμε ένα προϊόν. Είανι πολύ πιο εύκολοτερο να γίνει παράλληλα με την ανάπτυξη του έργου παρά όταν αυτό ολοκληρωθεί [57,58].

***Μήτρα Ανιχνευσιμότητας Απαιτήσεων***

Ο πιο συνηθισμένος τρόπος να αναπαραστήσουμε τις διασυνδέσεις μεταξύ των απαιτήσεων των υπόλοιπων στοιχείων του συστήματος είναι η το Requirements Traceability Matrix (αναφέρεται επίσης και σαν Requirements Trace Matrix ή Treceability Table).

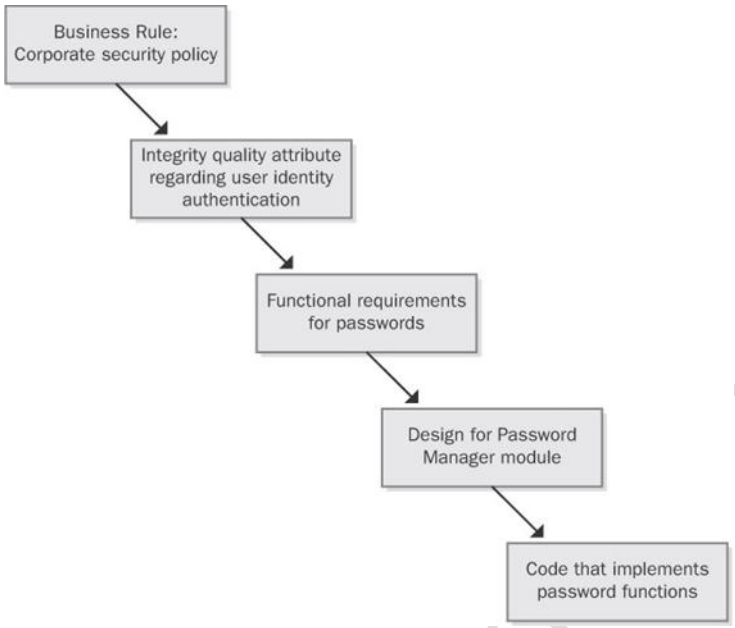
Ο Πίνακας 13 μας δείχνει πως συνδέεται Forward η κάθε λειτουργική απαίτηση με ένα συγκεκριμένο Use Case, τον σχεδιασμό, τον κώδικα και τα στοιχεία ελέγχου. Τα στοιχεία σχεδιασμού είναι αντικείμενα της ανάλυσης όπως διαγράμματα ροής δεδομένων, πίνακες και κλάσεις αντικειμένων. Οι κώδικες είναι αποθηκευμένες διαδικασίες, ονόματα πηγαίου κώδικα ή διαδικασίες και λειτουργίες στα αρχεία του κώδικα. Μπορούμε να προσθέσουμε περισσότερες στήλες, όπως η ηλεκτρονική τεκμηρίωση και το manual, για να διευρύνουμε τις διασυνδέσεις και σε άλλα απκέτα εργασίας. Για προσθέσουμε περισσότερες λεπτομέρειες στον πίνακα θα χρειαστεί να σπαταλήσουμε περισσότερους πόρους και χρόνο αλλά θα μας δώσει την ακριβή θέση των στοιχείων λογισμικού, κάτι το οποίο θα μας βοηθήσει γλιτώνοντάς μας χρόνο κατά την διάρκεια της ανάλυσης επιπτώσεων των αλλαγών και της συντήρησης του έργου [21,59].

| Απαιτήσεις Χρηστών | Λειτουργικές Απαιτήσεις | Σχεδιασμός Στοιχείων | Κώδικας | Περίπτωση Ελέγχου |
|---|---|---|---|---|
| **UC-28** | catalog.query.sort | Class catalog | catalog.sort() | search.7<br>search.8 |
| **UC-29** | catalog.query.import | Class catalog | catalog.import()<br>catalog.validate() | search.12<br>search.13<br>search.14 |

**Πίνακας 13.** Μία πιθανή εκδοχή της Requirements Traceability Matrix

Συμπληρώνουμε τις πληροφορίες στον πίνακα όταν αυτές γίνονται και όχι ότν προγραμματιστούν να γίνουν. Δηλαδή θα συμπληρώσουμε το "catalog.sort()" στο CodeModule μόνο όταν οκώδικας για αυτή τη λειτουργία έχει γραφτεί, έχει περάσει τους ελέγχους και έχει ενσωματωθεί στον πηγαίο κώδικα του έργου. Με τον τρόπο αυτό ο αναγνώστης γνωρίζει ότι όσα έχουν γραφτεί στα κελιά της Requirements Traceability Matrix έχουν ήδη ολοκληρωθεί. Οι δοκιμές ελέγχου που έχουν καταγραφεί για κάθε απαιτήση δεν υποδεικνύουν ότι έχει γίνει έλεγχος σε όλο το λογισμικό, αλλά υποδηλώνουν πως έχουν γραφτεί έτσι ώστε να επαληθεύουν τις απαιτήσεις την δεδομένη χρονική στιγμή. Η παρακολούθηση των δοκιμών είναι μία ξεχωριστή διαδικασία [21].

Οι μη λειτουργικές απαιτήσεις, όπως οι στόχοι απόδοσης και τα ποιοτικά χαρακτηριστικά δεν ανιχνεύονται πάντα άμεσα στον κώδικα. Μία λειτουργική απαίτηση μπορεί να υποδείξει την χρήση συγκεκριμένου hardware, αλγορίθμων, δομών δεδομένων και αρχιτεκτονικής δομής που πρέπει να χρησιμοποιηθεί. Μία απαίτηση φορητότητας μπορεί να περιορίσει τα χαρακτηριστικά της γλώσσας που χρησιμοποιεί ο προγραμματιστής αλλά δεν μπορεί να υποδείξει ποιος ακριβώς κώδικας πρέπει να γραφτεί. Η ακεραιότητα των απαιτήσεων, για την πιστοποίηση του χρήστη, προέρεται από λειτουργικές απαιτήσεις όπως κωδικοί πρόσβασης ή βιομετρική λειτουργικότητα. Σε αυτές τις περιπτώσεις η διασύνδεση γίνεται Backward στις αντίστοιχες λειτουργικές και Forward στα παραδοτέα (Deliverables) του συστήματος. Στο Σχήμα 60 φαίνεται μία πιθανή αλυσίδα ιχνηλασιμότητας των μη λειτουργικών απαιτήσεων.




**Σχήμα 60.** Αλυσίδα ιχνηλασιμότητας λειτουργικών απαιτήσεων για την ασφάλεια των εφαρμογών

Οι διασυνδέσεις ιχνηλασιμότητας μπορούν να καθορίσουν τις One-To-One, One-To-Many και Many-To-Many σχέσεις των στοιχείων του συστήματος.
**One-To-One.** Ένα στοιχείο σχεδιασμού που εφαρμόζεται σε ένα στοχείο κώδικα.
**One-To-Many.** Μία λειτουργική απαίτηση επαληθεύεται από πολλές περιπτώσεις δοκιμών ελέγχου.
**Many-To-Many.** Κάθε use-case οδηγεί σε πολλές λειτουργικές απαιτήσεις και ορισμένες λειτουργικές απαίτησεις έχουν κοινά use-cases.

Ένας άλλος τρόπος για να παρουσιαστεί η ιχνηλασιμότητα είναι μέσω μίας σειράς πινάκων που ορίζουν τα πιθανά ζεύγη των δεσμών των στοιχείων του συστήματος.

- Ένας τύπος απαίτησης για απαιτήσεις του ίδιου τύπου.
- Ένας τύπος απαίτησης για απαιτήσεις διαφορετικού τύπου.
- Ένας τύπος απαίτησης για περιπτώσεις ελέγχου.

Χρησιμοποιώντας αυτές τις τεχνικές ιχνηλασιμότητας μπορούμε να ορίσουμε ποιές απαιτήσες ορίζονται από κάποιες άλλες, ποιές εξαρτώνται ή περιορίζονται από ποιές και ποιά είναι η προέλευσή τους.

Ο Πίνακας 14 μας δείνει μία αμφίδρομη Traceability Matrix. Τα κελιά τα οποία έχουν μαρκαριστεί μας δείχνουν το σημείο τομής δύο στοιχείων του έργου. Εδώ βλέπουμε πως μία λειτουργική απαίτηση ανιχνεύεται από μία συγκεκριμένη περίπτωση χρήσης.

| Λειτουργικές Απαιτήσεις | Περιπτώσεις Χρήσης | | | |
|---|---|---|---|---|
| | **UC-1** | **UC-2** | **UC-3** | **UC-4** |
| **FR-1** | ← | | | |
| **FR-2** | ← | | | |
| **FR-3** | | | ← | |
| **FR-4** | | | ← | |
| **FR-5** | | ← | | ← |
| **FR-6** | | | ← | |

**Πίνακας 14.** RequirementsTraceabilityMatrix που δείνει τις διασυνδέσεις μεταξύ των λειτουργικών απαιτήσεων και των use-cases

Οι συνδέσεις ιχνηλασιμότητας θα πρέπει να γίνονται μόνο από όποιους έχουν τις κατάλληλες πληροφορίες. Ο Πίνακας 15 μας δείχνει τον καθορισμό των ρόλων των ατόμων που παρέχουν πληροφορίες ιχνηλασιμότητας. Οι συμμετέχοντες θα πρέπει να εξηγήσουν ποιες θα είναι οι απαιτήσεις ιχνηλασιμότητας και πώς αυτές θα προσθέσουν αξία στο έργο [21].

| Link Source Object Type | Link Target Object Type | Information Source |
|---|---|---|
| Απαίτηση Συστήματος | Απαίτηση Λογισμικού | Μηχανικός Συστήματος |
| Περίπτωση Χρήσης | Λειτουργική Απαίτηση | Αναλυτής Απαιτήσεων |
| Λειτουργική Απαιτήση | Λειτουργική Απαιτήση | Αναλυτής Απαιτήσεων |
| Λειτουργική Απαιτήση | Περίπτωση Ελέγχου | Μηχνικός Ελέγχου |
| Λειτουργική Απαιτήση | Στοιχείο Αρχιτεκτονικής Λογισμικού | Αναλυτής - Ομάδα ανάπτυξης |
| Λειτουργική Απαιτήση | Στοιχείο Σχεδιασμού | Σχεδιαστής - Ομάδα ανάπτυξης |
| Στοιχείο Σχεδιασμού | Κώδικας | Ομάδα ανάπτυξης |
| Επιχειρηματικός Κανόνας | Λειτουργική Απαίτηση | Αναλυτής Απαιτήσεων |

**Πίνακας 15**.Πηγές διασυνδέσων ιχνηλασιμότητας πληροφοριών



### 4.4 Εργαλεία Διαχείρισης Απαιτήσεων

Υπάρχουν εργαλεία διαχείρησης απαιτήσεων που έχουν δυνατότητες ιχνηλασιμότητας απαιτήσεων. Μπορούμε να αποθηκεύσουμε, στην βάση δεδομένων τους, απαιτήσεις και άλλες πληροφορίες και να καθορίσουμε τις συνδέσεις ιχνηλασιμότητας. Μερικά από αυτά τα εργαλεία μας επιτρέπουν να διαχωρίσουμε την "Traced-To" με την "Traced-From" σχέση διασύνδεσης. Όταν δηλαδή υποδείξουμε πως μία απαίτηση R ανιχνεύεται από μία δοκιμή ελέγχου T, το εργαλείο θα μας υποδείξει και την συμμετρική σχέση για το πώς η T ανιχνεύεται από την R.

Κάποια εργαλεία σηματοδοτούν αυτόματα ως ύποπτο έναν σύνδεσμο, βάζοντας ενα ερωτηματικό ή μία κόκκινη γραμμή, όταν αλλάξει μία από τις δύο άκρες του. Αν για παράδειγμα

αλλάξουμε το usecase 3 που σχετίζεται με τις απαιτήσεις FR3, FR4 και FR6 τότε αυτόματα θα μας επισημάνει αυτές τις απαιτήσεις μήπως πρέπει να γίνουν αλλαγές και εκεί.

| **Λειτουργικές Απαιτήσεις** | **Περιπτώσεις Χρήσης** | | | |
|---|---|---|---|---|
| | **UC-1** | **UC-2** | **UC-3** | **UC-4** |
| **FR-1** | ← | | | |
| **FR-2** | ← | | | |
| **FR-3** | | | ← ? | |
| **FR-4** | | | ← ? | |
| **FR-5** | | ← | | ← |
| **FR-6** | | | ← ? | |

**Πίνακας 16**. Ύποπτες διασυνδέσεις σε μία RequirementsTraceabilityMatrix

**Διαδικασία ιχνηλασιμότητας απαιτήσεων**

Μία διαδικασία ιχνηλασιμότηας απαιτήσεων θα μπορούσε να είναι η εξής:

**1**.Επιλογή των σχέσεων δισαύνδεσης που επιθυμούμε.

**2.**Επιλογή του τύπου της Traceability Matrix που θα χρησιμοποιήσουμε. Η την απλή ή την σύνθετη.

**3.**Επιλογή του μηχανισμού αποθήκευσης που θα χρησιμοποιήσουμε. Έναν πίνακα, ένα υπολογιστικό φύλλο ή ένα εργαλείο διχείρησης απαιτήσεων.

**4.**Προσδιορισμός των μερών του έργου που θέλουμε να διατηρήσουμε τις πληροφορίες ιχνηλασιμότητας.

**5**.Ενημέρωση των πληροφοριών ιχνηλασιμότητας αμέσω μόλις γίνει διεργασία που προκαλεί αλλαγή στην αλυσίδα των απαιτήσεων.

**6**.Ορισμός ετικετών που θα προσδιορίζουν με μοναδικό τρόπο όλα τα στοιχεία του συστήματος.

**7.**Δημιουργία scripts που θα αναλύουν τα αρχεία του συστήματος για την κατασκευή και την ενημέρωση των πινάκων ιχνηλασιμότητας.

**8.**Προσδιορισμός τόσο των ατόμων που θα παρέχουν πληροφορίες διασύνδεσης και όσο εκείνων που θα συντονίζουν τις δραστηριότητες και θα διαχειρίζονται τα δεδομένα.

**9**.Εκαπίδευση των ατόμων σχετικά με τις έννοιες και την σημασία των απαιτήσεων ιχνηλασιμότητας και των τεχνικών αποθήκευσής τους.

**10.**Καθορισμός των πληροφοριών ιχνηλασιμότητας καθώς το έργο εξελίσσεται και όχι όταν αυτό τελειώσει.

**11.**Περιοδικός έλεγχος των πληροφοριών ιχνηλασιμότητας.

Υπάρχουν έγγραφα στα οποία περιλαμβάνουν τις λειτουργικές και μη λειτουργικές απαιτήσεις, τις περιγραφές των περιπτώσεων χρήσης και τις επιχειριασκές απαιτήσεις. Μία Document-based προσέγγιση για την αποθήκευση των παραπάνω πληροφοριών μπορεί να μας οδηγήσει στους ακόλουθους περιορισμούς:

- Είναι δύσκολο να ενημερώνουμε και να συγχρονίζουμε τα έγγραφα.
- Είναι δύσκολο να αποθηκεύουμε τις συμπληρωματικές της κάθε απαίτησης.
- Η παρακολούθηση των καταστάσεων των απαιτήσεων είναι μία πολύ σύνθετη διαδικασία.
- Είναι δύσκολο να εντοπίζουμε την διασύνδεση μεταξύ των απαιτήσεων και των άλλων στοιχείων του συστήματος.
- Η διαχείριση του συνόλου των απαιτήσεων που έχουν σχεδιαστεί για διάφορες εκδόσεις γίνεται δύσκολα στο χαρτί καθώς θα πρέπει να αλλάζουμε συνεχώς τις προδιαγραφές των απαιτήσεων.
- Η επανξαχρησιμοποίηση των απαιτήσεων σημαίνει ότι ο αναλυτής θα πρέπει να αντιγράφει το πρωτότυπο SRS σε ένα SRS του άλλου συστήματος ή του προϊόντος που θα χρησιμοποιηθεί ξανά η απαίτηση.
- Είναι δύσκολο οι συμμετέχοντες του έργου να κάνουν τις απαράιτητες τροποποιήσεις αν βρίσκονται σε διαφορετικές γεωγραφικές τοποθεσίες.
- Δεν υπάρχει κατάλληλο μέρος να αποθηκευτούν οι απαιτήσεις που απορρίφθηκαν και αυτές που διαγράφτηκαν.

Ένα εργαλείο διαχείρισης απαιτήσεων που αποθηκεύει τις πληροφορίες σε μία κοινή βάση δεδομένων προς τους χρήστες δίνει μία λύση στους παραπάνω περιορισμούς. Σε μικρά έργα η ίδια δουλειά μπορεί να γίνει και από υπολογιστικά φύλλα ή απλές βάσεις δεδομένων που θα αποθηκεύουν όλες τις πληροφορίες. Σε μεγαλύτερα και πιο σύνθετα έργα θα πρέπει να χρησιμοποιούνται εργαλεία διαχείρησης. Τα εργαλεία αυτά επιτρέπουν στους χρήστες να εισάγουν απαιτήσεις, να καθορίζουν τα χαρακτηριστικά τους, να φιλτράρουν και να επεξεργάζονται τα περιεχόμενα της βάσης, να εξάγουν δεδομένα σε διάφορα Formats, να καθορίζουν τις διασυνδέσεις ιχνηλασιμότητας και να συνδέουν όλα τα αποθηκευμένα στοιχεία με άλλα εργαλεία ανάπτυξης [60,61].

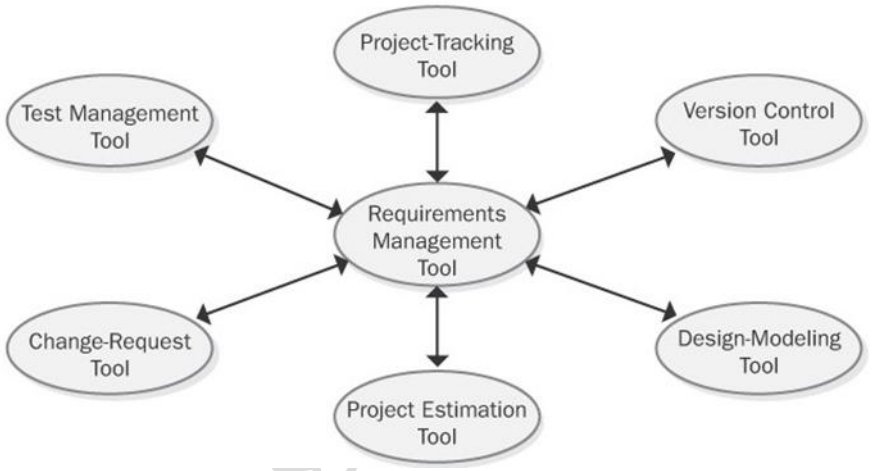
Τα εργαλεία αυτά δεν μας βοηθήσουν να ορίσουμε τους χρήστες και να συγκεντρώσουμε τις κατάλληλες προϋποθέσεις για το έργο μας. Σε καμμία περίπτωση δεν αντικαθιστούν τις διεργασίες της εκμαίευσης, της ανάλυσης και της τεκμηρίωσης των απαιτήσεων αλλά θα κάνουν πιο ευέλικτη την διαδικασία διαχείρησης των αλλαγών τους. Επίσης καταγράφουν την λογική σύμγωνα με την οποία άλλαξε η κάθε απαίτηση και μπορούν να κάνουν επαναφορά σε κάποια προηγούμενη έκδοση εφόσον κριθεί απαραίτητο.

Τα εργαλεία διαχείρισης μας επιτρέπουν να ορίσουμε τις απαιτήσεις σε διαφορετικούς τύπους όπως λειτουργικές ή επιχειρηματικές απαιτήσεις, απαιτήσεις Hardware και περιπτώσεις χρήσης. Αυτό έχει σαν αποτέλεσμα να διαφοροποιήσουμε τις πληροφορίες που περιέχονται στο Έγγραφο Προσδιορισμού των Απαιτήσεων και να εξορύξουμε τα αντικείμενα που θέλουμε να αντιμετωπίσουμε σαν απαιτήσεις. Όλα τα εργαλεία παρέχουν μεγαλύτερες δυνατότητες

καθορισμού των χαρακτηριστικών των απαιτήσεων από ότι μία τυπικό κειμενο-κεντρική προσέγγιση Εγγράφου Προσδιορισμού των Απαιτήσεων.

Ιεραρχούν τις απαιτήσεις με αριθμιτικές ετικέτες, έτσι ώστε η κάθε απαίτηση να έχει ένα ξεχωριστό χαρακτηριστικό που την καθιστά έυκολα και γρήρορα αναγνωρίσιμη. Οι ετικέτες αυτές συνήθως αποτελούνται από ένα πρόθεμα - UR αν είναι User Requirement – ακολουθούμενο από έναν μοναδικό ακέραιο αριθμό. Κάποια από τα εργαλεία παρέχουν και ένα σύστημα αναζήτησης, παρόμοιο με την αναζήτηση των Windows (MicrosoftWindowsExplorer) για να μπορύμε να διαχειριστούμε ευκολότερα το ιεραρχικό δέντρο των απαιτήσεων.

Παρέχουν την δυναντότητα να ορίσουμε ομάδες χρηστών και να δώσουμε σε αυτές τα κατάλληλα δικαιώματα να δούν ή να επεξεργαστούν τις απαιτήσεις ή τις τιμές τους.




**Σχήμα 61.** Η ενσωμάτωση των εργαλείων διαχείρησης απαιτήσεων με άλλα εργαλεία λογισμικού

***Caliber-RM*.** Το Caliber-RM είναι το κατάλληλο για χρήστες που επιθυμούν να έχουν μία προσέγγιση στην βάση δεδομένων παρά μία στο έγγραφο για τη διαχείρηση των απαιτήσεων. Προσφέρει χρηστικότητα, συμπεριλαμβανομένων των πολλών επιλογών που επιτρέπουν στους χρήστες να προσαρμόσουν τις προτιμήσεις τους. Το Caliber-RM έχει μικρή συμβατότητα με το Word αλλά είναι αρκετά ευέλικτο σε δυνατότητες εισαγωγής. Ένα ευέλικτο Document Factory επιτρέπει να καθορίσει ένα πρότυπο SRS που έχει συμπληρωθεί με τις ιδιότητες των απαιτήσεων από τη βάση δεδομένων και έτσι με συγκεκριμένες ερωτήσεις μπορεί να συμπληρωθεί ένα κείμενο Word.

Το Caliber-RM παρέχει μία υπηρεσία αναζήτησης για τον χειρισμό ιεραρχικών δέντρων δεδομένων με πρόσβαση στις λεπτομέρεις των απαιτήσεων. Μπορούμε επίσης να διαχειριστούμε τις σχέσεις ιχνηλασιμότητας των ίδιων ή και διαφορετικών τύπων απαιτήσεων. Παρέχει την εμφάνιση των απαιτήσεων αλλά μόνο για ανάγνωση εκτός από λίγε εξαιρέσεις

(threated discussion topics). Είναι ένα αρκετά καλοσχεδιασμένο εργαλείο, έυκολο στην χρήση και μπορεί να διαχειριστεί απαιτήσεις τόσο για μικρά όσο και για μεγάλα έργα.

***DOORS.*** Το DOORS είναι ένα εξελιγμένο προϊόν που μπορεί να διαχειριστεί απαιτήσεις για ένα μεγάλο έργο. Επεξεργάζεται τις επιμέρους απαιτήσεις σαν αντικείμενα αλλά τις παρουσιάζει σε μία οπτική μορφή που μοιάζει με ένα δομημένο ιεραρχικά έγγραφο απαιτήσεων. Μία εκτυπωμένη αναφορά των απαιτήσεων που έχουν επιλεγεί από την βάση δεδομένων μοιάζει με ένα δομημένο πίνακα SRS. Οι απαιτήσεις που εμφανίζονται έχουν και κάποιες χαρακτηριστικές τιμές, όπως δείκτες των συσχετίσεών τους με άλλες απαιτήσεις, καθώς και έγχρωμες γραμμές που υποδεικνύουν ότι μία απαίτηση άλλαξε κατάσταση. Το DOORS αποθηκεύει το ιστορικό των αλλαγών των επιμέρους αντικειμένων και των modules αλλά δεν δίνονται αριθμοί αναθεώρησης.

Η γλώσσα της DOORS, σε περίπτωση που θα χρειαστούν τυχόν επεκτάσεις στο κύριο προϊόν, μοιάζει πολύ με την C. έχει άμεσες διασυνδέσεις με το MicrosoftProject, το Teamwork και το RationalRose αλλά μπορούν ωστόσο να δημιουργηθούν και διασυνδέσεις με άλλα εργαλεία μέσω ενός ανοικτού API. Υποστηρίζει πολλές μορφής εισαγωγής και εξαγωγής δεδομένων, συμπεριλαμβανομένου του Word. Παρέχει ένα ολοκληρωμένο σύστημα που επιτρέπει στους χρήστες να σχολιάσουν και να αναθεωρήσουν ένα έργο ή μία ενότητα απαιτήσεων. Υπάρχει και μία ετερογενής client-server εφαρμογή για όποιον χρησιμοποιεί ένα μικτό Unix και Windows περιβάλλον. Είναι λιγότερα εύχρηστο από τα άλλα προϊόντα αλλά προσφέρει πολλούς μηχανισμούς επεξήγησης.

***RequisitePro.*** Το RequisitePro διανέμεται σαν ένα component του Rational Suite Analyst Studio. Έχει μίακειμενό-κεντρική (document centric) προσέγγιση για την διαχείρηση των απαιτήσεων και είναι το πιο συμβατό και απο τα τέσσερα εργαλεία με το Word. Μέσα από το έγγραφο του Word μπορούμε να επιλέξουμε κομμάτια κειμένου και να τα συμπεριλάβουμε στη βάση δεδομένων σαν ξεχωριστές απαιτήσεις. Μέσω ενός παραθύρου διαλόγου μπορούμε να έχουμε πρόσβαση στις λεπτομέρειες των απαιτήσεων, συμπεριλαμβανομένων των ιστορικών αναθεώρησης, τα χαρακτηριστικά, την ιχνηλασιμότητα την ιεραρχία και τις συζητήσεις. Οι μηχανισμοί που χρησιμοποιεί για τον συχγρονισμό των απαιτήσεων στη βάση δεδομένων με τα περιέχομενα του SRS είναι λίγο αδέξιοι. Παρόλα αυτά χειρίζεται το ιστορικό των εκδόσεων των απαιτήσεων πολύ καλά.

Μπορούμε να δούμε τον κατάλογο των απαιτήσεων στη βάση δεδομένων και να δούμε ή να αλλάξουμε τα χαρακτηριστικά τω απαιτήσεων. Είναι εύκολο στην χρήση αλλά δεν είναι τόσο φιλικό όπως το Caliber-RM στην αναζήτηση. Εμφανίζει μία μήτρα ιχνηλασιμότητας (matrix) που μας επιρέπει να ορίζουμε και να αλλάζουμε τις διασυνδέσεις.

***RTM Workshop.*** Το RTM Workshop μπορεί να διαχειριστεί απαιτήσεις σε πολύ μεγάλα έργα. Αντιμετωπίζει και αυτό τις απαιτήσεις σαν αντικείμενα, τα οποία ανήκουν σε κλάσεις που αντιστοιχούν σε διάφορους τύπους απαιτήσεων ή σε άλλα αντικείμενα του έργου όπως σχεδιαστικά στοιχεία ή στοιχεία ελέγχου. Καθε σχέδιο ορίζεται με την μορφή ένος διαγράμματος οντοτήτων συσχετίσεων. Η ιχνηλασιμότητα γίνεται από τον καθορισμό των δεσμών μεταξύ των αντικειμένων σε δύο κατηγορίες (ή εντός της ίδιας κατηγορίας), με βάση τις ταξικές σχέσεις που καθορίζονται στο σχήμα.

Έχει απόλυτη συμβατότητα με το Word και η μαχανή αναζήτησησης των για τον χειρισμό ιεραρχικών δέντρων δεδομένων είναι το ίδιο καλή με αυτή του Caliber-RM. Είναι πολύ ευέλικτο στην εισαγωγή απαιτήσεων από διάφορους τύπους εγγράφων και μπορεί να εξάγει τις απαιτήσεις σε πολλές μορφές αρχείων. Είναι το πιο αποτελεσματικό και χρηστικό από τα προηγούμενα. Είναι απλό τόσο στην ενσωμάτωση και την αλλαγή των απαιτήσεων και των χαρακτηριστικών τους όσο και στην διαγραφή τους.

Οποιαδήποτε από αυτά τα εργαλεία θα κάνει την διαχείριση των απαιτήσεων πιο εξελιγμένη και εύκολη. Διαφέρουν στην πλατφόρμα που χρησιμοποιούν, στην τιμή και στην προσέγγιση. Άλλα έχουν προσέγγιση στο κείμενο (document centric) και άλλα έχουν προσέγγιση στη βάση δεδομένων (database centric) [60].

***Λόγοι να χρησιμοποιήσουμε ένα εργαλείο διαχείρισης***

Ακόμη και αν έχουμε κάνει μία πολύ καλή δουλειά κατά την συγκέντρωση των απαιτήσεων του έργου, μία αυτοματοποημένη διαδικασία μπορεί να μας βοηθήσει να διαχειριστούμε τις απαιτήσεις κατά τη διάρκεια της ανάπτξης του έργου. Παρακάτων βλέπουμε κάποια διαχειριστικά καθήκοντα που τα εργαλεία αυτά μας βοηθούν να εκτελέσουμε.

**Διαχείρηση εκδόσεων και αλλαγών**. Το έργο θα πρέπει να προσδιορίσει μία βάση αναφοράς των απαιτήσεων, μία συγκεκριμένη συλλογή απαιτήσεων που θα περιέχει η συγκεκριμένη έκδοση. Το ιστορικό των αλλαγών που έχει γίνει σε κάθε απαίτηση θα εξηγήσει πώς πάρθηκε η συγκεκριμένη απόφαση και θα επιτρέπει να γυρίσουμε σε μία προηγούμενη έκδοση της απαίτησης αν κριθεί απαραίτητο.

**Αποθήκευση των χαρακτηριστικών των απαιτήσεων.** Πρέπει κάθε φορά να αποθηκευτούν πολλές πληροφοριές ή χαρακτηριστικά που αφορούν την κάθε απαίτηση. Ο οποισδήποτε δουλεύει στο έργο θα πρέπει να μπορεί να βλέπει τα χαρακτηριστικά και να τα ενημερώνει όποτε χρειαστεί. Τα εργαλεία διαχείρησης απαιτήσεων προσδίδουν στα χαρακτηριστικά την ημερομηνία που δημιουργήθηκαν, τον αριθμό της έκδοσής και διάφορους άλλους τύπους δεδομένων όπως τον συντάκτη τους, την καταγωγή τους, την κατάσταση τους, την προτεραιότητα τους , την σταθερότητα τους , το κόστος και τον κίνδυνο.

**Σύνδεση των απαιτήσεων με άλλα στοιχεία του συστήματος.** Η σύνδεση των απαιτήσεων με άλλα στοιχεία του συστήματος διασφαλίζει πως η ομάδα ανάπτυξης δεν θα παραβλέψει κατά λάθος κάποι απαίτηση. Μπορούμε να ορίσουμε διασυνδέσεις μεταξύ των διάφορων ειδών των

απαιτήσεων και μεταξύ των απαιτήσεων διάφορων υποσυστημάτων. Έτσι κατά την ανάλυση των επιπτώσεων της αλλαγής σε μία συγκεκριμένη απαίτηση οι συνδέσεις ιχνηλασιμότητας μας δείχουν και άλλα στοιχεία του συστήματος που επηρεάζονται από την αλλαγή αυτή.
**Παρακολούθηση της κατάστασης**. Η απαρακολούθηση της κατάστασης της κάθε απαίτησης κατά τη διάρκεια της ανάπτυξης βοηθά την συνολική παρακολούθηση ανάπτυξης του έργου. Αν ένας Διαχειριστής του Έργου ξέρει το ακριβές ποσοστό των απαιτήσεων που διατίθενται στην επόμενη έκδοση έχουν υλοποιηθεί κια ελεχθεί, το ακριβές ποσοστό που έχουν υλοποιηθεί αλλά δε έχουν ελεχθεί και το ακριβές ποσοστό που ούτε έχουν υλοποιηθεί και ελεχθεί, τότε έχει μία πολύ καλή εικόνα της συνολικής προόδου του έργου.

Επίβλεψη των υποσυνόλων των απαιτήσεων. Μπορούμε να ταξινομήσουμε, να φιλτράρουμε ή και να ρωτήσουμε την βάση δεδομένων έτσι ώστε να δούμε τα υποσύνολα των απαιτήσεων που έχουν συγκεκριμένες τιμές παραμέτρων.

Έλεγχος πρόσβασης. Μπορούμε να ορίσουμε δικαιώματα πρόσβασης σε ορισμένα άτομα ή σε ομάδες χρηστών. Η πρόσβαση μέσω διαδυκτίου μας επιτρέπει να μοιράζονται οι πληροφορίες των απαιτήσεων σε όλα τα μέλη της ομάδας ακόμα και ανα αυτά είναι γεωγραφικά διαμοιρασμένα.

Επικοινωνία με τους ενδιαφερόμενους φορείς. Τα περισσότερα εργαλεία διαχείρισης απαιτήσεων επιτρέπουν στις ομάδες να συζητούν για τα δεδομένα με ηλεκτρονική μορφή. Τα πρόσωπα που εμπλέκονται μπορούν να ενημερωθούν όταν γίνεται συζήτηση για μία νέα καταχώρηση ή για μία αλλαγή κάποιας απαίτησης με μυνήματα ηλεκτρονικού ταχυδρομείου [60,61].

## 4.5 Σύγκριση Εργαλείων Διαχείρισης Απαιτήσεων

Ο Πίνακας 18 συγκρίνει τα κύρια χαρακτηριστικά του προϊόντος για τα τέσσερα εργαλεία διαχείρησης απαιτήσεων που αναφέρθηκαν παραπάνω. Τα εργαλεία συνεχίζουν να εξελίσσονται, με αποτέλεσμα τα κύρια χαρακτηριστικά τους να αλλάζουν με τις μελλοντικές εκδόσεις. Όλα μας επιτρέπουν να ορίζουμε διαφορετικούς τύπους απαιτήσεων, όπως επιχειρισιακές απαιτήσεις, περιπτώσεις χρήσης, λειτουργικές και μη λειτουργικές απαιτήσεις, απαιτήσεις hardware και δοκιμές. Όλα τα εργαλεία είνξαι συμβατά με το MicrosoftWord σε κάποιο βαθμό. Τα πιο ακριβά εργαλεία υποστηρίζουν την εισαγωγή και την εξαγωγή μίας μεγάλης ποικιλίας αρχείων.

Μπορούμε να προσθέσουμε νέες απαιτήσεις κατευθείαν στη βάση δεδομένων, ή μπορούμε να επιλέηξουμε ένα συγκεκριμένο κομμάτι από ένα κείμενο και να το αντιμετωπίσουμε ώς μία ξεχωριστή απαίτηση. Μπορούν επίσης να αναλύσουν ένα Έγγραφο Προσδιορισμού των Απαιτήσεων ώστε να εξάγουμε μεμονομένες απαιτήσεις. Η μέθοδος ανάλυσης είναι ατελής αν δεν είμστε επιμελής σχετικά με την χρήση του κειμένου ή το στυλ του ή τις λέξεις κλειδιά όταν γράφουμε το Έγγραφο Προσδιορισμού των Απαιτήσεων. Τα εργαλεία υποστηρίζουν τις

ιεραρχικές αριθμιτικές ετικέτες των απαιτήσεων. Οι δυνατότητες εξόδου συμπεριλαμβάνουν τη διατήρηση του Εγγράφου Προσδιορισμού των Απαιτήσεων συγχρονισμένο με τη βάση δεδομένων των απαιτήσεων και τη δημιουργία ενός εγγράφου απαιτήσεων είτε σε μία μορφή που θα καθορίζεται από τον χρήστη είτε σε μορφή πίνακα.
Όλα τα εργαλεία έχουν ισχυρή ιχνηλασιμότητα συμπεριλαμβανομένης μία ένδειξης "ύποπτες απαιτήσεις" που μπορεί να επηρεαστούν αν μία απαίτηση με την οποία σχετίζεται αλλάξει ή διαγραφεί. Όλες επιτρέπουν την δημιουργία ομάδων χρηστών και των σχετικών δικαιωμάτων των ομάδων να διαβάζουν, να δημιουργούν ή να διαγράφουν projects, απαιτήσεις και τιμές παραμέτρων. Τα περισσότερα από αυτά είναι συμβατά σε εφαρμογές όπως φύλλα Microsoft Excel και σε εικόνες μέσα στις απαιτήσεις.

Τα εργαλεία αυτά έχουν μία τάση να ενσωματώνονται με άλλα εργαλεία ανάπτυξης εφαρμογών, όπως οι δοκιμές, τον σχεδιασμό των μοντέλων, τον εντοπισμό προβλημάτων και εργαλεία διαχείρησης του συνολικού έργου.

| **Εργαλείο** | **Κατασκευαστής - Προμηθευτής** | **Έμφαση σε έγγραφα ή σε βάσεις δεδομένων** |
|---|---|---|
| **DOORS 4.1** | Quality Systems and Software Inc. www.qssinc.com | Database-centric |
| **RTM Workshop 5.0** | Integrated Chipware www.chipware.com | Database-centric |
| **RequisitePro 4.0** | Rational Software www.rational.com | Document-centric |
| **Caliber-RM 1.1** | Technology Builders Inc. www.tbi.com | Database-centric |

**Πίνακας 17.** Κύρια εργαλεία διαχείρησης απαιτήσεων

| **Χαρακτηριστικά** | **DOORS** | **RTM Workshop** | **Caliber -RM** | **Requisite Pro** |
|---|---|---|---|---|
| Αναλύει το έγγραφο και αποθηκευέι τις απαιτήσεις στην Βάση Δεδομένων. | **x** | **x** | **x** | **x** |
| Εισάγει απαιτήσεις από κείμενο WORD στην Βάση Δεδομένων. | **x** | **x** | **x** | |
| Ενσωματώνει μη λεκτικά κείμενα, όπως φύλλα Excell και εικόνες στην Βάση Δεδομένων. | **x** | **x** | **x** | |
| Συγχρονίζει το περιεχόμενο του SRS με την Βάση Δεδομένων. | | **x** | | **x** |
| Καθορίζει χαρακτηριστικά στις διαφορετικού τύπου απαιτήσεις. | **x** | **x** | **x** | **x** |
| Ορίζει Baselines. | **x** | **x** | | |
| Εδοποιεί τους ενδιαφερόμενους για τις | **x** | **x** | **x** | |

| | | | | |
|---|---|---|---|---|
| αλλαγές που γίνονται. | | | | |
| Ορίζει διασυνδέσεις ιχνηλασιμότητας ανάμεσα στις απαιτήσεις και τα άλλα στοιχεία του συστήματος. | **x** | **x** | **x** | **x** |
| Χρηστικότητα. | **x** | **x** | **x** | **x** |
| Περιλαμβάνει εκπαιδευτικά βοηθήματα. | **x** | **x** | **x** | **x** |
| Διασυνδέεται με άλλα εργαλεία ελέγχου και σχεδιασμού. | **x** | **x** | **x** | **x** |
| Ορίζει προνόμια πρόσβασης στις διάφορες κατηγορίες χρηστών. | **x** | **x** | **x** | **x** |
| Περιλαμβάνει Web Interface. | **x** | **x** | **x** | **x** |
| Περιλαμβάνει σύστημα διαδικασίας αλλαγών. | **x** | **x** | | |

**Πίνακας 18.** Σύγκριση χαρακτηριστικών των κύριων εργαλείων διαχείρησης απαιτήσεων

Οποιοδήποτε από αυτά τα εργαλεία θα ανεβάσει την διαχείρηση των απαιτήσεων του έργου σε υψηλότερο επίπεδο. Η επιλογή όμως ενός τέτοιου εργαλείου δεν είναι μία εύκολη απόφαση και θα πρέπει να γίνει συνυπολογίζοντας τον τρόπο λειτουργίας και την κουλτούρα της κάθε εταιρείας. Υπαρχούν ειδικοί σύμβουλοι που αξιολογούν τις ανάγκες τις εταιρείας και προτείνουν το κατάλληλο εργαλείο για αυτή [60,61].

# 5 Μελέτη Περίπτωσης

Σκοπός της συγκεκριμένης μελέτης είναι να παρουσιαστεί η διαδικασία της μηχανικής των απαιτήσεων στο χώρο της υγείας και, συγκεκριμένα, κατά τη δημιουργία ενός συστήματος κατ'οίκον νοσηλείας. Θα παρουσιαστούν οι λειτουργικές και οι μη λειτουργικές απαιτήσεις του συστήματος, το περιβάλλον το οποίο θα χρησιμοποιείται και οι χρήστες του. Στην συνέχεια θα γίνει η μοντελοποίηση του συστήματος με την βοήθεια διαγραμμάτων οντοτήτων συσχετίσεων, διαγραμμάτων κλάσεων, διαγραμμάτων ροής δεδομένων και περιπτώσεων χρήσης.

Η υγεινομική περίθαλψη στο σπίτι είναι ζωτικής σημασίας τόσο για τους ασθενείς με χρόνια, όσο και για τους ασθενείς με οξεία προβλήματα υγείας. Υπάρχουν αρκετοί παράγοντες που αλλάζουν σημαντικά την προσέγγιση θεραπείας των χρόνιων παθήσεων. Τέτοιοι παράγοντες είναι οι νέες τεχνολογίες, οι βελτιώσεις στην φαρμακευτική βιομηχανία, ο σχεδιασμός των φαρμάκων και τα αιτήματα των ασθενών να παραμείνουν στα σπίτια τους. Η οικιακή περίθαλψη είναι κατάλληλη μόνο για τους ασθενείς που οι ιατρικές τους ανάγκες μπορούν να καλυφθούν με ασφάλεια στο σπίτι και ενώ έχουν συνυπολογιστεί τόσο οι απαιτούμενοι οικονομικοί πόροι, όσο και ο χρόνος που απαιτείται.

Ο αυξανόμενος αριθμός ηλικωμένων πολιτών, σημαίνει αυτόματα και μεγαλύτερη οικονομική επιβάρυνση στον χώρο της ιατρικής περίθαλψης. Η αύξηση της φροντίδας στο σπίτι είναι ένα υποκατάστατο της βασικής ιατρικής περίθαλψης και μπορεί να μειώσει σε μικρό ή μεγάλο βαθμό τα έξοδα στη υγεία. Με την βοήθεια αισθητήρων και ψηφιακών συστημάτων, οι ηλικιωμένοι μπορούν να συνεχίσουν να με μένουν στα σπίτια τους, ενώ παράλληλα να φροντίζουν από εκεί την υγεία τους. Δεν υπάρχει καμμία αμφιβολία ότι με τον τρόπο αυτό η ποιότητα ζωής τους θα είναι πολύ καλύτερη από το αν ήταν σε κάποιο ιατρικό κέντρο.

Βασισμένοι στα παραπάνω, διαπιστώθηκε πως είναι πολύ σημαντικό να αναπτυχθεί ένα σύστημα λογισμικού για κατ' οίκον νοσηλεία. Η Μηχανική των Απαιτήσεων ενός τέτοιου συστήματος είναι η πρώτη διεργασία που θα πρέπει να γίνει. Πρέπει να προσδιοριστούν οι στόχοι που θα υλοποιηθούν από το σύστημα, οι προδιαγραφές που θα έχει και οι περιορισμοί στους οποίους θα πρέπει να συμμορφώνεται. Θα πρέπει επίσης να διασφαλιστεί ότι θα ικανοποιηθούν οι ανάγκες των ενδιαφερομένων και για να γίνει αυτό θα πρέπει να οριστούν με ακρίβεια οι λειτουργικές και οι μη λειτουργικές απαιτήσεις. Η διαδικασία μέσω της οποίας προσδιορίζονται αυτές οι ανάγκες είναι η Μηχανική Απαιτήσεων. Προκειμένου η συγκεκριμένη διαδικασία να είναι επιτυχής, δεν αρκεί να αναγνωρίζονται οι ανάγκες των πελατών και των

χρηστώναλλά θα πρέπει επέρχεται πλήρης κατανόηση του πλαισίου εντός του οποίου θα λειτουργεί το λογισμικό. Η Μηχανική των Απαιτήσεων περιλαμβάνει την μοντελοποίηση, την ανάλυση, τη διαπραγμάτευση, την τεκμηρίωση, την επικύρωση και την διαχείριση των απαιτήσεων. Υπάρχει πληθώρα τεχνικών για την ανάπτυξη των προδιαγραφών των απαιτήσεων. Σε αυτές περιλαμβάνονται οι δομημένες, οι αντικειμενοστραφείς, οι αλγεβρικές, οι προτυποποίησης και οι μέθοδοι βασικών μοντέλων. Ένα κρίσιμο ζήτημα είναι η κατανόηση των απαιτήσεων των χρηστών. Η παρανόηση των απαιτήσεων μεταξύ των προγραμματιτσών του λογισμικού και των ενδιαφερόμενων μερών μπορούν να οδηγήσουν σε προβλήματα ικανοποίησης των αναγκών, σε απομόνωση και διόρθωση ελαττωμάτων και στην εκτίμηση του κόστους και των χρονοδιαγραμμάτων κατά τη διάρκεια της διαδικασίας της ανάπτυξης του λογισμικού [62].

Ένας από τους κύριους στόχους της Μηχανικής των Απαιτήσεων είναι ο εμπλουτισμός της μοντελοποίησης των συστημάτων και των δυνατοτήτων ανάλυσης έτσι ώστε οι επιχειρήσεις να μπορούν να κατανοήσουν καλύτερα τις κύριες πτυχές του συστήματος πριν την ανάπτυξή του. Οι λειτουργικές απαιτήσεις μαζί με τα χαρακτηριστικά ποιότητας και άλλες μη λειτουργικές απαιτήσεις δημιουργούν την προδιαγραφή των απαιτήσεων λογισμικού. Οι λειτουργικές απαιτήσεις είναι οι κύριες δυνατότητες του συστήματος. Αναπαριστούν το ¨τι¨ θα κάνει το σύστημα που θα αναπτυχθεί, χωρίς να αναφέρονται στον τρόπο με τον οποίο (¨πώς¨) το σύστημα θα το κάνει. Οι μη λειτουργικές απαιτήσεις είναι οι περιορισμοί που τίθενται στις λειτουργικές απαιτήσεις, ή στις απαιτήσεις ποιότητας. Αυτές περιλαμβάνουν πληθώρα ιδιοτήτων συμπεριλαμβάνοντας την επίδοση, τους περιορισμούς πολιτικής, την ασφάλεια, την προστασία προσωπικών δεδομένων, την αξιοπιστία. Καθορίζονται γενικά ως ένα βαθμό μετά την μοντελοποίηση των επιχειρηματικών διαδικασίων. Η μοντελοποίηση των μη λειτουργικών χαρακτηριστικών της επιχείρησης θεωρείται ως ένα δύσκολο πρόβλημα, καθώς η μοντελοποίηση επικεντρώνεται στην λειτουργική συμπεριφορά. Στις συνθήκες κατά τις οποίες εμπλέκονται ασθενείς, η ασφάλεια αποτελεί ένα κρίσιμο ζήτημα. Οι στόχοι της ασφάλειας εξελίσονται όταν τα εμπλεκόμενα μέρη, ως απτά ή μη απτά αντικείμενα, εντός του πλαισίου του συστήματος, απολαμβάνουν άμεση ή έμμεση αξία από το σύστημα. Η αξία αυτή προστατεύεται από τα ίδια τα εμπλεκόμενα μέρη. Με βάση αυτούς τους στόχους, οι απαιτήσεις της ασφάλειας προκύπτουν ως περιορισμοί των λειτουργικών απαιτήσεων. Προκειμένου η ασφάλεια των πληροφοριών να είναι αποτελεσματική, είναι απαραίτητο να δομείται κατά τη διάρκεια της διαδικασίας ανάπτυξης του λογισμικού. Οι κατάλληλες απαιτήσεις ασφαλείας, αυτές δηλαδή που ευθυγραμμίζονται με τους στόχους της επιχείρησης, είναι απαραίτητες για τον ασφαλή σχεδιασμό που βελτιώνει την ικανότητα του προϊόντος και την απόδοση της επένδυσης (ROI).

Η Μηχανική Απαιτήσεων για την φροντίδα υγείας στο σπίτι αποτελεί μία σκληρή διαδικασία, καθώς αφορά πληθώρα ενδιαφερόμενων μερών, συμπεριλαμβάνοντας τον ασθενή, τον κύριο γιατρό, τους άλλου γιατρούς, την νοσοκόμα, τον επίσημο φροντισή, τον ανεπίσημο φροντισή, τις κυβερνητικές και άλλες υπηρεσίες. Τα συστήματα φροντίδας υγείας στο σπίτι ενδιαφέρουν πληθώρα εμπλεκόμενων μερών με διαφορετικά ενδιαφέροντα. Το γεγονός αυτό μπορεί να

επηρεάσει την επίδοση, τη λειτουργία και τη συμπεριφορά του συστήματος με διαφορετικούς τρόπους. Κάθε συμβαλλόμενο μέρος του συστήματος έχει διαφορετική προοπτική, διαφορετικές επιθυμίες και διαφορετικές υποχρεώσεις. Αυτά περιπλέκονται περισσότερο όταν μεταβάλλονται οι συνθήκες και οι συμπεριφορές των εμπλεκόμενων μερών και του συστήματος.

Τα συστήματα κατ' οίκον νοσηλείας διακρίνονται από βασικά χαρακτηριστικά. Είναι συστήματα πολλαπλών χρηστών, πολλαπλών ενδιεφερομένων και πολλαπλών συνεργατών. Είναι ένα πολυτροπικό σύστημα αλληλεπίδρασης και οι ανάγκες του είναι δυναμικές. Επιπλέον οι συνθήκες περίθαλψης και φροντίδας είναι πολύπλοκες, πράγμα το οποίο την καθιστά την όλη διαδιακασία πολύ δύσκολη. Τα πληροφοριακά συστήματα κατ' οίκον νοσηλείας αναπτύσσονται συνεχώς και ξεχωρίζουν από τις άλλες υπηρεσίες, παροχής υπηρεσιών, με διάφορους τρόπους. Για την κατασκευή τέτοιων συστημάτων, θα πρέπει να γίνεται εμπεριστατωμένη ανάλυση και να πληρούνται συγκεκριμένες απαιτήσεις, έτσι ώστε να είναι ασφαλή και αξιόπιστα.

Υπάρχουν διάφορες τεχνικές μοντελοποίησης και αναπαράστασης των απαιτήσεων. Μερικές από αυτές τις προσεγγίσεις είναι: βασισμένες σε περιπτώσεις χρήσης, βασισμένες σε απόψεις, βασισμένες σε ανάλυση συμπεριφοράς προτύπων, βασισμένες στην αρχιτεκτονική του λογισμικού και βασισμένες σε τυπικές προσεγγίσεις.

Το συγκεκριμένο έγγραφο επιχειρεί να αναλύσει ένα σύστημα που παρέχει υπηρεσίας οικιακής περίθαλψης. Παρουσιάζονται τόσο οι λειτουργικές, όσο και οι μη λειτουργικές απαιτήσεις, καθώς επίσης και διάφοροι περιορισμοί που αφορούν αυτά τα συστήματα. Τέλος, διερευνούνται και παρουσιάζονται διαγραμματικά ορισμένα μοντέλα του συστήματος.

## 5.1 Λειτουργικές Απαιτήσεις

### 5.1.1 Λειτουργικότητα

Η λειτουργικότητα ενός συστήματος μετράται από το πόσο καλά ικανοποιεί τις λειτουργικές απαιτήσεις των ενδιαφερομένων. Παρακάτω υπάρχει μία λίστα τέτοιων απαιτήσεων:

- Το σύστημα θα πρέπει να επιτρέπει στους χρήστες να ενημερώνουν τις προσωπικές τους πληροφορίες.
- Το σύστημα θα πρέπει να επιτρέπει στον κύριο γιατρό να ενημερώνει πληροφορίες για τους ασθενενείς, όπως διάγνωση, κατάσταση υγείας και συνταγές.
- Το σύστημα θα πρέπει να επιτρέπει να στον γιατρό να βλέπει το αρχείο ενός συγκεκριμένου ασθενή.
- Το σύστημα θα πρέπει επιτρέπει στον γιατρό να καθορίζει ένα πλάνο περίθαλψης για κάποιον ασθενή,το οποίο θα περιέχει βραχυπρόθεσμους και μακροπρόθεσους στόχους βασισμένους στην κατάσταση του ασθενή.

- Θα πρέπει να επιτρέπει στον γιατρό να αλλάζει το πλάνο οικιακής περίθαλψης.
- Θα πρέπει να επιτρέπει στον γιατρό να γράφει συνταγές και να τις προωθεί στο φαρμακείο του ασθενή.
- Θα πρέπει να επιτρέπει στον γιατρό να βλέπει όλες τις ιατρικές συνταγές που έχουν χορηγηθεί στον ασθενή.
- Θα πρέπει να επιτρέπει στον γιατρό να βλέπει όλες τις αναφορές σχετικά με τον ασθενή.
- Θα πρέπει να επιτρέπει να στον γιατρό να κατηγοριοποιεί τους ασθενείς σε διάφορα επίπεδα, έτσι ώστε να δέχονται την κατάλληλη θεραπεία όταν προκύψει κάτι επείγον.
- Όταν θα υπάρξει κάποιαν επείγουσα ανάγκη, το σύστημα θα πρέπει να ειδοποιεί τον επιβλέπων γιατρό του ασθενή και αν χρειαστεί και τα διαθέσιμα κέντρα υγείας.
- Θα πρέπει να επιτρέπει στον γιατρό να βλέπει και να συνοψίζει την κατάσταση οικιακής περίθαλψης του ασθενή.
- Θα πρέπει να επιτρέπει στον γιατρό να κλείνει επισκέψεις με τον ασθενή, τόσο στον ιατρείο, όσο και στο σπίτι.
- Θα πρέπει να επιτρέπει την επικοινωνία μεταξύ του γιατρού και του προσώπου που προσέχει τον ασθενή στο σπίτι.
- Θα πρέπει να επιτρέπει στον γιατρό να ψάχνει και να βλέπει πληροφορίες για υπηρεσίες οικιακής περίθαλψης και να τις συστήνει στον ασθενή.
- Θα πρέπει να επιτρέπει την επικοινωνία μεταξύ του γιατρού και των υπηρεσιών οικιακής περίθαλψης.
- Το σύστημα θα πρέπεινα επιτρέπει στους παρόχους οικιακής περίθαλψης να βλέπουν και να καταγράφουν την θερμοκρασία, την πίση του αίματος και τα επίπεδα ζαχάρου στον οργανισμό του ασθενή χρησιμοποιώντας ιατρικά μηχανήματα και σένσορες.
- Θα πρέπει να επιτρέπει στους παρόχους οικιακής περίθαλψης να δέχονται την συνταγή που έχει γράψει ο γιατρός για κάποιον συγκεκριμένο ασθενή.
- Θα πρέπει να επιτρέπει στους ασθενείς να βλέπουν τις συνταγές που τους έχουν γράψει οι γιατροί και τις οδηγίες που τους έχουν δώσει.
- Θα πρέπει να επιτρέπει στα νοσοκομεία να έχουν πρόσβαση στους ιατρικούς φακέλους των ασθενών όποτε εκείνοι νοσηλεύονται για κάποιο λόγο σε αυτά.
- Το σύστημα θα πρέπει να δημιουργεί αποδείξεις και να τις προωθεί στις ασφαλιστικές εταιρείες, στην αντίστοιχη κυβερνητική υπηρεσία και στους ασθενείς.

***Δεδομένα***

Οι Απαιτήσεις Δεδομένων (DataRequirements) αντιστοιχούν στον τύπο των δεδομένων, την ακρίβειά τους και την μορφή τους. Παρακάτω υπάρχουν κάποια δεδομένα της οικιακής περίθαλψης.

- Σταθερά Πεδία Δεδομένων (Fixed data Fields): Τα πεδία αυτά περιλαμβάνουν μεταβλητές που μοιράζονται όλοι οι χρήστες , όπως όνομα, ημερομηνία γέννησης, διεύθυνση, αριθμός τηλεφώνου, ηλεκτρονική διεύθυνση, φύλλο και κωδικό ID.
- Ιατρικά Δεδομένα (Medical Data): Περιλαμβάνει δεδομένα διαγνώσεων και θεραπείας χρησιμοποιώντας ιατρικούς κώδικες και κείμενο.
- Δεδομένα Αισθητήρων (SensorsData): Τα δεδομένα αυτά περιλαμβάνουν πληροφορίες από διάφορους αισθητήρες, όπως θερμοκρασία, πίεση αίματος και επίπεδα ζαχάρου στον οργανισμό.
- Δεδομένα Εικόνων (ImageData): Στην κατηγορία αυτή υπάρχουν αποθηκευμένες εικόνες, όπως ακτινογραφίες (X-Rays) και εικόνες EKG.
- Αριθμητικά Δεδομένα (AccountingData): Εδώ υπάρχουν τα απαραίτητα δεδομένα για να δημιουργηθούν λογαριασμοί και τιμολόγια.
- Δεδομένα Υπηρεσιών (ServiceData): Στην κατηγορία αυτή υπάρχουν όλες οι πληροφορίες που χρειάζονται για τις κυβερνητικές υπηρεσίες και τις υπηρεσίες παροχής οικιακής περίθαλψης.

### 5.1.2 Περιορισμοί Σχεδιασμού

Οι σχεδιαστές πρέπει να δημιουργήσουν ένα σύστημα το οποίο θα προσαρμόζεται σε κάποιους περιορισμούς. Αν αυτοί οι περιορισμοί δεν συνυπολογιστούν τότε το σύστημα δεν θα λειτουργεί σωστά και θα καταλήξει σε αποτυχία.

***Φυσικό Περιβάλλον***

Το φυσικό περιβάλλον αναφέρεται στα εξαρτήματα και τις συσκευές που που θα χρησιμοποιεί το σύστημα, στις τοποθεσίες που θα λειτουργεί, στα συμπεράσματα - στις εξόδους δηλαδή - που θα παρέχει όσον αφορά την θερμοκρασία του ασθενή, την πίεση του αίματος και τον θόρυβο.
Επίσης αναφέρεται στους περιορισμούς της γλώσσας προγραμματισμού που θα χρησιμοποιηθεί, του λειτουργικού συστήματος, της βάσης δεδομένων, της τροφοδοσίας ρεύματος και στην θέρμασνη και την ψύξη που θα πρέπει να παρέχεται στις συσκευές. Παρακάτω φαίνονται αναλυτικότερα:

- Το σύστημα θα πρέπει να χρησιμοποιεί συσκευές και αισθητήρες για διάφορες μετρήσεις, όπως θερμοκρασία σώματος, πίεση αίματος και επίπεδα ζαχάρου στον οργανισμό.

- Οι συσκευές και οι αισθητήρες θα πρέπει να είναι εγκατεστημένοι στο σπίτι του ασθενή.
- Η θερμοκρασία του δωματίου θα πρέπει να είναι τέτοια που να επιτρέπει στις συσκευές και τους αισθητήρες να λειτουργούν σωστά.
- Ο κώδικας θα πρέπει να είναι γραμμένος στην γλώσσα προγραμματισμού Java.
- Το σύστημα διαχείρησης της Βάσης Δεδομένων θα πρέπει να είναι ίδιο με εκείνο που χρησιμοποιεί ο γιατρός.
- Το σύστημα θα πρέπει να λειτουργεί σε Windows 7, Linux και Unix.
- Οι υπολογιστές και οι συσκευές θα πρέπει να είναι συνδεδεμένες σε Σταθεροποιητές Τάσης ( UninterruptedPowerSupply – UPS).
- Μπορεί να χρειαστεί και η χρήση γεννήτριας, αναλόγως με την κατάσταση του ασθενή.
- Θα πρέπει να συνδεθεί στο σπίτι και μία συσκευή συναγερμού έκτακτης ανάγκης.

***Διεπαφές Χρήστη***

Οι διεπαφές χρήστη περιλαμβάνουν όλες τις διασυνδέσεις μεταξύ των ανθρώπων, των υπολοίπων συστημάτων λογισμικού και των διάφορων συσκευών του συστήματος οικιακής περίθαλψης. Θα πρέπει να είναι σαφή, εύκολα στην χρήση, ευέλικτα και να απευθύνονται στα διάφορα επίπεδα δεξιοτήτων και εξωτερικού περιβάλλοντος. Παρακάτω υπάρχουν κάποιες διαπεφές χρήστη που θα συμπεριληφθούν:

- Το σύστημα θα πρέπει να παρέχει διάφορες τεχνικές διεπεφών συμπεριλαμβανομένων της GUI, φωνής και οθόνες αφής.
- Το σύστημα θα πρέπει να προσαρμόζει τις διεπαφές ανάλογα με τις ανάγκες των χρηστών και να εξασφαλίζει την ιδιωτικότητα και την ασφάλεια των πληροφοριών των ασθενών.
- Το σύστημα θα πρέπει να συνδέεται με τα συστήματα των νοσοκομείων.
- Θα πρέπει να συνδεέται με τα συστήματα των κυβερνητικών υπηρεσιών.
- Θα πρέπει να συνδέεται με τις υπηρεσίες παροχής οικιακής περίθαλψης.
- Θα πρέπει να συνδέεται με τα εργαστήρια για να γίνεται αυτόματα η λήψη των αποτελεσμάτων.
- Θα πρέπει να συνδέεται άμμεσα τις μονάδες ακτινογραφίας και εικόνων EKG, για να λαμβάνονται αμμέσως τα αποτελέσματα.

***Χρήστες***

Ένας από τους λόγους της πολυπλοκότητας του συστήματος οικιακής περίθαλψης είναι ο μεγάλος αριθμός ενδιαφερομένων που εμπλέκονται και διαφορετικές απόψεις που έχει ο κάθε ένας από αυτούς. Οι ενδιαφερόμενοι είναι τα νοσοκομεία, οι γιατροί, οι ασθενείς, οι πάροχοι υπηρεσιών υγείας, οι νοσηλευτές, οι ασφαλιστικές εταιρείες, το ΕΚΑΒ (166), τα φαρμακεία και οι άτυποι πάροχοι οικιακής περίθαλψης – ένας συγγενής ή ένας γείτονας που έχει δυνατότητες να προσφέρει βοήθεια στον ασθενή.

**Περιορισμοί Διαδικασίας**

***Πόροι***

Οι πόροι είναι ένας κρίσιμος παράγοντας επιτυχίας ενός τέτοιου συστήματος. Περιλαμβάνουν προσωπικό ηλεκτρονικών υπολογιστών, εξοπλισμό και εργαλεία που χρειάζονται για την συνεχή λειτουργία του συστήματος. Τέτοια παραδείγματα είναι:

- Μηχανικοί λογισμικού.
- Ομάδα συστήρησης πληροφοριακού συστήματος.
- Μηχανικοί δικτύων.
- Γιατροί, πάροχοι υπηρεσιών οικιακής περίθαλψης και άτυποι πάροχοι υπηρεσιών περίθαλψης.
- Σταθερούς και φορητούς υπολογιστές.
- Εξυπηρετητές (Servers).
- Ιατρικές συσκευές.
- Μηχανικούς συντήρησης ιατρικών συσκευών.
- Μπαταρίες για τα διάφορα ιατρικά μηχανήματα.

***Τεκμηρίωση***

Μία φτωχή τεκμηρίωση θα επιδράσεια αρνητικά στην λειοτουργία του συτήματος, ανεξάρτητα από το πόσο αποδοτικό και αξιόπιστο είναι αυτό. Η τεκμηρίωση των απαιτήσεων θα πρέπει να συνυπολογίσει τις ανάγκες των διάφορων χρηστών του συστήματος. Κάποιοι τύποι τεκμήριωσης που θα χρειαστούν είναι οι ακόλουθοι:

- Μία online βόηθεια χρήσης για κάθε χρήστη.
- Online αντίγραφα και εγχειρίδια χρήσης για κάθε χρήστη.
- Ένα Έγγραφο Προσδιορισμού Απαιτήσεων.
- Ένα έγγραφο απαιτήσεων λογισμικού.
- Τεκμηρίωση της αρχιτεκτονικής.
- Τεκμηρίωση της σχεδίασης.
- Τεκμηρίωση του ελέγχου και των περιπτώσεων δοκιμών.
- Τεκμηρίωση συντήρησης των διαδιακασιών.

- Εγχειρίδια προγραμματισμού.
- Τεκμηρίωση διεπαφών συμπεριλαμβανομένων των συσκευών και των διεπαφών των αισθητήρων.

***Πρότυπα***

Τα συστήματα θα πρέπει να τηρούν κάποιους εξωτερικούς και εσωτερικούς κανόνες. Αυτοί μπορεί να περιλαμβάνουν νομικά, ηθικά και περιβαλοντολογικά πρότυπα. Το σύστημα οικιακής περίθαλψης θα πρέπει να ακολουθεί τα εξής πρότυπα:

- Το σύστημα θα πρέπει να περιέχει τους διαγνωστικούς κώδικες ICD-9-CM της Διεθνής Στατιστικής Ταξινόμησης Ασθενειών και Συναφών Προβλημάτων Υγείας (International Statistical Classification of Diseases and Related Health Problems (ICD), που απαιτούνται για να συμπληρωθούν τα Τυποποιημένα Έντυπα Απαιτήσεων (Standard Claim Forms).
- Το σύστημα θα πρέπει να ακολουθεί τις οδηγίες του Medicare.
- Το σύστημα θα πρέπει να ακολουθεί το πρότυπο πληρωμών ProspectivePaymentSystem.
- Το σύστημα θα πρέπει να τηρεί τους κανόνες προστασίας της ιδιωτικής ζωής του Φορέα Ασφάλισης Υγείας και Πράξης Ευθύνης ( Health Insurance Portability and Accountability Act – HIPPA).

## 5.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις αντιπροσωπεύουν τους περιορισμούς και την ποιότητα που θα πρέπει να τηρεί και να παρέχει το σύστημα. Απεικονίζουν τα χαρακτηριστικά ποιότητας που πρέπει να έχουν τα συστήματα οικιακής περίθαλψης, όπως επίδοση, ασφάλεια, προστασία των ιδιωτικών δεδομένων και της αξιοπιστίας.

### 5.2.1 Επίδοση (Performance)

Η επίδοση έχει να κάνει με περιορισμούς της ταχύτητας που θα πρέπει να εκτελούνται οι διεργασίες, την ποσότητα των δεδομένω που θα αποθηκεύονται και τους χρόνους απόκρισης του συστήματος. Παρακάτω υπάρχουν κάποιοι τέτοιοι περιορισμοί:

- Το σύστημα θα πρέπει να επιτρέπει στον χρήστη να εισέρχεται κάθε 5 δευτερόλπετα.
- Θα πρέπει να εμφανίζει την λίστα με το αρχείο του ασθενή μέσα σε 30 δευτερόλεπτα απόόταν του ζητηθεί.
- Θα πρέπει να ανακτά τις πληροφορίες του ασθενή σε 5 δευτερόλεπτα.
- Θα πρέπει να εμφανίζει μία λίστα με τις λεπτομέρειες τις οικιακής περίθαλψης κάποιου συγκεκριμένου ασθενή, όπως πλάνο περίθαλψης, συνταγές και αναφορές μέσα σε 5 δευτερόλεπτα.
- Θα πρέπει να στένει την συνταγή στον φαρμακοπειό μέσα σε 15 δευτερόλεπτα.
- Η αναφορά της εξέτασης του ασθενή θα πρέπει να ανακτάται σε 5 δευτερόλεπτα.
- Εάν υπάρξει έκτακτη ανάγκη θα πρέπει να σημαίνεται συναγερμός σε 2 δευτερόλεπτα.

### 5.2.2 Χρηστικότητα και Ανθρώπινοι Παράγοντες (Usability and Human Factors)

Στην συνέχεια παρουσιάζονται οι παράγοντες που έχουν να κάνουν με την εκπαίδευση των χρηστών και το πόσο έυκολα μπορεί να χρησιμοποιηθεί το σύστημα ως συνάρτηση αυτής της εκπαίδευσης.

- Οι γιατροί θα πρέπει να εκπαιδευτούν στο να ενημερώνουν – τροποποιούν, διαγράφουν, καταχωρούν – δεδομένα στα αρχεία των ασθενών.
- Οι πάροχοι οικιακής περίθαλψης θα πρέπει να μπορούν να χρησιμοποιούν όλες τις συσκευές και τους αισθητήρες.
- Οι πάροχοι οικιακής περίθαλψης θα πρέπει να μπορούν να εισάγουν πληροφορίες από τα αρχεία των ασθενών.
- Οι χειριστές του οικονομικού κομματικού θα πρέπει να είναι σε θέση να κόβουν τιμολογία και λογαριασμούς τόσο για τοςυ ασθενείς, όσο και για τις ασφαλιστικές.
- Οι εταιρείες οικιακής περίθαλψης θα πρέπει να μπορούν να έχουν πρόσβαση σε πληροφορίες σχετικές με τις υπηρεσίες που δέχονται οι ασθενείς.
- Θα πρέπει τόσο οι ασθενείς, όσο και οι πάροχοι υπηρεσιών υγείας να μπορούν να σημάνουν τον συναγερμό σε περίπτωση ανάγκης.
- Εφόσον οι γιατροί θα έχουν να κάνουν με ευαίσθητα προσωπικά δεδομένα των ασθενών, θα πρέπει η εκπαίδευσή τους να τους αποτρέπει από καταστροφικές συνέπειες.
- Η οποιαδήποτε κατάχρηση των συσκευών και των αισθητήρων θα δημιουργήσει αυξημένα τιμολόγια. Έτσι, η εκπαίδευση των ασθενών και των παρόχων περίθαλψης θα τους βοηθήσει να τα χρησιμοποιούν συνετά.

### 5.2.3 Ασφάλεια (Security)

Η ασφάλεια είναι ένας κρίσιμος παράγοντας για την συγκεκριμένη εφαρμογή. Στο σύστημα αυτό συνεργάζονται πολλοί οργανισμοί συνπεριλαμβανομένων των ιατρικών κέντρων, κυβερνητικών υπηρεσιών, ασφαλιστικών εταιρειών και φαρμακείων. Αυτοί οι οργανισμοί έχουν ορίσει αυστηρά μέτρα ασφάλειας. Οι απαιτήσεις ασφάλειας θα διασφαλίσουν ότι τα μέτρα αυτά λαμβάνονται υπόψη. Επίσης θα εντοπιστούν τα τρωτά σημεία και θα αξιοποιηθούν από τυχόν επιθέσεις που θα θέλουν να το βλάψουν. Μερικοί από τους περιορισμούς ααφάλειας είναι οι εξής:

- Το σύστημα θα πρέπει να χορηγεί ένα μοναδικό αναγνωριστικό, ένα όνομα και έναν κωδικό σε κάθε εξουσιοδωτημένο χρήστη.
- Θα πρέπει να αυθεντικοποιεί τον κάθε ασθενή μέσω των παραπάνω γνωρισμάτων.
- Θα πρέπει να ελέγχει τα εμπλεκόμενα μέρη πριν ξεκινήσει μία επικοινωνία μεταξύ τους.
- Δεν θα πρέπει να εισβάλουν τρίτοι στην επικοινωνία δύο χρηστών.

- Η συνταγή που θα στέλνεται στα φαρμακεία, θα πρέπει να στέλνεται με έναν αυστηρά καθορισμένο τρόπο.
- Θα πρέπει να διασφαλιστεί ότι το αντίγραφο της συνταγής, εφόσον ζητηθεί από τον χρήστη, θα μεταφέρεται με ασφαλή τρόπο.
- Με τον ίδιο τρόπο θα πρέπει να διασφαλίζεται και η μεταφορά των λογαριασμών και των τιμολογίων στους εκάστοτε χρήστες.
- Εάν ο ασθενής είναι ασφαλισμένος, τότε η απόδειξη θα πρέπει να στέλνεται και στην ασφαλιστική του εταιρεία με ασφαλή τρόπο.
- Θα πρέπει να διασφαλίζονται τα στοιχεία των πιστωτικών καρτών των ασθενών, όταν αυτές χρησιμοποιούνται για την αποπληρωμή μίας οφειλής.
- Τα αρχεία των ασθενών θα πρέπει να επεξεργάζονται αυτηρά και μόνο από τους προσωπικούς τους γιατρούς.

### 5.2.4 Προστασία προσωπικών πληροφοριών (Privacy)

Τα προσωπικά δεδομένα των ασθενών είναι είναι προστατευμένα βάση νόμουυ. Η επεξεργασία των πληροφοριών θα πρέπει αν γίνεται με εμπιστευτικό τρόπο. Τέτοιες απαιτήσεις είναι οι παρακάτω:

- Θα πρέπει να διασφαλίζεται η επικοικινωνία μεταξύ των γιατρών και των ασθενών.
- Θα πρέπει να διασφαλίζονται τα αποτελέσματα των εξετάσεων και να στέλνονται με εμπιστευτικότητα τόσο στον ασθενή, όσο και στους άτυπους παρόχους οικιακής περίθαλψης.
- Θα πρέπει να διασφαλίζεται η συνταγογράφηση που πραγματοποιείται.
- Η επικοινωνία γιατρού και ασθενή για τα αποτελέσματα των εξετάσεων θα πρέπει να γίνεται άκρως εμπιστευτικά.
- Στις ασφαλιστικές εταιρείες θα πρέπει να στέλνονται μόνο οι πληροφορίες που τις αφορούν.
- Στις υπηρεσίες παροχής οικιακής περίθαλψης θα πρέπει να στέλνονται μόνο οι πληροφορίες που τις αφορούν.
- Στους κυβερνητικούς οργανισμούς που σχετίζονται με την οικιακή περίθαλψη θα πρέπει να στέλνονται πληροφορίες που έχουν να κάνουν με τις υπηρεσίες που προσφέρονται στους ασθενής και στοιχεία για τους λογαριασμούς.
- Οι εργαζόμενοι των εργαστηρίων θα πρέπει να μπορούν να εισάγουν τα αποτελέσματα στο σύστημα, αλλά δεν θα πρέπει να είναι σε θέση να βλέπουν τις υπόλοιπες ιατρικές πληροφορίες.

### 5.2.5 Αξιοπιστία και Διαθεσιμότητα (Reliability and Availability)

Τα ποιοτικά συστήματα οικιακής περίθαλψης δεν θα πρέπει να έχουν αποτυχίες ή αν έχουν θα πρέπει να γίνονται ανά μεγάλα χρονικά διαστήματα. Θα πρέπει να έχουν μεγάλο βαθμό

αξιοπιστίας επειδή διαχειρίζονται ευαίσθητες πληροφορίες. Οι απαιτήσεις αξιοπιστίας και διαθεσιμότητας μπορούν να διασφαλίσουν την αποφυγή των ανεπιθύμητων επιπτώσεων ενός μη αξιόπιστου συστήματος. Τέτοιες απαιτήσεις είναι:

- Το σύστημα θα πρέπει να είναι διαθέσιμο 24 ώρες την μέρα και 365 μέρεες τον χρόνο.
- Οι συσκευές και οι αισθητήρες στο σπίτι του ασθενή θα πρέπει είναι διαθέσιμοι 24 ώρες την μέρα και 365 μέρεες τον χρόνο.
- Το σύστημα θα πρέπει να εντοπίζει και να απομονώνει τα σφάλματα και να είναι σε θέση να τα διορθώνει.
- Ο μέσος χρόνος μεταξύ σφαλμάτων (Mean-Time-Between-Failures - MTBF) θα πρέπει να είναι τουλάχιστον 6 μήνες.
- Το σύστημα θα πρέπει να επανακινείται μέσα σε 15 λεπτά από κάθε σφάλμα.
- Θα πρέπει να δημιουργουνται αντίγραφα των πληροφοριών (BackedUp) σε καθημερινή βάση.
- Τα αντίγραφα θα πρέπει να αποθηκεύονται σε έναν εξωτερικό σκληρό δίσκο και να σώζονται σε διαφορες τοποθεσίες για να μπορούν να προστατεύονται από πλημμύρες ή φωτιές.

### 5.2.6 Συντηρησιμότητα (Maintainability)

Τα συστήματα υγείας μπορεί να αντιμετωπίσουν ελλαττώματα που να μην ανιχνεύονται από τις διάφορες τεχνικές ελέγχου που χρησιμοποιούνται. Επίσης μπορεί με την πάροδο με του χρόνου να δημιουργηθούν νέα εξαρτήματα στα οποία να μην έχει προβλευθεί κάποιος έλεγχος. Αν για την ενσωμάτωση αυτών των νέων εξαρτημάτων χρειαστεί να γίνει η ανάπτυξη του συστήματος από την αρχή, τότε μπορεί να δημιουρφηθούν προβλήματα που έχουν να κάνουν με την συντηρησιμότητα. Για να διασφαλιστεί αυτό το πρόβλημα είναι απαραίτητες οι παρακάτω απαιτήσεις:

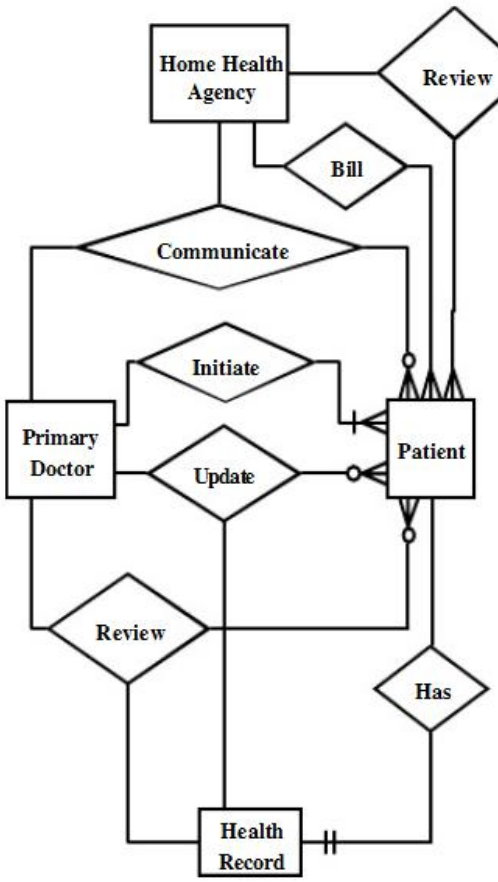
- Το σύστημα θα πρέπει να είναι αρκετά ευέλικτο για να επιτρέπει την διόρθωση των σφαλμάτων.
- Θα πρέπει να είναι αρκετά ευέλικτο για να επιτρέπει μελλοντικές βελτιώσεις.
- Θα πρέπει να μπορεί να συντηρείται από μόνο του όποτε είναι εφικτό.
- Θα πρέπει να μπορεί να λειτουργεί για τουλάχιστον ένα χρόνο χωρίς να γίνει κάποια βελτίωση.
- Να μπορεί να προστεθεί λειτουργικότητα μετά τον πρώτο χρόνο λειτουργίας του.
- Θα πρέπει να έχει υποστήριξη εφεδρικής μονάδας ρεύματος για να αποφευχθεί η απώλεια λειτουργίας του.
- Θα πρέπει να μπορεί να δημιουργεί αναφορές, να τις αποθηκεύει σε ένα αρχείο, να τις εμφανίζει στην οθόνη και να μπορεί να τις τυπώνει.

## 5.3 Μοντελοποίηση του Συστήματος

Οι απαιτήσεις του συστήματος οικιακής περίθαλψης μπορούν να αναπαραστηθούν με γραφικά μοντέλα, που περιγράφουν τις επιχειρισιακές διεργασίες. Είναι η γέφυρα μεταξύ των διαδικασιών της ανάλυσης και της σχεδίασης. Παρέχουν μία δομή του τι θα κατασκευαστεί και πώς θα αυτό θα προσαρμοστεί στον όποιον οργανισμό. Παρακάτω υπάρχουν εικόνες από τέτοια μοντέλα, που έχουν γίνει με την χρήση της αρχιτεκτονικής Rational Software Architect της IBM.

### 5.3.1 Διάγραμμα Οντοτήτων Συσχετίσεων

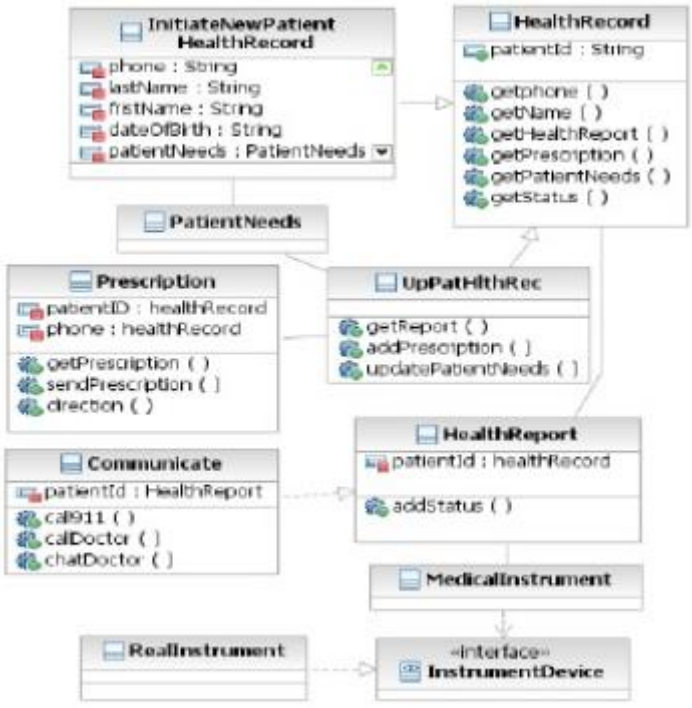
Το μοντέλο του Σχήματος 62 που παρουσιάζει τις οντοτήτες που εμπλέκονται και τις μεταξύ τους διεργασίες.




**Σχήμα 62 .** Διάγραμμα Οντοτήτων Συσχετίσεων Συστήματος Οικιακής Περίθαλψης

### 5.3.2 Διάγραμμα Κλάσεων

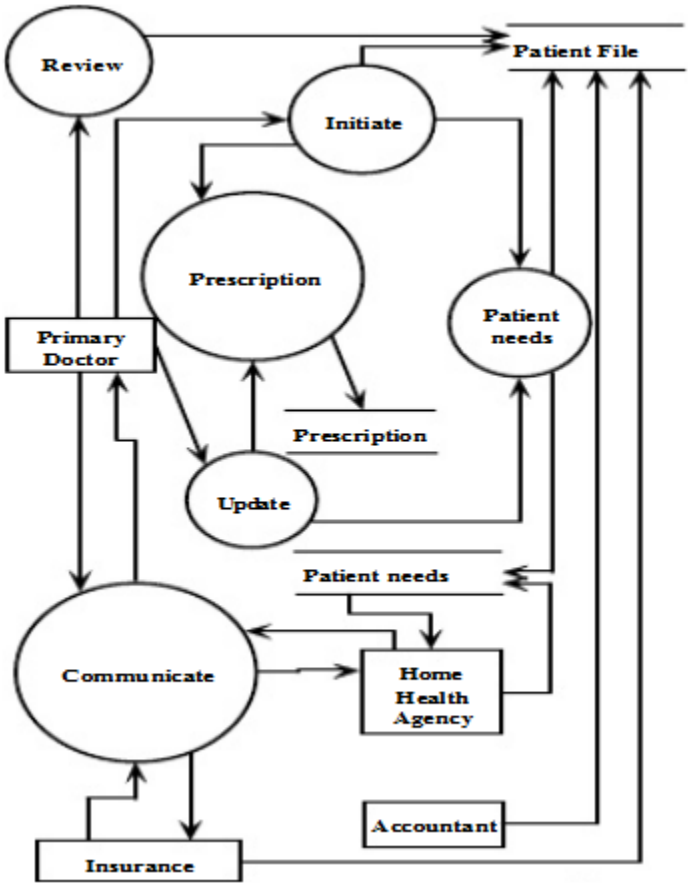
Στο Σχήμα 63 φαίνονται οι κλάσεις και οι οι μεταξύ τους διασυνδέσεις.




**Σχήμα 63.** Διαγράμματα Κλάσεων Συστήματος Οικιακής Περίθαλψης

### 5.3.3 Διάγραμμα Ροής Δεδομένων

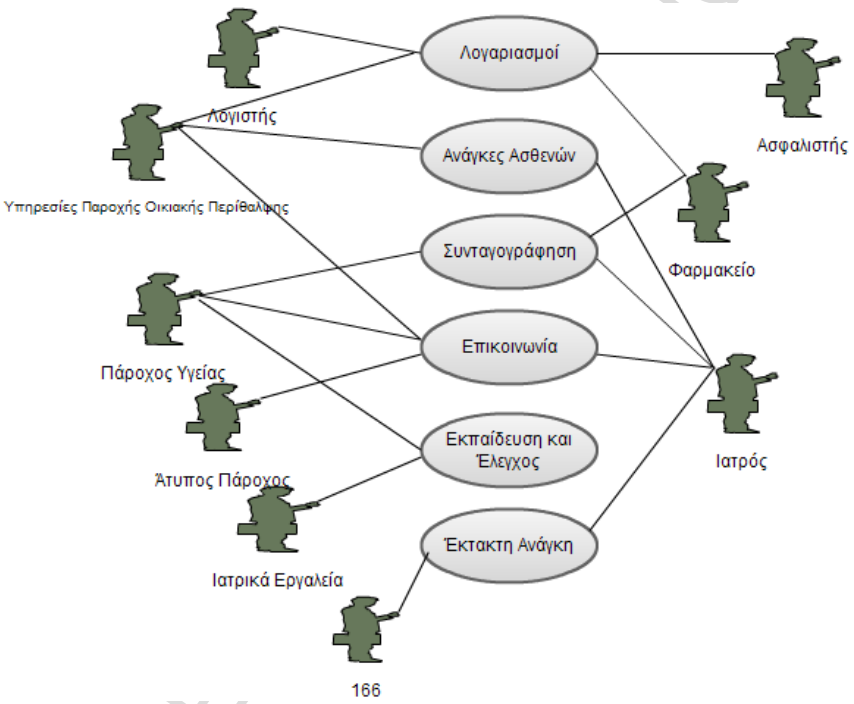
Στο διάγραμμα του Σχήματος 64 φαίνεται πώς μεταφέρεται η πληροφορία μεταξύ των διάφορων οντοτήτων και της λειτουργικότητας του συστήματος.



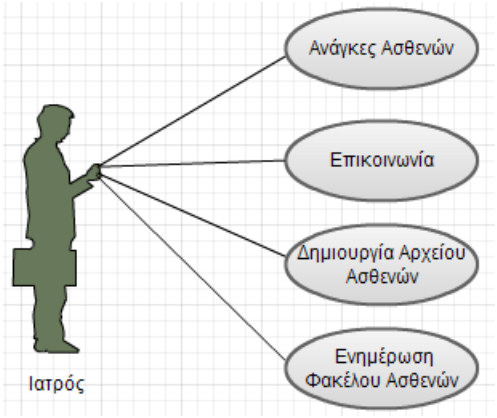

**Σχήμα 64.** Διάγραμμα Ροής Δεδομένων Συστήματος Οικιακής Περίθαλψης

### 5.3.4 Περιπτώσεις Χρήσης

Οι περιπτώσεις χρήσης, Σχήματα 65 και 66, είναι τα διάφορα πιθανά σενάρια επικοινωνίας και αλληλεπίδρασης του συστήματος με το εξωτερικό του περιβάλλον.




**Σχήμα 65**. Διάγραμμα Περίπτωσης Χρήσης Συστήματος Οικιακής Περίθαλψης




**Σχήμα 66.** Διάγραμμα Περίπτωσης Χρήσης Ιατρού



# 6 Σύνοψη - Χρησιμότητα Μηχανικής Απαιτήσεων

Στην παρούσα διπλωματική εργασία παρουσιάστηκε η διαδικασία της Μηχανικής των Απαιτήσεων για την ανάπτυξη λογισμικού πληροφοριακών συστημάτων. Αρχικά καταγράφηκαν τα υπάρχοντα μοντέλα κύκλου ζωής του λογισμικού και οι μεθοδολογίες ανάπτυξης πληροφοριακών συστημάτων. Ορίστηκαν τα επίπεδα απαιτήσεων και οι προκλήσεις στις οποίες πρέπει να απαντήσει η Μηχανική των Απαιτήσεων. Αναλύθηκαν οι διαδικασίες της εκμαίευσης, της ανάλυσης, της προδιαγραφής, της επικύρωσης και της διαχείρισης που την απαρτίζουν και έγινε εκτενή αναφορά στις μεθοδολογίες και τα εργαλεία που χρησιμοποιεί η κάθε μία.

Παρουσιάστηκε μία μελέτη περίπτωσης, αναφορικά με υπηρεσίες υγείας που προσφέρονται στο σπίτι, στην οποία φαίνονται πώς χρησιμοποιείται πρακτικά η Μηχανική των Απαιτήσεων κατά την δημιουργία ενός πληροφοριακού συστήματος. Αναλύθηκαν οι λειτουργικές, οι μη λειτουργικές απαιτήσεις και οι περιορισμοί του συστήματος, ενώ παρουσιάστηκαν οι διεπαφές του χρήστη και τα πρότυπα που έπρεπε να χρησιμοποιηθούν. Στην συνέχεια έγινε η μοντελοποίηση του συστήματος με την βοήθεια Διαγραμμάτων Οντοτήτων Συσχετίσεων, Διαγραμμάτων Κλάσεων, Διαγραμμάτων Ροής Δεδομένων και παρουσιάστηκαν τα πιθανά σενάρια επικοινωνίας και αλληλεπίδρασης του συστήματος με το εξωτερικό του περιβάλλον χρησιμοποιώντας περιπτώσεις χρήσης.

Αναδείχθηκε η αναγκαιότητα της Μηχανικής των Απαιτήσεων για την δημιουργία ενός πληροφοριακού συστήματος και επισημάνθηκε η σημαντικότητα των απαιτήσεων στον χρονοπρογραμματισμό του έργου και στην κατανομή των πόρων, στην αρχιτεκτονική που θα ακολουθηθεί, στην σχεδίασμό του συστήματος, αλλά και στον τελικό του έλεγχο.

Όσον αφορά τα συστήματα κατ' νοίκον νοσηλείας, η Μηχανική των Απαιτήσεων αποκτά ακόμα μεγαλύτερη αναγκαιότητα και σημασία. Με την αύξηση του ποσοστού των ηλικιωμένων που χρειάζονται φροντίδα στο σπίτι, η διαδικασία της Μηχανικής θα πρέπει να διασφαλίσει ότι θα ικανοποιηθούν όλες οι ανάγκες των χρηστών και να αποφευχθούν τυχόν μοιραίες συνέπειες από την εφαρμογή του συστήματος. Οι απαιτήσεις του συστήματος αποτελούν έναν οδηγό για όλες τις διαδικασίες, συμπεριλαμβανομένων του σχεδιασμού, του ελέγχου και της συντήρησης. Εάν, λοιπόν, οι απαιτήσεις είναι ελλιπείς, τότε αυτομάτως θα καθιστούν ελλιπείς και οι υπόλοιπες εξαρτώμενες, απο αυτές, διαδικασίες.

Αναφορικά με το χρόνο που πρέπει να αφιερώνεται ώστε να ολοκληρώνεται αποτελεσματικά το στάδιο της ανάπτυξης των απαιτήσεων, έχει παρατηρηθεί ότι: Σε περιπτώσεις κατά τις οποίες, το στάδιο ανάπτυξης των απαιτήσεων έιχε μεγάλη διάρκεια, η ανάπτυξη του έργου, στο σύνολό της, είχε σημαντικά μειωμένη διάρκεια σε σύγκριση με άλλες περιπτώσεις έργων, κατά την ανάπτυξη των οποίων, η διάρκεια του σταδίου ανάπτυξης των απαιτήσεων δεν ήταν, αντίστοιχα, μεγάλη.

Το απόλυτο παραδοτέο ενός έργου ανάπτυξης λογισμικού είναι ένα σύστημα που ανταποκρίνεται στις προσδοκίες των πελατών. Ένα σημαντικό βήμα που οδηγεί στην επιτυχία είναι η σωστή ανάπτυξη των κατάλληλων απαιτήσεων. Εάν ο σχεδιασμός του λογισμικού και η διαδικασία του ελέγχου δεν βασίζονται σε υψηλής ποιότητας απαιτήσεις, ο κίνδυνος αποτυχίας είναι πολύ μεγάλος. Από την άλλη είναι σημαντικό να δίνεται ιδιαίτερη προσοχή ώστε, οι απαιτήσεις να είναι εντός του πεδίου εφαρμογής του έργου και να υιοθετούν τους περιορισμούς του έργου και τους επιχειρηματικούς κανόνες. Σε κάθε άλλη περίπτωση, είναι πολύ πιθανό να αφιερώνεται πολύτιμος χρόνος στην ανάλυσή τους, χωρίς ουσιαστικό αποτέλεσμα. Η δημιουργία τέλειων και τεκμηριωμένων απαιτήσεων είναι κάτι ουτοπικό ζήτημα. Τονίζεται σε αυτό το σημείο ότι, η ισορροπία κατά την την τεκμηρίωση των απαιτήσεων έργων ανάπτυξης Πληροφοριακών Συστημάτων, μειώνει την πιθανότητα ενδεχόμενης αποτυχίας του έργου στις επιμέρους φάσεις, καθώς και στο σύνολό του.

## Βιβλιογραφία


1. Βεσκούκης Β. Τεχνολογία Λογισμικού Πειραιάς; 2000.
2. Σκορδαλάκης Ε. Λογισμική Μηχανική Αθήνα; 2003.
3. http://www.ct.aegean.gr. [Online]. [cited 2012. Available from: http://www.ct.aegean.gr/people/t.daradoumis/4ETDE104/Paradeigmata%20Ekpaideytikoy%20Ylikou/3-Kefalaio_me%20sxolia.pdf.
4. Πάγκαλος. Δομημένος Προγραμματισμός Θεσσαλονίκη.
5. Βεσκούκης Β. Τεχνολογία Λογισμικού ΙΙ Πάτρα: ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ; 2001.
6. P Z. Classification of Reasearch Effortsin Requirements Engineering; 1997.
7. Shams Ul Arif QKAKG. Requirements Engineering Processes, Tools/Technologies, & Methodologies. 2010;(International Journal of Reviews in Computing).
8. Kauppinen Marjo VMKJKSRS. Implementing requirements engineering processes throughout organizations: success factors and challenges. Information and Software Technology. 2004.
9. Maguire M, Bevan N. User requirements analysis. 2002;(World Computer Congress).

10. Ιωαννίδης Ι, Λέπουρας Γ. Σημειώσεις Επικοινωνίας Ανθρώπου-Μηχανής Αθήνα; 2005.

11. Elizabeth Hull, Ken Jackson, Jeremy Dick. Requirements Engineering. Second Edition ed. United States of America: Springer; 2004.

12. Kontonya Gerald, Sommervile Ian. REQUIREMENTS Engineering PROCESSES AND TECHNIQUES: Wiley; 2003.

13. IEEE Recommended Practice for Software Requirements SpeciÞcations; 1998.

14. Khaled M. Khan, Mahesha Kapurubandara, Urvashi Chadha. Incorporating Business Requirements and Constraints in Database Conceptual Models. Australia: School of Computing and Information Technology University of Western Sydney.

15. Bauer E. The Business Rule Approach. University of Paderborn.

16. Mahmud ZB. Integrating Training Needs Analysis In Existing Staff Management Information System Malaysia; 2002.

17. Brackett JW. Software Requirements. Boston: Boston University; 1990.

18. Young RR. Twelve Requirements Basics for Project Success. The Journal of Defense Software Engineering. 2006 December.

19. Cleland-Huang J. Software Requirements. DePaul University, School of Computer Science, Telecommunications, and Information Systems.

20. Φιτσιλής Π. ΣΧΕΔΙΑΣΜΟΣ ΛΟΓΙΣΜΙΚΟΥ – ΠΛΗ24 ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ ΙΙ: ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ; 2004.

21. Wiegers KE. Software Requirements. Second Edition ed.: Microsoft Press; 2003.

22. Δρανίδης Δ. ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ.

23. Gottesdiener E. Capturing Business Rules. SOFTWARE DEVELOPMENT MAGAZINE: MANAGEMENT FORUM. 1999 December; 7(12).

24. Wiegers K. Vision and Scope Document for Cafeteria Ordering System. ; 2002.

25. Michael G. Christel, Kyo C. Kang. Issues in Requirements Elicitation. Pennsylvania: Carnegie Mellon University Pittsburgh; 1992.

26. Joseph A. Goguen, Charlotte Linde. Techniques for Requirements Elicitation. ; 1993.

27. Ann M. Hickey, Alan M. Davis. Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. Colorado Springs:, University of Colorado.

28. Bashar Nuseibeh, Steve Easterbrook. Requirements Engineering: A Roadmap..

29. Gunda SG. Requirements Engineering: Elicitation Techniques. University West, Department of Technology, Mathematics and Computer Science; 2008.

30. Tuunanen T. Requirements Elicitation For Wide Audience End-Users Helsinki; 2005.

31. Κακογιάννη Ε. Τεχνικές Εκμαίευσης Απαιτήσεων. Τρίπολη:; 2005.

32. Williams L. Use Case-based Requirements. ; 2004.

33. USE-CASE MODEL: WRITING REQUIREMENTS IN CONTEXT. In.

34. Simmons E. From Requirements to Release Criteria: Specifying, Demonstrating, and Monitoring Product Quality. ; 2001.

35. McBreen P. Using Use Cases for requirements capture. ; 1998.

36. E. Astesiano, M. Martelli, V. Mascardi, G. Reggio. From Requirement Specification to Prototype Execution: a Combination of a Multiview Use-Case Driven Method and Agent-Oriented Techniques..

37. Jicheng Fu, Farokh B. Bastani, I-Ling Yen. Model-Driven Prototyping Based Requirements Elicitation. The University of Texas at Dallas, Department of Computer Science.

38. Wu H. Supporting Participatory Requirement Engineering in an ERP Software Community. Norfolk: Old Dominion University.

39. Gabriel RP. Requirements for a Common Prototyping System. ; 2010.

40. Markus Mannio, Uolevi Nikula. Requirements Elicitation Using a Combination of Prototypes and Scenarios. LAPPEENRANTA, FINLAND: Lappeenranta University of Technology, Telecom Business Research Center Lappeenranta.

41. Behrens H. Requirements Analysis and Prototyping using Scenarios and Statecharts. Hagen: Fern Universität Hagen.

42. AAQIB IQBAL, FARHAN M, KHAN, SHAHBAZ, A. KHAN. A CRITICAL ANALYSIS OF TECHNIQUES FOR REQUIREMENT PRIORITIZATION AND OPEN RESEARCH ISSUES. International Journal of Reviews in Computing. .

43. ΑΝΑΠΤΥΞΗ ΣΥΝΑΡΤΗΣΗΣ ΠΟΙΟΤΗΤΑΣ – QUALITY FUNCTION DEPLOYMENT. In.

44. Faulk SR. Software Requirements: A Tutorial..

45. Smith A. BUSINESS REQUIREMENTS SPECIFICATION. ; 2008.

46. Donn Le Vie J. http://techwhirl.com/. [Online].; 2010 [cited 2012. Available from: http://techwhirl.com/writing-software-requirements-specifications/.

47. T.Y. Chen, Pak-Lok Poon, Sau-Fun Tang, T. H. Tse, Yuen Tak Yu. Applying Testing to Requirements Inspection for Software Quality Assurance. Iformation Systems Control Journal 6. 2006.

48. Lulu He, Dr. Jeffrey C. Carver, Dr. Rayford B. Vaughn. Using Inspections to Teach Requirements Validation. The Journal of Defense Software Engineering. 2008 January.

49. January. In Life Cycle Management System For Construction.

50. A. Terry Bahill, Steven J. Henderson. Requirements Development, Verification, and Validation Exhibited in Famous Failures. ; 2004.

51. Grehag Å. Requirements Management in a Life Cycle Perspective – A Position Paper. SWEDEN: University of Skövde, Department of Computer Science.

52. Engineering B. Burns Quality Manual Revision I..

53. Ryan A. Carter, Annie I. Antón, Aldo Dagnino, Laurie Williams. Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model..

54. Organization EC. Guidance for Configuration Control Board Charter Development and Submission. ; 2006.

55. Change Management Process. ; 2005.

56. Love B. Operational Change Control Best Practices..

57. Orlena C. Z. Gotel ,Anthony C. W. Finkelstein. An Analysis of the Requirements Traceability Problem. Imperial College of Science, Technology & Medicine.

58. Leffingwell D. The Role of Requirements Traceability in System Development. ; 2002.

59. Winkley M. VoteCal Statewide Voter Registration System Project..

60. Wiegers KE. Automating Requirements Management. ; 1999.

61. Rajat R. Sud, James D. Arthur. Requirements Management Tools A Qualitative Assessment. Virginia Tech, Department of Computer Science.

62. Kevin Daimi, Luming Li, Xiaodan Lu, Nazar El-Nazeer. Requirements Engineering for Home Health Care Software Systems. Detroit: University of Detroit Mercy, Department of Mathematics and Computer Science.

63. Patrik Berander, Anneliese Andrews. Requirements Prioritization. In.